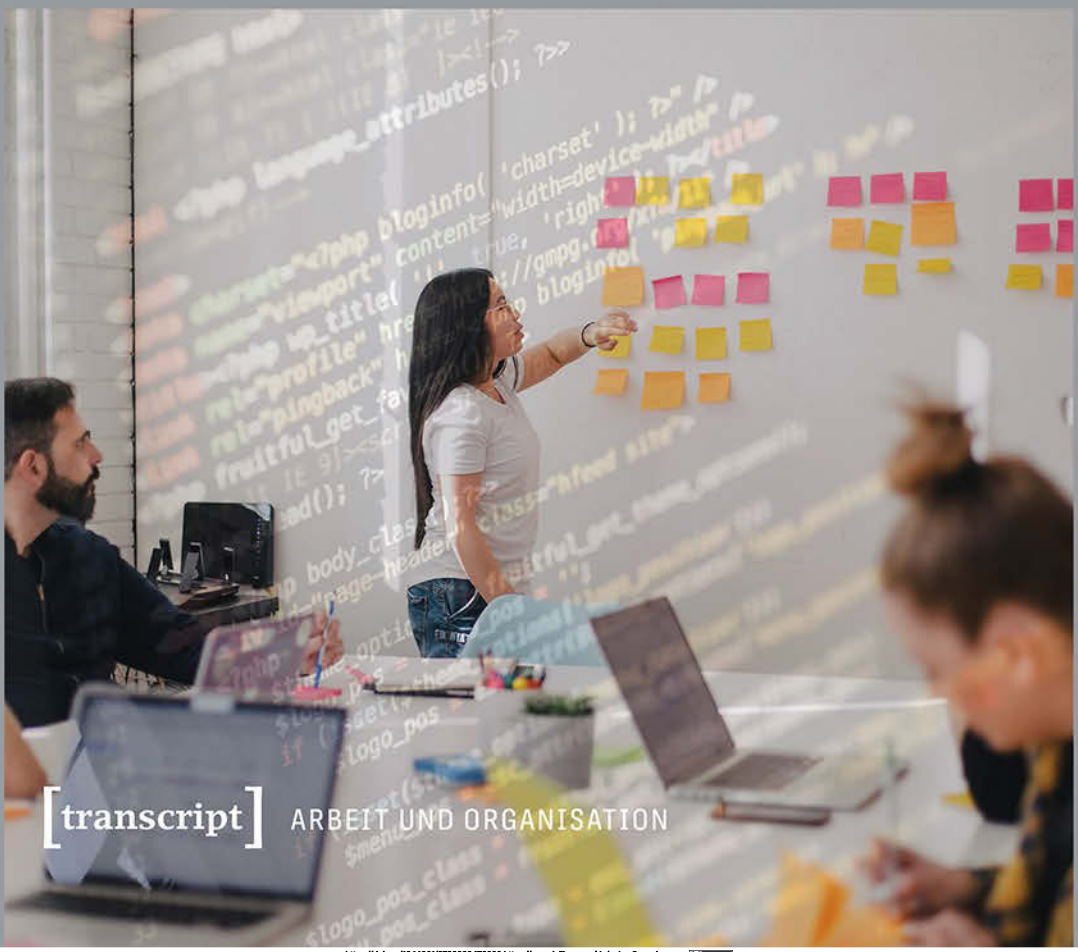
Johannes Sonnenholzner

FORMEN UND FOLGEN DER SOFTWARE-GESTALTUNG

Digitale Transformation am Beispiel der deutschen Energiewirtschaft



Johannes Sonnenholzner
Formen und Folgen der Softwaregestaltung

Editorial

Die Reihe **Arbeit und Organisation** bietet theoretischen und empirischen Studien der Arbeits- und Industriesoziologie sowie der Organisations- und neuen Wirtschaftssoziologie eine gemeinsame editorische Plattform. Dabei stehen Themen wie die Digitalisierung der Arbeitswelt, Analysen gegenwärtiger Organisationsentwicklungen und deren Effekte auf Individuum und Gesellschaft sowie Untersuchungen von (alternativen) Wirtschaftsformen, Märkten und Netzwerken im Zentrum. Dies macht einen umfassenden Diskurs sichtbar, der den soziotechnischen und sozioökonomischen Wandel nebst dessen Konstitution und Ursachen zu verstehen hilft. Die Reihe schließt Monographien und Sammelbände ebenso ein wie Qualifikationsarbeiten und längere Essays.

Johannes Sonnenholzner (Dr.), geb. 1982, promovierte am Wissenschaftszentrum Berlin für Sozialforschung (WZB) und der Universität der Bundeswehr Hamburg. Er war sieben Jahre als IT-Berater tätig, hat sein Magisterstudium der Soziologie, Philosophie und Pädagogik in Regensburg, Heidelberg und Louvain-la-Neuve absolviert und ist gelernter Anwendungsentwickler (IHK).

Johannes Sonnenholzner

Formen und Folgen der Softwaregestaltung

Digitale Transformation am Beispiel der deutschen Energiewirtschaft

[transcript]

Dissertation zur Erlangung des Grades eines Doktors der Philosophie der Fakultät für Geistes- und Sozialwissenschaften der Helmut-Schmidt-Universität/Universität der Bundeswehr Hamburg vorgelegt von Johannes Sonnenholzner aus Wasserburg am Inn.
Hamburg 2024.
Erstgutachter: Prof. Dr. Martin Krzywdzinski
Zweitgutachterin: Prof. Dr. Katharina Liebsch

Die Open-Access-Publikation wurde gefördert durch den Publikationsfonds für Open-Access-Monografien der Leibniz-Gemeinschaft sowie durch Publikationszuschüsse des Wissenschaftszentrums Berlin für Sozialforschung und der Hans-Böckler-Stiftung.

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über https://dnb.dnb.de/ abrufbar.

transcript Verlag | Hermannstraße 26 | D-33602 Bielefeld | live@transcript-verlag.de

Umschlaggestaltung: Maria Arndt, Bielefeld
Umschlagabbildung: Collage aus Bildern von Pexels/Pixabay und Jason Goodman/Unsplash
Korrektorat: Anette Nagel, CONTEXTA Lektrorat
Druck: Elanders Waiblingen GmbH, Waiblingen
Print-ISBN: 978-3-8376-7688-4
PDF-ISBN: 978-3-8394-7688-8
Buchreihen-ISSN: 2702-7910
Buchreihen-eISSN: 2703-03261

Gedruckt auf alterungsbeständigem Papier mit chlorfrei gebleichtem Zellstoff.

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AP	Arbeitsprozess
BNetzA	Bundesnetzagentur
BPO	Business Process Outsourcing
BR	Betriebsrat
CRM	Customer Relationship Management
DV	Datenverarbeitung
ERP	Enterprise Ressource Planning
EVU	Energieversorgungsunternehmen
EW	Energiewirtschaft
FB	Fachbereich
FK	Führungskraft
IT	Informationstechnologie
IT-DL	IT-Dienstleistungsunternehmen
PM	Projektmanagement
PO	Product Owner:in
SA	Softwareanwendung
SE	Softwareentwicklung
SF	Softwarefirma
SG	Softwaregestaltung
SLA	Service Level Agreement
SM	Scrum Master:in
ÜNB	Übertragungsnetzbetrieb
VNB	Verteilnetzbetrieb

Danksagung

Zunächst ein herzliches Dankeschön an alle Gesprächspartner:innen von IT-Dienstleistungsunternehmen, Stadtwerken, Energiekonzernen, Start-ups, Softwareunternehmen, Verbänden und anderen Organisationen für ihre Offenheit und die Zeit, die sie sich für die Gespräche genommen haben. Ohne sie hätte diese Arbeit keine empirische Basis.

Vielen Dank an die Hans-Böckler-Stiftung für die finanzielle und ideelle Förderung der Promotion. Zudem danke ich meinem Vertrauensdozenten der Hans-Böckler-Stiftung Prof. Dr. André Bleicher.

Ich hatte das Glück, meine Dissertation am WZB Wissenschaftszentrum Berlin für Sozialforschung schreiben zu können. Dort habe ich hervorragende Rahmenbedingungen vorgefunden. Ein sechsmonatiges Abschlussstipendium des WZB half mir, meine Arbeit finanziell abgesichert abzuschließen. Mein Dank gilt der gesamten Forschungsgruppe »Globalisierung, Arbeit und Produktion« am WZB, insbesondere Dr. Fabio Ascione, Dr. Christine Gerber, Maximilian Greb, Dr. Tatiana López und Dr. Robert Scholz für das Feedback zu einzelnen Textteilen und Samantha Gupta und Eileen Jahnke für alles drum herum. Die geselligen Mittagessen u.a. mit Barbara Schlüter, Dr. Jana Flemming, Dr. Sana Ahmad und meinen netten Bürokolleg:innen Lea Schneidemesser und Or Yosevof waren eine angenehme menschliche Bereicherung des Dissertationsalltags. Auch vom Austausch mit den Kolleg:innen des Weizenbaum-Instituts für die vernetzte Gesellschaft habe ich sehr profitiert. Einen herzlichen Dank an Maren Zychla und dem Open-Access-Team vom WZB für die hilfsbereite Unterstützung im Publikationsprozess.

Prof. Dr. Ulrich Jürgens und Dr. Paul Bauer danke ich für die anfängliche Unterstützung. Dr. Nicole Bögelein danke ich für die begleitende Unterstützung während der gesamten Dissertation und das Feedback zu einzelnen Kapiteln.

Vielen Dank an die zahlreichen Korrekturleser:innen: Julia Campos, Ludwig Filser, Max Franks, Veronika Hager, Eileen Jahnke, Kristin Tröndle, Charlotte von Knobelsdorff.

Vielen Dank an meine Mentoren Dr. Philip Wotschack und Dr. Patrick Feuerstein für die Ermutigung in schwierigen Phasen, die richtigen Worte zur richtigen Zeit und das Feedback zur Arbeit.

Besonderen Dank schulde ich meinem Betreuer Prof. Dr. Martin Krzywdzinski, der die Arbeit von Anfang bis Ende unterstützt hat, mir als Spätberufenem eine Chance gegeben hat und mich großzügig an der Forschungsgruppe hat teilhaben lassen.

1. Einleitung: Softwaregestaltung als Kind von Bürokratie und Rationalismus

> »Die [Software] ist ›rationalen‹ Charakters: Regel, Zweck, Mittel, ›sachliche‹ Unpersönlichkeit beherrschen ihr Gebaren. Ihre Entstehung und Ausbreitung hat daher überall in jenem besonderen, noch zu besprechenden Sinne ›revolutionär‹ gewirkt, wie dies der Vormarsch des Rationalismus überhaupt auf allen Gebieten zu tun pflegt. Sie vernichtet dabei Strukturformen der Herrschaft, welche einen, in diesem Sinn, rationalen Charakter nicht hatten.« (Weber 1980: 578f.)

In diesem manipulierten Zitat aus Max Webers »Wirtschaft und Gesellschaft« von 1921/22 steht »Software« anstelle von Webers Begriff »Bürokratie«. Denn wie die Bürokratie erfüllt auch Software die Forderung nach einer »beschleunigten, dabei präzisen, eindeutigen, kontinuierlichen Erledigung von Amtsgeschäften« (Weber 1980: 562). Sie agiert »entmenschlicht« (ebd.): leidenschaftslos, berechnend; ignoriert alle »irrationalen, dem Kalkül sich entziehenden Empfindungselemente aus der Erledigung der Amtsgeschäfte« (ebd.). Sie ist – wie die »lebende Maschine« (Weber 1988: 835) der Bürokratie und die »leblose Maschine« (ebd.) in der Fabrik – »geronnener Geist« (ebd.).

Die Bürokratie wird hier zum Vergleich herangezogen, weil Weber sie als Trägerin des Rationalismus sieht. In ebendiesem Sinne erledigen heutzutage Software einsetzende Organisationen Aufgaben effizienter als eine rein auf Mitarbeitenden basierende Organisation. Aus rein menschlichen Apparaten sind soziotechnische Organisationen geworden. Sie verdrängen Organisationen, welche die universelle Maschine Computer weniger effizient zu nutzen wissen. So ergibt sich eine intensivierte Rationalisierung, bei der nicht mehr allein die Leistung des Menschen im Fokus steht, sondern die jeweils effizienteste Software für die Datenverarbeitung oder – je nach Industrie – die effizienteste Kombination aus Mensch, digitaler und mechanischer Maschine.

Die Bürokratie dient als Einstieg, weil sie im Zeitalter der Software weiterhin eine Rolle spielt. Es braucht immer noch eine Organisation von Mitarbeitenden, es braucht menschliche Arbeitskraft – ob Software anwendend oder entwickelnd. Für Weber (vgl. Weber 1980: 125f.) ist die Bürokratie eine Organisation u.a. mit entsprechendem Personal (dazugehöriger Persönlichkeitsstruktur und entsprechendem Fachwissen) sowie

definierten Aufgabenbereichen und Arbeitsweisen (Arbeiten nach Regeln, aktenmäßige bzw. schriftliche Aufgabenerfüllung).

Die im Folgenden untersuchten Organisationen aus der Energiewirtschaft unterscheiden sich von der Bürokratie bei Weber natürlich insofern, als Software bei Weber noch gar keine Rolle gespielt hat. Manche dieser Organisationen wenden Software nicht nur an. Sie entwickeln selbst Software oder sind zumindest Teil jener Phase der Softwareentwicklung, in der den Programmierenden gesagt wird, was die gestaltete Software können soll. Diese Phase bezeichnet die vorliegende Untersuchung als Softwaregestaltung. Um Software gestalten zu können, muss es eine interdisziplinäre Zusammenarbeit zwischen Fachleuten der energiewirtschaftlichen Anwendungsbereiche und der Softwareentwicklung geben. Es geht immer um dieselbe Frage: Was sind die branchenspezifischen Bedarfe und wie können dafür die softwaretechnischen Möglichkeiten genutzt werden? Es gilt, das jeweils (zu spezifizierende) Nützliche für einen branchenspezifischen Anwendungsbereich zu programmieren.

Diese Arbeit zwischen Anwendung und Programmierung braucht – wie die Bürokratie bei Weber – Personal, Wissen, Aufgabenverteilung und Regeln, nach denen gearbeitet wird. Die Betriebe müssen sie organisieren. Anders als bei Weber wiederum ist das, was sich zwischen Softwareanwendung und -programmierung abspielt, ein wichtiges Rationalisierungsfeld. Das zeigt sich z.B. in strategischen Fragen wie jener, ob Firmen Wettbewerbsvorteile durch eine individuell gestaltete oder durch eine Standardsoftware gewinnen können.

In der hier untersuchten Energiewirtschaft gibt es zum einen Organisationen, die eine Standardsoftware einsetzen, an die sie sich anpassen müssen. Für solche Softwarelösungen existieren selbst für industriespezifische Prozesse wie die Erstellung von Strom- und Gasrechnungen oder den Handel mit Energie mehrere softwareanbietende Unternehmen. Zum anderen existieren Organisationen, die eigens eine individuelle Software gestalten. So zum Beispiel in der Netzinstandhaltung, um regionale Besonderheiten oder spezifische Gefahrenlagen von Monteur:innen zu berücksichtigen.

Die sieben Fallstudien der vorliegenden Untersuchung liegen zwischen diesen Polen: Energieversorgungsunternehmen (EVU), die sich auf eine industriespezifische Standardsoftware verlassen und diese höchstens noch einstellen, aber selbst gar nicht programmieren. EVU, die eine Standardsoftware mit eigenen Programmierern erweitern, oder solche, die dies in Kooperation mit anderen EVU tun. Untersucht wurde auch ein energiewirtschaftliches Start-up, das von Anfang an darauf ausgerichtet ist, eine eigene, individuelle Software für seine Dienstleistung zu entwickeln und Teile davon als Standardlösung anderen Organisationen zu verkaufen. So bewegen sich die Fallstudien zwischen rein anwendenden EVU und solchen, die letztendlich Softwareentwicklungsfirmen sind – auch wenn sie ihr Geld nicht primär mit dem Verkauf von Software, sondern in der Wertschöpfungskette der Energiewirtschaft verdienen.

Mit den sieben Fallbeispielen aus der Energiewirtschaft zeigt die Untersuchung, wie industriespezifische Softwaregestaltung in verschiedenen Konstellationen aussieht und was sie voneinander unterscheidet. Die Branche eignet sich für diese Fragestellung besonders gut, weil sich durch Energiewende und Liberalisierung viel verändert hat und immer noch verändert und weil die Energiewirtschaft einer starken Regulierung unterworfen ist, wofür Firmen »Paragraphen-Automaten« (Weber 1988: 322) in Form von

(Standard-)Software entwickeln. Es gibt sowohl Organisationen in öffentlicher als auch in privater Hand und aufgrund der vielen EVU einen großen Markt für industriespezifische Softwarelösungen. Eine Organisation ganz ohne Software gibt es auch in der Energiewirtschaft schon lange nicht mehr. Solche softwarebasierten Organisationen und die Softwaregestaltung selbst lassen sich nicht mehr mit dem Bürokratie-Konzept von Weber beschreiben. Deshalb erarbeitet die Untersuchung neue Begriffe und Konzepte unabhängig von Weber.

2. Untersuchungsgegenstand und Gliederung

Wie bereits die ersten Seiten zeigen, bewegt sich die vorliegende Untersuchung zwischen allgemeinen, abstrakteren Begriffen und der konkreten Empirie der Energiewirtschaft. Es geht um beides. Zum einen um eine allgemeine Problemstellung, die alle Industrien betrifft: industriespezifische Softwaregestaltung und welche Konzepte nützlich sind, um diese zu analysieren. Zum anderen um konkrete, industriespezifische Anwendungsbereiche von Software in der Energiewirtschaft, wie die Firmen der Branche Software gestalten und was die Besonderheiten der Branche sind, die sich in der Softwaregestaltung niederschlagen. Sowohl allgemeine als auch spezifisch energiewirtschaftliche Softwaregestaltung sind Forschungslücken und Gegenstand der Untersuchung.

2.1. Forschungsgegenstand und -fragen: Formen und Folgen von Softwaregestaltung

Ganz allgemein ist eine zentrale Eigenschaft von Computern, dass sie als universale Maschine programmiert werden können (und müssen). Alles Mögliche kann als Quellcode entwickelt werden: kleine Apps für Smartphones, Standardpakete zur Textverarbeitung, Programme für den automatisierten Börsenhandel oder Steuerungsprogramme für Atomkraftwerke. Die Folgen der Anwendung von Software auf Arbeit sind vielfältig untersucht (meist in Verbindung mit einer jeweils spezifischen Hardware): ob zu den Softwarelösungen der Gig-Ökonomie (vgl. Wood et al. 2019, Wu et al. 2019), Crowdworking (vgl. Gerber/Krzywdzinski 2019), ERP-Systemen (vgl. Hohlmann 2007, Walker 2016, Howcroft/Richardson 2012) oder in Call-Centern (vgl. Longen 2015). Auch zur Organisation von Softwareentwicklung bzw. der Arbeit der Programmierenden gibt es viel Forschung (vgl. Friedman/Cornford 1993, Barett 2005, Upadhya 2009, Feuerstein 2013).

Beide Forschungsbereiche der Digitalisierung sind in Abbildung 1 schematisch dargestellt: der Arbeitsprozess der Anwendung einer Software (A) und jener ihrer Programmierung (C). Die vorliegende Arbeit untersucht (B), **die Arbeit zwischen Anwendung und Programmierung und wie sie organisiert ist.** Der Arbeitsprozess heißt **Softwaregestaltung**. Wie bereits in der Einleitung angesprochen, erfahren hier die Programmie-

renden, was zu programmieren ist. Diese drei in Abbildung 1 dargestellten Perspektiven auf die Digitalisierung sollten – so die These dieser Arbeit – voneinander unterschieden werden.

Abbildung 1: Drei Arbeitsprozesse digitaler Arbeit

Softwaregestaltung ist die Phase der Konzeption (bzw. Spezifikation oder Anforderungsaufnahme) der Softwareentwicklung. Abhängig vom Fall spielt dabei auch die Phase der Tests eine Rolle, weil in dieser die Anwendenden bzw. Testenden durch ihre Rückmeldungen Input für die Gestaltung geben. Fragen der Bedienoberfläche (Design) oder der Architektur, die Teil der Gestaltung von Software sein können, gehören hier nicht zum Kern des Arbeitsprozesses. Die Softwarearchitektur ist meist eine bereits entschiedene Ausgangsbedingung: z.B. welche Schnittstellen vorhanden sind, welche Einstellungsmöglichkeiten eine Software hat oder welche Erweiterungsmöglichkeiten eine Standardsoftware bietet. Das Software-Design in Sinne von z.B. anwendungsfreundlichen Oberflächen ist in den untersuchten Fallstudien kein gesonderter Arbeitsschritt mit den entsprechenden Spezialist:innen dafür.

Damit steht für die hier vorliegende Untersuchung im Mittelpunkt, dass Software eine gestaltungsoffene Technologie ist, die je nach Umsetzung bspw. zu Formalisierung, Standardisierung, Überwachung oder Automatisierung führt. Das heißt, das Rationalisierungsziel des Softwareeinsatzes ist nicht per se die Formalisierung oder die Sammlung möglichst vieler Daten. Vielmehr geht es darum, für einen Anwendungsbereich die Entwicklungsmöglichkeiten der Technologie auszureizen.

Eine besondere Relevanz hat der Arbeitsprozess zwischen Anwendung und Programmierung bei industriespezifischer Software. Anders als bspw. bei Standardsoftwarepaketen wie LibreOffice oder MS Office ist der Wissenstransfer zwischen Industriefachleuten entscheidend. Organisationen überprüfen regelmäßig Entscheidungen über Make-or-Buy und müssen die Softwarelösungen kontinuierlich anpassen, weil sich die Regulierungen durch den Staat ändern oder um sich im Wettbewerb zu behaupten. Die Forschungsarbeit zeigt, dass der Ablauf industriespezifischer Softwaregestaltung ein eigenständiger, zentraler Arbeitsprozess der Digitalisierung ist und die Arbeit von Beschäftigten der jeweiligen Industrie (hier: der Energiewirtschaft) prägt. Die These ist, dass

für das Verständnis von heutiger Erwerbsarbeit das Verständnis der Softwaregestaltung neben anderen Einflüssen relevant ist: ob Finanzmärkte und der Shareholder Value (vgl. Windolf 2005, Vitols 2002, Dörre 2001, Höpner, 2003), der Position in der Wertschöpfungskette (vgl. Gereffi et al. 2005, Flecker/Meil, 2010, Mezihorak 2018), bestimmte Management- oder Organisationspraktiken (vgl. Gerst 2006, Boes et al. 2018) oder Produktionsmodelle (vgl. Kern/Schumann 1984, Herrigel 2010).

Die Softwareentwicklung als Ganzes ist Teil jeder Branche geworden. Jedes Unternehmen muss sich fragen, auf welcher Seite der Wissensgrenze zwischen Anwendung und Entwicklung es steht und an welchen Phasen des Entwicklungsprozesses – bestehend aus Spezifikation/Konzeption, Programmierung, Test, Support und Betrieb – es beteiligt sein will. Gleichzeitig ist für einen großen Teil der Nicht-Softwareunternehmen und ihrer industriespezifischen Fachabteilungen die Softwareentwicklung mit ihren Werkzeugen und Methoden zunächst einmal etwas Fremdes. Wie die Fallstudien zeigen werden, bedeutet branchenspezifische Software zu entwickeln eine Abkehr von der Fixierung auf eine nach Abteilungen gegliederte hierarchische Organisation und rein marktbasierte Beziehungen zu IT-Zulieferern wie IT-Dienstleistungsunternehmen (IT-DL) oder Softwarefirmen. Intern müssen Teams und Abteilungen (bspw. durch IT-Projekte) und extern unterschiedliche Organisationen kooperieren (bspw. bei der Zusammenarbeit mit Start-ups). Wenn gefragt wird: »Welche Ebene ist bestimmend für die Arbeit unter Bedingungen der Informatisierung: der Betrieb, das Unternehmen, das Projekt, das Netzwerk?« (Baukrowitz 2006: 82), so ist hier die Antwort: keine Ebene alleine, sondern die Softwaregestaltung, die sich über mehrere Ebenen erstrecken kann.

Es geht um die Konstellationen, in denen angewendete Software entsteht und welche Folgen das für die Anwendung hat: ob sie eine Softwarefirma entwickelt, ein eigenes Team an Programmierenden, eine Kooperation mehrerer Firmen; ob ein EVU eine Standardsoftware einsetzt, anpasst oder eine individuelle Lösung gestaltet. Es stehen zwei Kernfragen im Mittelpunkt der Analyse industriespezifischer Softwaregestaltung:

1. **Kontrolle der Softwaregestaltung**

   - Wie kontrollieren Organisationen in unterschiedlichen Konstellationen den Arbeitsprozess der Softwaregestaltung? (Formen)
   - Welche Auswirkungen hat die jeweilige Konstellation und Form der Softwaregestaltung für die Arbeit der Softwaregestaltenden? (Folgen)

2. **Verhältnis zur Softwareanwendung**

   - Welche Auswirkungen hat die jeweilige Konstellation und Form der Softwaregestaltung für die Arbeitsgestaltung der Softwareanwendung? (Folgen)

Analytisch gesehen müssen, um die beiden Fragen nach den Formen und den Folgen untersuchen zu können, **a) Technikgestaltung, b) Arbeitsgestaltung und c) der Kontext beider gemeinsam untersucht werden**. Die reine Anwendung der Software ist nur insofern relevant, als sie diese Zusammenhänge erhellt. Das Methodenkapitel stellt unter 3.2.2 die zu den Kernfragen gehörenden Leitfragen für die Expert:inneninterviews vor.

Softwaregestaltung ist dabei weder primär eine Form von Innovation noch Teil einer Reorganisation. Auch wenn Softwaregestaltung innovativ im Sinn von ›etwas Neues machen‹ ist, so liegt doch der Fokus auf den Gemeinsamkeiten mit anderen Arbeitsprozessen. In der Praxis nehmen Beschäftigte den Bereich nicht als gesonderten Innovationsbereich wahr: Programmierende, Anforderungsmanagende oder IT-Projektleitende sind in den untersuchten Fällen nicht allesamt in einer Innovations- oder F&E-Abteilung angesiedelt (höchstens einzelne Beteiligte). Auch wenn Softwaregestaltung mit organisatorischen Veränderungen einhergehen kann, ist dies in den untersuchten Fällen nicht der primäre Zweck der Software. Genauso wenig wie Innovationen nicht das primäre Ziel sind und in den untersuchten Fällen die Softwaregestaltung nicht per se als »innovativ« gilt – eher die verwendeten Methoden wie z.B. Scrum. Software ist eine schlichte Notwendigkeit, um das Tagesgeschäft (auch in Zukunft) effizient zu bewältigen.

Die analytische Perspektive auf den Arbeitsprozess fokussiert die Frage, womit Firmen und ihre Beschäftigten konfrontiert sind, wenn sie Software gestalten. Einerseits will das Management die Effizienz steigern, andererseits müssen sich Beschäftigte auf neue Methoden wie Scrum einlassen und unterschiedliche Abteilungen und Organisationen miteinander kooperieren. Untersuchungsgegenstand sind dabei einzelne Gestaltungsprozesse von Software(paketen) und nicht eine gesamte IT-Landschaft (aus einer Vielzahl an Softwarepaketen). Zudem geht es beim Arbeitsprozess um die kleinen, iterativen Schritte, zu denen regelmäßige Treffen gehören, um Anforderungen aufzunehmen, oder auch darum, den Anwendenden regelmäßige Updates der Software zur Verfügung zu stellen. Das Ziel ist, die Unterschiede und Gemeinsamkeiten der Formen der Softwaregestaltung in den unterschiedlichen Fallstudien inklusive der jeweiligen Folgen herauszuarbeiten.

2.2. Vorgehen: empirische, qualitativ-explorative Untersuchung der Energiewirtschaft

Die oben genannten Fragen werden **qualitativ-explorativ** erforscht (Näheres dazu im Methodenkapitel). Als empirischer Gegenstand fungiert die Energiewirtschaft und die dort betriebene industriespezifische Softwaregestaltung. Diese Branche ist nicht nur deshalb von Interesse, weil sie in der Soziologie kaum erforscht ist, sondern auch wegen ihrer jüngeren Liberalisierungsgeschichte, ihrer regulierten und unregulierten Bereiche: Die Fallstudien zeigen, dass das Folgen für die Softwaregestaltung hat, denn z.B. ist es im regulierten Netzbereich für die Unternehmen einfacher, sich auf einen Standard zu einigen. Im unregulierten Bereich der Energielieferung, in dem Wettbewerb herrscht, gestalten EVU häufiger individuelle Software, z.B. um ihrer Kundschaft besondere Angebote machen zu können. Durch die vielen und vielfältigen privaten, öffentlichen, großen und kleinen Unternehmen in der Energiewirtschaft gibt es einerseits einen großen Markt für Standardsoftware und andererseits Möglichkeiten der Kooperation zwischen den EVU. Da der Energiesektor insgesamt nicht zu den Vorreitern der Digitalisierung gehört, prallen *digital natives* und *digital immigrants* besonders heftig und ersichtlich aufeinander. Software innerhalb der EVU zu gestalten, ist in so einer Konstellation eine be-

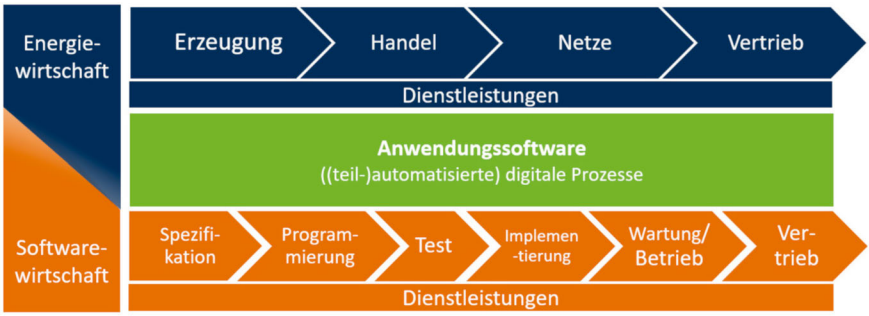
sondere Herausforderung. Zuletzt dient die Energiewirtschaft als Untersuchungsfeld, weil der Verfasser selbst als IT-Berater in der Branche tätig war.

Die vielfältigen und komplexen Anwendungsbereiche von Software in dieser Branche stellen eine Herausforderung für die Softwaregestaltung dar. Umfangreiche Wissensbestände müssen in Software übersetzt werden.

- Sei es technisches Wissen über Kraftwerke oder Netze: um sie zu steuern, zu überwachen oder zu warten. Will ein EVU bspw. seine Netz-Monteur:innen mit mobilen Apps und Drohnen ausstatten, ist ein intensiver Wissensaustausch mit ihnen über deren Arbeitsweise und die technischen Spezifika der Netzwartung notwendig, damit die Programmierenden wissen, was sie zu tun haben.
- Sei es regulatorisches Wissen über Gesetze und Verordnungen, welche den automatisierten Datenaustausch zwischen den Marktteilnehmenden regeln, damit bspw. die Stromkundschaft einfach per Internet das stromanbietende Unternehmen wechseln kann. Jede:r kann einen Blick auf die eigene Stromrechnung werfen, um zu sehen, was dort alles an Daten stehen muss: Abgaben, Umlagen, Steuern, technische Angaben, der Strommix etc. Oder jene Gesetze und Verordnung, welche regeln, wie Erneuerbare-Energie-Anlagen abgerechnet werden. Da kann es sein, dass Jurist:innen bei der Softwaregestaltung mitarbeiten, damit die Software auch das Richtige macht und nichts falsch abrechnet.
- Sei es die Regulierung zu neuen Themen wie dem Emissionshandel, der in Software übersetzt werden muss, damit ihn die Branche effizient betrieben kann.
- Sei es letztendlich Wissen über Marketing, Kund:innenservice oder Rechnungsstellung, die, auch wenn sie in anderen Branchen existieren, ihre Industriespezifika haben und für die deswegen eigene Softwarelösungen entwickelt oder zumindest Standardsoftwarepakete angepasst werden müssen.

Empirisch ist der Arbeitsprozess der Softwaregestaltung ein Teil der **Konvergenz zweier Industrien**. Es treffen zwei Wertschöpfungsketten aufeinander (Software- und Energiewirtschaft), die sich verschränken und meist auf verschiedene Organisationen verteilt sind. In Abbildung 2 sind die beiden Wertschöpfungsketten dargestellt. Jene der Energiewirtschaft setzt sich aus Energieerzeugung, Handel, Verteilung via (Strom-/Gas-)Netze und Vertrieb zusammen. In der Softwarewirtschaft besteht sie aus der Spezifikation, der Programmierung, den Tests, der Implementierung, Wartung, dem Betrieb bis zum Vertrieb von Software. Beide Wertschöpfungsketten arbeiten an bzw. mit der Anwendungssoftware. Aus Sicht der Energiewirtschaft ist Software zentral für die Datenhaltung, -verarbeitung und die Leistungserbringung. Dazu gehören Transaktionen zwischen Firmen und Kundschaft (B2C, Business to Customer) wie auch zwischen Firmen (B2B, Business to Business). In der Energiewirtschaft ist das zu einem großen Teil mit einem hohen Grad an Automatisierung verbunden, weil die vielen unterschiedlichen Marktteilnehmenden viele Massendaten (Strommengen, Geldbeträge etc.) untereinander austauschen müssen. Kapitel 7 geht näher auf die industriespezifischen Aspekte der Softwaregestaltung ein. Es reflektiert die Bedeutung der Industriestrukturen für die Softwaregestaltung und gibt Hintergrundwissen zum besseren Verständnis der Fallstudien.

Abbildung 2: Konvergenz zweier Branchen?



Die Fallstudien beleuchten verschiedene Konstellationen der Softwaregestaltung und vergleichen sie anschließend (kooperativ-firmenübergreifend, intern, durch Softwarefirmen, in einem Start-up etc.). Die Firmen sind jeweils auf unterschiedliche Weise Teil der Wertschöpfung der Softwarewirtschaft und tragen jeweils unterschiedlich zur Softwaregestaltung bei.

2.3. Zusammenhang von Fragestellung, Technologie, Praxis und Theorie

In dem qualitativen Forschungsprozess haben sich aus einem iterativen Prozess der Reflexion von Theorie und Empirie erst im Laufe der Zeit die Kernfragen, zentralen Probleme der Praxis, konzeptionelle Ausrichtung und theoretischen Einsichten ergeben. Ausgehend von der Fragestellung zu den Formen und Folgen des Arbeitsprozesses der industriespezifischen Softwaregestaltung ergeben sich die relevanten technologischen Grundlagen: der Quelltext, die unterschiedlichen technischen Schichten und sprachlichen Strukturierungen, mit denen die Beteiligten auf Software blicken, worauf sie sich in ihrer Arbeit beziehen und welche sie für die Kommunikation nutzen (Quelltext, Bedienoberfläche, Einstellungsmöglichkeiten, Schnittstellen, Modelle, Algorithmen etc.). Fragestellung und Technologie werfen in der Empirie bzw. den Fallstudien den Blick auf zentrale Mittel der Arbeitspraxis der Softwaregestaltung und typische Probleme.

Sowohl die Software selbst als auch ihr Anwendungsbereich kann sehr komplex sein und die Beteiligten müssen sich trotzdem auf eine Gestaltung einigen, weswegen Wissen und Kommunikation untereinander die zentralen Arbeitsmittel sind.

Als typische praktische Kernprobleme für den Arbeitsprozess der Softwaregestaltung hat die empirische Analyse die **softwaretechnische Interdisziplinarität** und die **softwaretechnischen Gestaltungsmöglichkeiten** identifiziert. Die Fallstudien sind unterschiedliche Beispiele dafür, wie Organisationen diese beiden Probleme lösen. Allgemein stellen sich beide Probleme unabhängig vom Kontext – egal ob ein Start-up oder ein seit Jahrzehnten existierender Industriekonzern Software gestaltet. Für die softwaretechnische Interdisziplinarität müssen die Agierenden das Wissen über die Potenziale der Softwaregestaltung mit dem Wissen über die fachlichen Bedarfe zusammenbringen. Erst der Dialog der Wissensdomänen (IT und Energiewirtschaft) schöpft die Möglichkeiten der Softwareentwicklung aus und überwindet Wissensgrenzen. Rol-

len wie IT-Beratende oder IT-Projektleitende stehen exemplarisch dafür, zwischen IT und (energiewirtschaftlichen) Fachbereichen zu vermitteln. Vor allem bei industriespezifischer Software können Organisationen die Möglichkeiten ihrer Gestaltung erst in Bezug auf den Arbeits-, Organisations- und Branchenkontext ausreizen. Der Begriff der softwaretechnischen Gestaltungsmöglichkeiten betrifft die Gestaltung von Software und von Organisation: Zum einen zeichnet Software aus, dass einerseits Synergien durch die Entwicklung eines Standardsoftwarebausteins, den viele Firmen nutzen, entstehen können. Andererseits kann eine individuelle Software Wettbewerbsvorteile garantieren, indem organisationsspezifisches Wissen einfließt. Zum anderen gibt es die Möglichkeit, die Organisation an einer Standardlösung auszurichten oder selbst Software zu gestalten. Eine ausführliche Beschreibung beider Kernprobleme ist unter 4.2 zu finden.

Mit **soziotechnisch** ist gemeint, dass sich in der Softwaregestaltung Technisches und Soziales »vermengen« (Conrad 2017): ob soziale Organisation und Softwarepaket, Organisation der Anwendung und Organisation der Entwicklung, energiewirtschaftliche Arbeitsaufgaben und technische Softwareanwendung, das Wissen über die Energiewirtschaft und das Wissen über Software(entwicklung), Rollen für IT-Arbeit und Rollen für Anwendungsarbeit. Um eine soziotechnische Arbeitsgestaltung handelt es sich, weil der Arbeitsprozess der Softwaregestaltung nicht nur Software verändert, sondern via Software auch Arbeit gestaltet. Je nach Verhältnis der Arbeitsprozesse von Softwaregestaltung und -anwendung kann dies eine umfassende Reorganisation für die anwendende Organisation bedeuten – vor allem dann, wenn bspw. ein Energieversorgungsunternehmen anfängt, selbst Software zu gestalten, dafür die eigene Organisation ändert und die Anwendungsbereiche (Abteilungen, Teams) reorganisiert.

Eine Kernaussage der vorliegenden Untersuchung ist, dass die Arbeit der Softwaregestaltung die Digitalisierung vorantreibt. Dabei konstituiert sich aus den wissensarbeitenden Softwaregestaltenden und den softwaretechnischen und sozialen Strukturen eine Kontrollform von Arbeit, die Kommunikation und Wissensaustausch ermöglicht. Die analytische Perspektive auf die Softwaregestaltung und wie sie die Softwareanwendung verändert, ist daher jene der **Kontrolle von Arbeit**. Unternehmen kontrollieren Arbeit, weil sie das Transformationsproblem von Arbeitskraft lösen müssen: Anders als bei anderen Produktionsfaktoren ist die Realisierung des Arbeitsvermögens nicht gesichert. Der Betrieb hat aber das Ziel, das Arbeitsvermögen möglichst optimal zu nutzen und an den Zielen der Organisation auszurichten (vgl. Minssen 2017: 301). Wie können Unternehmen sicherstellen, dass Beschäftigte Unbestimmtheitslücken im Arbeitsvertrag nicht in ihrem Interesse ausnutzen und bspw. »Dienst nach Vorschrift« machen? Was im Arbeitsvertrag steht, reicht dafür nicht aus (vgl. Marrs 2010: 331f.).

(Legitime) Herrschaft oder Macht, wie sie z.B. Weber versteht (vgl. Weber 1980: 27), ist dabei nur ein Mittel neben anderen. Die Forschung fasst Kontrolle theoretisch unterschiedlich auf: So kann Kontrolle als die Leitung, Anweisung, Überwachung, Disziplinierung und Belohnung von Arbeit verstanden werden, wozu Bürokratien oder Fließbänder gleichermaßen dienen können (vgl. Edwards 1981). Andere Autoren unterscheiden zwischen Input-, Verhaltens-, Ergebnis-, Clan- und Selbstkontrolle (vgl. Wiener et al. 2016). Wieder andere unterscheiden zwischen persönlicher Kontrolle durch Führungskraft und Kollegenschaft einerseits und unpersönlicher Kontrolle durch standardisierte Arbeits-

prozesse (Bürokratie), Qualifikationen (Professionen) oder Ergebnisse andererseits (vgl. Apitzsch 2010: 89). Wie sich Kontrolle im Fall der Softwaregestaltung am besten konzeptionell fassen lässt, arbeitet die vorliegende Forschungsarbeit heraus.

Dafür wurde nach konzeptionellen Bezügen gesucht, welche die konkreten praktischen Probleme der Softwaregestaltung analysieren helfen und die vorliegende Untersuchung in der Forschungslandschaft verorten. Vor allem die Forschung rund um organisationale, interpersonale und informationstechnische Netzwerke, zu Wissensarbeit und zur Rolle von Software bei der Kontrolle von Arbeit hat sich dafür als hilfreich erwiesen. Der Forschungsstand hat Studien aus der Informatik, der Information Science/Wirtschaftsinformatik und unterschiedlichen Bereichen der Soziologie (Organisation, Technik, Arbeit) berücksichtigt. Insgesamt ist es der Versuch, die Rezeptionssperre der Sozialwissenschaften zu beenden, da diese die IT bisher weitgehend als Blackbox betrachten (vgl. Lenk 2016: 352).

Aus Theorie und Empirie sind konzeptionelle Ergebnisse entstanden. Zentral sind dabei der Analyserahmen und die beiden Konzepte der soziotechnischen Netzwerkarbeit und soziotechnischen Arbeitsgestaltung. Das Konzept der technikentwicklungsbezogenen Rationalisierung unterstreicht die Bedeutung der Softwaregestaltung für Organisationen.

Das erste konzeptionelle Ergebnis der Arbeit ist der **Analyserahmen der Softwaregestaltung**. Er basiert primär auf der Auswertung der empirischen Daten. Er dient dazu, die Fallstudien zu strukturieren und vergleichen zu können. In Abbildung 3 sind die drei zentralen Elemente des Rahmens dargestellt:

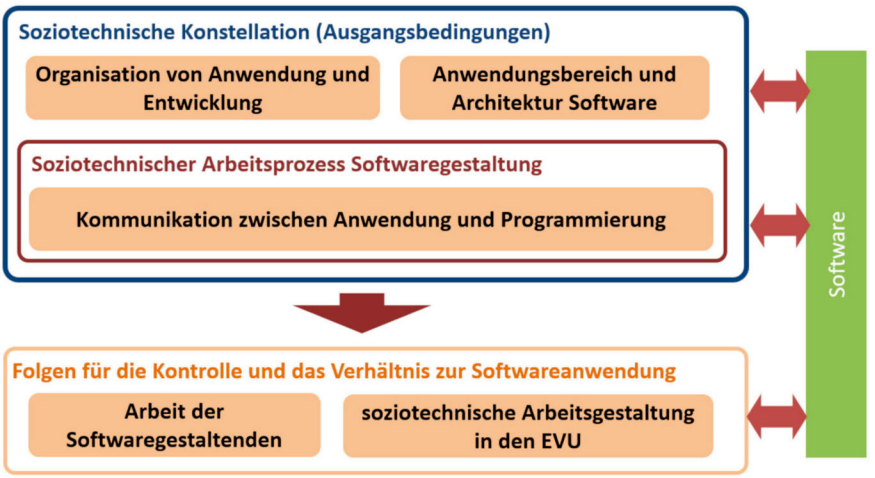
- die soziotechnische Konstellation aus Software (Anwendungsbereich und Architektur) und Organisation von Anwendung und Entwicklung (Gestaltung und Programmierung),
- der soziotechnische Arbeitsprozess der Softwaregestaltung als Kommunikation zwischen Anwendung und Programmierung,
- die Folgen für die Arbeit der Softwaregestaltenden und die soziotechnische Arbeitsgestaltung der Softwareanwendung in den Firmen der Energiewirtschaft.

Alle Elemente des Analyserahmens berücksichtigen, dass es sich um eine soziotechnische Arbeit handelt, wie es für die Arbeit mit Software typisch ist.

Für die Frage nach der Kontrolle der Softwaregestaltung stellt die **soziotechnische Netzwerkarbeit** den konzeptionellen Rahmen dar. Es geht um eine spezifische, netzwerkcharakteristische Form der Kontrolle von Arbeit. Sie ist abhängig von der soziotechnischen Konstellation, in der sie stattfindet. Sie besteht aus dem Arbeitsprozess der Softwaregestaltung und der Arbeit der Softwaregestaltenden. Für beide arbeitet das Empirie-Kapitel typische Unterschiede heraus (siehe 8.6.1.1). So gibt es z.B. Fälle, bei denen sich mehrere Organisationen in einem **zentralen** Arbeitsprozess der Softwaregestaltung abstimmen und die Softwaregestaltenden in einer **Matrixorganisation** arbeiten müssen. Das bedeutet für die soziotechnische Netzwerkarbeit zum einen viel Koordinationsarbeit. Zum anderen besteht die Möglichkeit, viele Perspektiven zu berücksichtigen und einen konsensbasierten Standard zu gestalten. In anderen Fällen gestaltet eine Organisation **dezentral** eine Software für sich selbst. Hier konzentriert sich die Arbeit im Netz-

werk auf das direkte Ausarbeiten der Anforderungen und der Koordinationsaufwand ist gering. Die Softwaregestaltenden arbeiten in einer **reinen Netzwerkorganisation** – ohne formale Hierarchien und Marktbeziehungen. Unabhängig von diesen Unterschieden zeichnet die soziotechnische Netzwerkarbeit aus, dass sie für Feedbackmöglichkeiten zwischen Anwendung und Programmierung sorgt, kooperative Beziehungen auch über Organisations- und Abteilungsgrenzen hinweg schafft und erhält, Softwarewerkzeuge für die verteilte Eingabe von Anforderungen nutzt und Softwaregestaltende erwartungsgeleitet kontrolliert (und z.B. nicht durch die Vorgabe konkreter Arbeitsschritte/-anweisungen).

Abbildung 3: Analyserahmen für die Darstellung und den Vergleich der Fallstudien



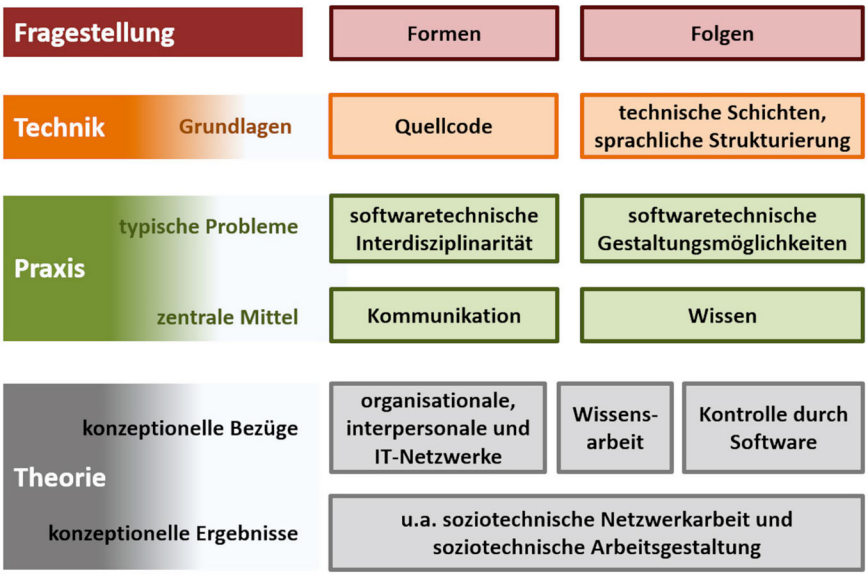
Für die Frage nach dem Verhältnis der Softwaregestaltung zur Softwareanwendung nutzt die vorliegende Untersuchung das Konzept der **soziotechnischen Arbeitsgestaltung**. Dabei unterscheiden sich die Fallstudien, ob sie **unabhängig** oder **abhängig** in der soziotechnischen Arbeitsgestaltung sind (siehe 8.6.1.3). Unabhängig sind sie z.B., wenn die EVU eigenständig eine individuelle Software gestalten. Dann können sie mögliche Konflikte zwischen Anwendung und Gestaltung selbst intern lösen und die Anwendung hat direkt Einfluss auf die Softwaregestaltung sowie die Softwaregestaltung auf die Softwareanwendung. Abhängig sind die EVU, wenn sie eine industriespezifische Standardsoftware verwenden, die eine Softwarefirma gestaltet. Dann müssen sie mit einer externen Firma mögliche Konflikte bei der Softwaregestaltung verhandeln und der Einfluss beschränkt sich primär darauf, wie sie eine Standardsoftware einsetzen, und nicht auf deren Gestaltung.

Mit dem Typ der **technikentwicklungsbezogenen Rationalisierung** grenzt die Untersuchung die Rationalisierung durch Softwaregestaltung von anderen Rationalisierungsformen ab (siehe 6.5.1 und 8.6.3). Es steht z.B. nicht die Arbeitsteilung zwischen Kopf- und Handarbeit im Mittelpunkt wie im Taylorismus, sondern zwischen Softwareanwendung, -gestaltung und -programmierung. Softwaregestaltung kann zudem ein

Mittel für unterschiedliche Zwecke sein (z.B. Automatisierung oder Prozessintegration). Sie rationalisiert die Softwareanwendung nicht nur, indem sie eine Software zur Verfügung stellt. Sie verändert auch die anwendende Organisation, weil diese entweder von einer Standardsoftware abhängt und sich auf die Softwareanwendung konzentriert oder selbst Software gestaltet. Indem Software immer zentraler für die Leistungserbringung von Unternehmen wird, rückt sie in den Mittelpunkt der Arbeit und der Organisation von Unternehmen. Der konkurrenzfähige Betrieb vieler Geschäftsbereiche ist ohne Software gar nicht vorstellbar. Viele Beschäftigte sind über Software in die Wertschöpfung eingebunden. Letztlich lassen sich viele branchenspezifische Prozesse nicht mehr getrennt von der Software(gestaltung) rationalisieren. Dabei muss die Rationalisierung durch Softwaregestaltung mit den Spannungen aus wissensintensiver Kommunikation und Effizienz, aus Innovation und Kostendruck umgehen – vor allem, wenn bestehende Strukturen nicht auf einen interdisziplinären Wissensaustausch ausgelegt sind. Denn auch für die Softwaregestaltung zählen wirtschaftliche Indikatoren: Effizienz, Erhalt und Gewinnung von Marktanteilen und Erzielung von Renditen. Die Unternehmen prüfen regelmäßig, ob es nicht eine bessere, billigere und innovativere Softwarezulieferfirma gibt oder ob sie die Gestaltung und auch Programmierung selbst übernehmen sollten.

Abbildung 4 stellt Fragestellung, Technik, Praxis und Theorie schematisch dar:

Abbildung 4: Übersicht über Kernbestandteile der Untersuchung



Aus der Untersuchung ergeben sich drei Kernthesen, die einen Beitrag zur Arbeits-, Organisations- und Techniksoziologie leisten sollen:

1. Ein zentraler Träger der Digitalisierung ist der Arbeitsprozess der Softwaregestaltung und die Softwaregestaltenden. Die Kontrolle ihrer Wissensarbeit lässt sich als **soziotechnische Netzwerkarbeit** konzeptionieren: erwartungsgeleitet, prozessförmig, beziehungsgebunden, softwarebezogen, netzwerkförmig, wissens- und kommunikationsbasiert und zwischen Anwendung und Programmierung vermittelnd.
2. Der Arbeitsprozess der Softwaregestaltung und sein Verhältnis zur Softwareanwendung stellt eine **soziotechnische Arbeitsgestaltung** dar: Es macht einen Unterschied für anwendende Organisationen wie EVU, ob sie eine individuelle oder eine Standardsoftware einsetzen, ob sie unabhängig gestalten können oder abhängig sind (z. B. von einer Softwarefirma).
3. Softwaregestaltung als **entwicklungsbezogene Rationalisierung** von Technik zielt darauf ab, die Möglichkeiten der Softwareentwicklung (Mittel) für den jeweiligen industriespezifischen Anwendungsbereich (Zweck) effizient zu nutzen.

2.4. Überblick über Kapitel und Argumentation

Die grundlegende Argumentationsstruktur ist wie folgt aufgebaut: 1. Die Softwaregestaltung ist ein eigenständiger und relevanter Arbeitsprozess mit spezifischen Problemen und Mitteln, zu dem weder ausreichend Konzepte noch Empirie existieren (Problem/Forschungsstand). 2. Der Arbeitsprozess der Softwaregestaltung lässt sich als soziotechnische Netzwerkarbeit und in ihren Folgen als soziotechnische Arbeitsgestaltung konzeptionieren (Theorie/Forschungsstand). 3. Die Branchenstruktur der Energiewirtschaft hat Folgen für die Softwaregestaltung und die Branche hat spezifische Anwendungsbereiche für Software (Forschungsfeld). 4. In den Fallstudien zeigen sich unterschiedliche Formen und Folgen der Softwaregestaltung. Der Analyserahmen dient dazu, sie darzustellen und zu analysieren (empirische Analyse). 5. Die Ergebnisse zeigen, dass Softwaregestaltung ein eigenständiger Arbeitsprozess ist, wie ihn Firmen kontrollieren und dass er ein Faktor ist, der erklärt, wie EVU organisiert sind, wie sie arbeiten und wie ein Teil der digitalen Transformation vonstattengeht (Bewertung der Ergebnisse).

Die Argumentation gliedert sich in die folgenden Kapitel: Das 3. Kapitel stellt das Forschungsdesign und die Forschungsmethode vor. Kapitel 4 verweist allgemein auf die veränderte Rolle der Softwareentwicklung in den Organisationen. Die zentralen Probleme in der Praxis der Softwaregestaltung werden dabei genauer ausgeführt. Aus diesen ergeben sich zusammen mit den Besonderheiten der Softwaretechnik (Kapitel 5), der Fragestellung und dem Forschungsdesign die konzeptionellen Bezüge. Diese werden im 6. Kapitel dargestellt und zeigen, dass die Formen und Folgen von Softwaregestaltung an bestehende Forschung anknüpfen und gleichzeitig Probleme darstellen, für deren Verständnis ein Zusammendenken von Software- und Arbeitsgestaltung notwendig ist. Als analytische Grundlage dient dafür der Begriff der soziotechnischen Netzwerkarbeit, bei dem Software in mehrfacher Hinsicht für Kontrolle sorgt und Softwaregestaltende und kooperative Beziehungen auf organisationaler und interpersoneller Ebene eine tragende Rolle spielen. Das Kapitel führt auch den Begriff der soziotechnischen Arbeitsgestaltung ein. Kapitel 7 führt in die Arbeitsweise und Strukturen der Energiewirtschaft ein. Es stellt industriespezifische Anwendungsgebiete von Software und einen Teil der Softwarezulie-

ferindustrie vor. Es beleuchtet das Verhältnis von Software(gestaltung) und Branchenstrukturen. Das 8. Kapitel präsentiert als ein Ergebnis der Untersuchung den Analyserahmen, stellt mit ihm die Fallstudien dar und vergleicht sie. Die Zusammenfassung des Kapitels stellt Bezüge zum Theorieteil her (6. Kapitel), bildet Idealtypen und diskutiert einige Hypothesen. Der Schluss (9. Kapitel) fasst die Untersuchung zusammen, stellt den Beitrag zu zwei arbeits- und industriesoziologischen Debatten dar und zeigt weiterführende Forschungsfragen auf.

Die vorliegende Untersuchung versucht so weit wie möglich auf technische Begriffe zu verzichten. Wo dies unumgänglich ist, sind sie erklärt. Wichtiges empirisches Vorwissen über SAP (S. 22), Scrum (S. 33) und Holokratie (S. 93) liefern kurze Exkurse. Fachbegriffe sind direkt in Fußnoten erläutert.

Da der Text einige Wörter sehr häufig verwendet, sind deren Abkürzungen zu Beginn der Untersuchung aufgelistet. Wichtig ist: Wenn Softwareentwicklung geschrieben steht, dann ist damit immer der gesamte Prozess inklusive Gestaltung und Programmierung gemeint.

3. Forschungsdesign und -methode

3.1. Methode

3.1.1. Qualitative Sozialforschung: Expert:inneninterviews für Fallstudien

Die Forschungsfrage an sich ist bis auf einzelne Themen in der Soziologie (Energiewirtschaft, organisationsübergreifende Arbeitsprozesse, Softwaregestaltung) noch nicht untersucht worden und auch im Zusammenspiel nur spärlich erforscht. Daher hat die Untersuchung einen qualitativ-explorativen Ansatz gewählt. Die Teilbereiche der Arbeits-, Organisations- und Techniksoziologie sollen zusammen betrachtet und die dafür notwendigen Begrifflichkeiten entwickelt werden.

Der qualitative Ansatz erlaubt es, mit der notwendigen Offenheit an die Fragen heranzugehen und dabei zugleich die unterschiedlichen, bereits existierenden Erkenntnisse zu berücksichtigen. Wie in qualitativen Untersuchungen üblich, wird die Komplexität erst später im Forschungsprozess reduziert, was eine sehr umfassende Betrachtung des Feldes ermöglicht. Drei Verfahren wurden angewendet, nämlich Fallstudien, Expert:inneninterviews und Grounded Theory. Das Expert:inneninterview bietet zwei Vorteile, indem es Themen bündelt und Daten evoziert. Im Austausch während des Gesprächs kann der Verfasser seinen eigenen Expertenstatus als ehemaliger Softwareprogrammierer und IT-Berater fruchtbar machen. Die Grounded Theory bietet sich besonders in Bereichen an, zu denen bisher wenig Forschung durchgeführt wurde – eines der Merkmale des hier aufgespannten Feldes. Fallstudien erlauben es, Einzelfälle in ihrer Komplexität zu untersuchen. Dazu gehört der Kontext über die befragten Individuen hinaus inklusive der Branche.

Die Expert:inneninterviews wurden entsprechend der inhaltlichen Ausrichtung der Arbeit als Untersuchungsmethode gewählt. Es geht dabei nicht um »die Gesamtperson […], d.h. die Person mit ihren Orientierungen und Einstellungen im Kontext des individuellen oder kollektiven Lebenszusammenhangs« (Meuser/Nagel 2002: 71). Nicht Biografie oder Deutungsmuster eines Individuums sind im Fokus, sondern die Handlungen von Menschen in einem konkreten Arbeitskontext, also deren »Kenntnisse über Sachverhalte und […] Einschätzungen« (Klemm/Liebold 2017: 309). Der Expert:innenstatus entspricht einer Rolle, die jede Person einnimmt, wenn sie ihrer Arbeit

nachgeht. Es geht um fachspezifisches Wissen bzw. »Betriebswissen« (Meuser/Nagel 2002). Auch Sachbearbeitende sind in ihrem Feld und für ihre Firma Spezialist:innen bzw. Fachmenschen und gelten im Sinne dieser Arbeit als Expert:innen. Expert:inneninterviews komplementieren zudem Fallstudien, weil die Vergleichbarkeit der Interviews gewährleistet wird durch den »geteilte[n] institutionell-organisatorische[n] Kontext der ExpertInnen« (Meuser/Nagel 2002: 81) und »durch die leitfadenorientierte Interviewführung« (ebd.). Bei dieser Methode kann ein Nutzen daraus gezogen werden, dass der Autor selbst als IT-Fachkraft in der Branche gearbeitet hat und somit den Expert:innen auf Augenhöhe begegnen kann: In so einem Fall soll sich der Forschende nach Bogner/Menz (2002: 67), anders als in anderen Interviewmethoden, nicht rein passiv verhalten. Interaktionseffekte sollen vielmehr als produktive Komponenten des Interviewverlaufs begriffen werden. Schätzt der Interviewte die Forschenden als Co-Expert:in ein, hat das Vorteile, wenn sachdienliche Informationen und Aufklärung über Sachverhalte erhoben werden sollen.

> »Wenn der Forscher sein fachlich-inhaltliches Interesse beweist, sein eigenes Wissen einbringt und engagiert diskutiert, ist auch der Befragte zu entsprechendem Engagement bereit und gibt Informationen und Wissen preis, das bei anderen Rolleneinschätzungen und Kompetenzzuschreibungen kaum zugänglich würde« (Bogner/Menz 2002: 51).

Ein hohes fachliches Niveau im Gespräch von Co-Expert:innen erzeugt Daten, die dann »gewinnbringend für detaillierte Sachanalysen« (Bogner/Menz 2002: 52) sind.

Zur Analyse solcher Daten ist die Grounded-Theory-Methode (GTM) besonders geeignet. Sie zählt nicht nur analytische Kategorien und fokussiert sich nicht auf sprachliche Feinheiten wie etwa die objektive Hermeneutik (vgl. Ley 2010). Kruse (2015) hält es bei qualitativen Interviewauswertungen für wichtig, eine »umfassende[…] gesprächs- bzw. textlinguistische[…] Beschreibung der konkreten Versprachlichungen« (Kruse 2015: 374) vorzunehmen. Er berücksichtigt in seinem rekonstruktiv-hermeneutischen Verfahren die »Art und Weise der Versprachlichung von Wirklichkeit« (Kruse 2015: 375). Diesem »(mikro-)sprachliche[n] Analyseansatz« (Kruse 2015: 376) folgt die GTM nicht:

> »Beim Kodieren geht es nicht um das Herausfinden des wahren Sinns, der wahren Be-/Deutung im Einzelfall […]. Das herauszufinden ist nicht Ziel und Anspruch der Kodierungsprozedur in der GTM […]. Die Daten eines Untersuchungsteilnehmers werden vielmehr dazu benutzt, um Vorstellungen über Grundkonzepte, Komponenten, Dimensionen, Bedingungsgefüge, Verlaufsmuster o.Ä. zu entwickeln« (Breuer/Dieris/Lettau 2009: 78f.).

Wenn sich die Interpretation in der Fallbeschreibung – bestehend aus unterschiedlichen Befragten-Perspektiven – bewährt, genügt das. Sie muss nicht den individuellen Sinn des Befragten widerspiegeln. Dadurch reduziert sich der Arbeitsaufwand für die Analyse eines Interviews, wodurch die Methode ermöglicht, größere Textmengen in einem gegebenen Zeitraum zu analysieren. Das Kodieren hilft dabei, Texte sowohl zu strukturieren als auch zu paraphrasieren und zusammenzufassen. Zudem ist die Grounded

Theory der Informatik nicht fremd, wie die Verwendung im *Information Systems Research* zeigt (vgl. Wiesche et al. 2017)[1]. Das lässt sie für die vorliegende Untersuchung besonders fruchtbar erscheinen.

3.1.2. Bezug zu Forschungsstand und Theorieentwicklung

Die GTM ermöglicht es, den umfangreichen Forschungsstand aufgrund der vielfältigen Forschungsfelder wie IT-Outsourcing, Softwareentwicklung oder Folgen von IT für Organisationen und Arbeit zu berücksichtigten, ohne auf die Offenheit von qualitativer Sozialforschung verzichten zu müssen.

Die Grounded Theory kennt unterschiedliche Vorstellungen darüber, ob mit theoretischen Vorannahmen ins Feld gegangen werden sollte. Mehrere Quellen vertreten die Ansicht, dass es nicht zielführend sei, komplett ohne Theorie an Daten heranzutreten (vgl. Breuer et al.: 2009: 58f.). Man spricht hier von »informed grounded theory« (Thornberg 2012). Der Rekurs auf den Forschungsstand ist notwendig und sinnvoll, um nicht bereits gemachte Erkenntnisse zu ignorieren. Anderenfalls könnte eine Erkenntnis redundant sein, »without knowing whether it had already been done, what were the main findings, and what remaining theoretical puzzles and empirical gaps needed to be addressed« (Deterding/Waters 2021: 714). Wichtig ist, dass der Fokus auf den Daten bleibt und nicht auf der Literatur und dass »concept or theoretical idea he or she constructs must be grounded in data« (Thornberg 2012: 54). Offenheit soll so lange wie möglich aufrechterhalten werden: Die Daten sollen aufzeigen, wie sie analysiert werden wollen, und entsprechend darf die Strukturierung nicht zu stark sein. Kruse spricht von »schielender Hermeneutik«, weil mit dem einen »Auge so offen wie möglich geschaut werden muss, mit dem anderen Auge zugleich aber stets theoretisch versiert beobachtet werden muss« (Kruse 2015: 363). Eine rigide Anwendung eines theoretischen Rahmens würde die Untersuchung stark einschränken (vgl. Collins/Stockton 2018: 9).

In der vorliegenden Arbeit wird von konzeptionellen Ergebnissen gesprochen und nicht von Theoriebildung. Theorie wird dabei verstanden als ein Zusammenhang von Ideen, der erklärt, warum Faktoren auftreten und warum sie miteinander im Beziehung stehen (vgl. Gläser/Laudel 2006: 275).

> »Theory is about the connections among phenomena, a story about why acts, events, structure, and thoughts occur« (Sutton/Staw 1995: 378).

Der erarbeitete Analyserahmen stellt zwar die Verbindung zwischen den beobachteten Handlungen und Strukturen her. Allerdings kann aufgrund der Vielzahl der unterschiedlichen Konstellationen, in denen Softwaregestaltung stattfindet, nicht davon ausgegangen werden, dass durch sieben Fallstudien in einer Branche eine Theorie der Software-

1 Wenn auch nicht immer in der vorgesehenen Tiefe: Von 43 untersuchten Studien nutzen die GT 23 %, um eine Theorie zu entwickeln, und bei den anderen werden Modelle oder dichte Beschreibungen angefertigt. Das Memoing wird selbst bei den Theoriearbeiten nur bei jeder zweiten verwendet. Grundsätzlich wenden die Theoriearbeiten aber mehr Methoden an (vor allem offenes Kodieren, axiales Kodieren und theoretisches Kodieren) (vgl. Wiesche et al. 2017).

gestaltung entwickelt werden kann. Der hypothetische Charakter der Studienergebnisse soll durch die Verwendung des Begriffs des Konzepts klargestellt werden.

3.1.3. Selbst-Positionierung

Die verwendeten Methoden müssen es zulassen, dass Vorwissen eingebracht werden kann, denn der Autor war selber in der Softwaregestaltung tätig. Wie bereits erwähnt, lassen dies alle drei Methoden zu. Zudem wurde beachtet, was als Positionierung in der Literatur bezeichnet wird: »Als Forschender positioniere ich mich und werde positioniert; ich beobachte und werde beobachtet.« (Breuer et al. 2009: 30) Am Beginn eines jeden Interviews hat sich der Verfasser als Wissenschaftler positioniert, der nicht nur eine Forschungsarbeit zur Digitalisierung in der Energiewirtschaft durchführt, sondern auch als ehemaliger IT-Berater in der Branche gearbeitet hat.

Der Verfasser war von Januar 2011 bis Juli 2018 als IT-Berater tätig. Zunächst hat er als Trainee die Grundlagen der Branche und die Branchenlösung von SAP für die Versorgungsindustrie kennengelernt. Im Anschluss war er mit der Betreuung und Entwicklung von Software betraut und im Rahmen verschiedener Projekte bei EVU tätig. Letztere waren in zwei Fällen auch längerfristig, d.h., er hat zweimal fast zwei Jahre lang mit den Angestellten im Büro vor Ort gearbeitet. Somit war es kaum möglich, an das Thema unvoreingenommen heranzugehen. Themenwahl, Auswahl der Interviews, Interviewführung und Analyse sind durch seine Erfahrungen beeinflusst. Im Forschungsverlauf fand eine ständige Kontrastierung der gemachten Erfahrungen, der eigenen Alltagstheorien, des soziologischen Zugangs, der Forschungsliteratur, der Aussagen der befragten Personen und der methodischen Auswertung statt. Als Material sind die Erfahrungen des Verfassers jedoch nicht in die Fallstudien eingeflossen.

3.2. Forschungsverlauf

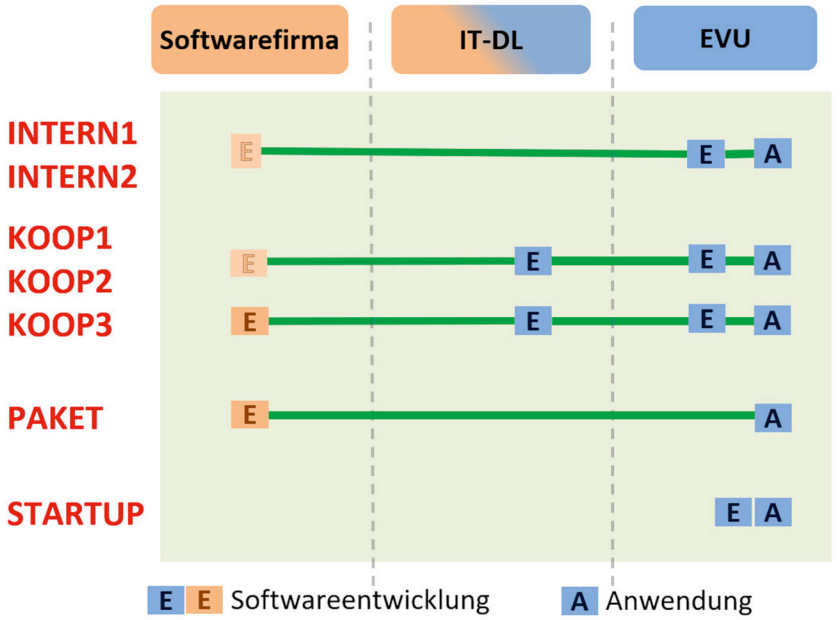
3.2.1. Feldzugang und Sampling

Es wurde versucht, Fälle zu erforschen, die verschiedene Formen der Softwaregestaltung abbilden. Wie diese beschaffen sein sollten und welche Organisationen dafür in Frage käme, wurde ausgehend von der eigenen Branchenerfahrung, regelmäßigem Lesen der Zeitung für kommunale Wirtschaft (ZfK) und der Forschungsliteratur zu IT-Outsourcing und Softwareentwicklung festgelegt. Obwohl der Feldzugang schwierig war, konnten verschiedene Fälle abgedeckt werden: interne Entwicklung in einem Konzern, kooperative Entwicklung mehrerer EVU, die Entwicklung durch eine Softwarefirma, die Entwicklung in einem Start-up, große (> 5000 Mitarbeitende) und kleine (≤ 800 Mitarbeitende) EVU. Letztendlich lassen sich die Fälle danach sortieren, wie sich die Softwareentwicklung verteilt: ob sie durch mehrere Organisationen wie Softwarefirma oder IT-Dienstleistungsunternehmen (IT-DL), außerhalb oder innerhalb der EVU stattfindet.

In der folgenden Abbildung 5 sind die Bezeichnungen der Fallstudien rot markiert. INTERN1 und INTERN2 sind Fallstudien, bei denen einen Teil der angewendeten Soft-

ware eine Softwarefirma entwickelt hat (SAP). Einen anderen Teil entwickeln die anwendenden EVU, weshalb in der Spalte »EVU« die Kästchen »E« für Entwicklung und »A« für Anwendung platziert sind. Ein Kästchen mit »E« ist heller, weil diese Softwareentwicklung (bei SAP) nicht Teil der Analyse ist. Die grüne Linie steht für den Austausch zwischen Organisationen oder Teams bzw. Abteilungen. Bei KOOP1, KOOP2 und KOOP3 entwickeln sowohl eine Softwarefirma als auch ein IT-DL und einige EVU. Dabei ist nur im Fall KOOP3 auch die Entwicklung in der Softwarefirma Teil der Fallstudie ist. Bei PAKET entwickelt die Softwarefirma und die EVU wenden die Software nur an. Bei STARTUP finden Anwendung und Entwicklung in einer Organisation statt und Abteilungsgrenzen existieren nicht.

Abbildung 5: Übersicht Fallstudien



Für die Ansprache von Firmen und potenziellen Interviewpartner:innen wurde ein Handzettel entworfen und an versendete Nachrichten angehängt. Zusätzlich wurde seit Mitte 2020 immer noch ein Artikel mitversendet. Diesen hat der Verfasser für die Zeitung für kommunale Wirtschaft (ZfK) geschrieben (vgl. Sonnenholzner 2020). Das Ziel des Artikels war, Aufmerksamkeit zu erregen. Allerdings hat sich nur ein Berater gemeldet, mit dem es zu keinem Interview kam. Über den Verband kommunaler Unternehmen (VKU) ergab sich zwar die Teilnahme an einem Treffen mit IT-Abteilungsleitenden aus EVU – aber ebenfalls keine Interviews. Auch über die Gewerkschaften Verdi und IGBCE konnten zwar Erstgespräche mit Personalvorständen arrangiert werden, aber leider keine weitergehenden Interviews. Erfolgreicher war die Vermittlung von Verdi an einen Betriebsrat eines größeren EVU, die zu mehreren Interviews führte. Dreimal war ich auf Kongressen: Bundesverband der Energiemarktdienstleistungsunternehmen

(BEMD), Bundesverband Informationswirtschaft, Telekommunikation und neue Medien (bitkom) und eines kommunalen IT-DL. Es ergaben sich nur kurze Gespräche – aber keine Interviews.

Aufgrund des zähen Erfolges über Gewerkschaften und Netzwerkpartner:innen wurde parallel und verstärkt vor allem ab 2020 eine Internetrecherche zu Firmen durchgeführt. Zuerst hat der Verfasser einzelne Firmen (Stadtwerke größerer Städte, bekannte Softwarezulieferfirmen) per E-Mail angeschrieben. Daraus ergaben sich zwar Kennenlerngespräche, aber keine Interviews. Mit der Zeit wurden massiv die sozialen beruflichen Netzwerke (Xing und Linkedin[2]) genutzt. Zwar hat nur ein geringer Anteil auf Kontaktanfragen reagiert, doch kam es dadurch zu vielen Interviews.

Die Sampling-Strategie bestand grundsätzlich darin, möglichst viele Interviews zu einem Fall aus verschiedenen Perspektiven auf die Softwareentwicklung zu bekommen (Anwendende, Programmierende, Gestaltende, Führungskräfte). Der Fokus lag auf den Handelnden zwischen Anwendung und Entwicklung, die Softwaregestaltung betreiben (bspw. IT-Projektleitung, Scrum Master, Product Owner, Anforderungsmanagement). Dadurch wurde der Forschungsgegenstand »facettenreich erfasst« (Merkens 2012: 291): sowohl in Bezug auf die Formen der Softwaregestaltung als auch auf die Perspektiven auf den jeweiligen Arbeitsprozess. Teilweise kamen die Interviewkontakte über die »Schneeballmethode« (ebd. 293) zustande und es entstanden »geklumpte Stichproben« (ebd.) durch die persönlichen Netzwerke in einer Organisation. Gleichzeitig wurden Interviews außerhalb dieser Netzwerke geführt, weil über die sozialen beruflichen Netzwerke Kontakte zustande kamen.

3.2.2. Ausgangsforschungsfragen und letztendliche Leitfragen

Die Ausgangsfrage des Dissertationsprojektes war jene nach digitalen Wertschöpfungsnetzwerken in der Energiewirtschaft. Der Arbeitstitel lautete »Governance und Arbeit in den Organisationsnetzwerken der digitalen Energiewirtschaft«. Ausgehend von Untersuchungen zu den Folgen von Arbeitsverlagerungen von Mezihorak (2018) und Flecker/Haidinger/Schönauer (2013) stellten sich folgende Forschungsfragen hinsichtlich der digitalen, netzförmigen Wertschöpfungsketten der Energiewirtschaft:

1. Welche Formen von Governance sind dort zu finden?
2. Wie verortet sich in ihr welche Art der Arbeit mit IT-Systemen?
3. Wie verändert sich der Arbeitsprozess in ihr?

Nach Gläser/Laudel (2006) wurden für diese Leitfragen Unterfragen entwickelt. Erstere fragen nach »Beziehungen und Vorgängen im Untersuchungsfeld, nach Merkmalen von Individuen, Gruppen, Organisationen usw.« (Gläser/Laudel 2006: 89). Der Forschungsstand konzentrierte sich zunächst nur auf Energiewirtschaft und Netzwerkarbeit, wurde im Verlauf dann aber mehr und mehr auf IT und Software ausgeweitet, was dann auch

2 Die beiden Netzwerke wurden auch angefragt, ob es die Möglichkeit gäbe, die Daten für Recherchen bzw. Auswertungen zu nutzen, was aber beide ablehnten.

andere Fächer einschloss (bspw. Information Science, Wirtschaftsinformatik, Anforderungsmanagement). Im späteren Verlauf der Forschung verschob sich das Thema immer mehr in Richtung Softwareentwicklung. Das lag zum einen am Feldzugang (siehe oben). Zudem fiel im Lauf der Zeit auf, dass eine Forschungslücke dahingehend besteht, wie verschiedene Formen der Softwaregestaltung die Arbeitsgestaltung prägen und wie jene Wissensarbeiter:innen arbeiten, die eine tragende Rolle dabei spielen. Als zentrale Leitfragen stellten sich letztendlich folgende heraus, die teilweise auch den Kategorien des Analyserahmens entsprechen:

- Welche Rolle spielt die **Organisation von Anwendung und Programmierung** für die Softwaregestaltung? (Arbeitsteilung, Koordinationsformen)
- Welche Rolle spielen die **Softwarearchitektur und der Anwendungsbereich** der Software für deren Gestaltung?
- Wie wird in verschiedenen Konstellationen ein **Arbeitsprozess der Softwaregestaltung** zwischen Anwendung und Programmierung etabliert? (Wissensgrenzen überwinden: Rollen, Ablauf, Kommunikation, Werkzeuge)
- Welche **Folgen hat die Softwaregestaltung für die Arbeitsgestaltung**? (Umfang der Digitalisierung, Kontrolle, Beschäftigungssystem etc.)
- **Wer darf wie mitgestalten** an der entstehenden Software? Welche Rolle spielt das Management? (Partizipation)
- Inwiefern sind bei der Gestaltung **industriespezifische Muster** zu erkennen? (am Beispiel der Energiewirtschaft)

3.2.3. Weiterer Forschungsverlauf und durchgeführte Interviews

Es wurden Expert:inneninterviews mit Leitfäden durchgeführt. Der Leitfaden wurde mehrmals angepasst: 1. aufgrund des Fortschritts der Untersuchung und 2. für jedes Gespräch abhängig von der Rolle des Gegenübers im Arbeitsprozess, den offenen Punkten bzw. Unklarheiten aus vorhergehenden Interviews eines Falles, der Interviewdauer etc. Der im Anhang befindliche Leitfaden stellt somit eine Sammlung von Fragen dar. Es wurde versucht, ausführlichere Antworten zu evozieren durch Formulierungen wie »Könnten Sie das bitte noch einmal erklären?«, »Es scheint mir, andere sagen …« (vgl. Martin 2017: 88). Abstrakte Fragen und Zusammenhänge wurden vermieden (vgl. ebd. 91).

Das erste Interview wurde am 20.01.2019 geführt und das letzte am 17.02.2023 (Schwerpunkte waren Mitte 2019, Anfang und Mitte 2020 und Mitte 2021). Die Länge der Interviews war mit 90 Minuten veranschlagt, wobei sich einige Interviewpartner:innen nicht so viel Zeit nehmen konnten. Insgesamt wurden 137 Interviews geführt, für 128 davon konnten Aufnahmen gemacht werden. Letztendlich konnten für die Fallstudien nur 62 verwendet werden. Wie oben zum Thema Sampling beschrieben, war das Ziel, mehrere (mindestens drei) Interviews zu einem Fall zu sammeln. Das ist in sieben Fällen geglückt. Die restlichen Interviews sind Einzelfälle (maximal zwei Interviews) zum Arbeitsprozess der Softwaregestaltung.

Tabelle 1: Anzahl und Zeitraum geführte Interviews je Fall

Fall	Anzahl	w	m	Erstes	Letztes
INTERN1	6	1	5	14.07.2020	16.05.2022
INTERN2	8	1	7	02.07.2020	03.12.2021
KOOP1	17	3	14	17.03.2020	20.05.2022
KOOP2	12	4	8	21.04.2020	01.03.2022
PAKET	12	3	9	01.04.2020	15.07.2022
KOOP3	4	0	4	30.03.2020	22.10.2021
STARTUP	3	1	2	04.10.2022	17.02.2023
Summe	62	13	49		

Von den 62 Interviews wurde eines per E-Mail geführt. Die anderen 61 wurden aufgenommen und transkribiert. Nur zwei Interviews fanden persönlich statt – der Rest telefonisch oder per Video-Konferenz. Aus methodischer Sicht können E-Mails bei Leitfadeninterviews sogar Vorteile bieten (vgl. Schiek 2022), ebenso wie telefonische gegenüber solchen von Angesicht zu Angesicht (vgl. Schulz & Ruddat 2012). Es gab nur einen Fall, in dem die befragte Person merklich zurückhaltend war und nur sehr allgemein gesprochen und wenig Konkretes über die eigene Arbeit gesagt hat. Die Fallbeschreibungen zu INTERN1 und STARTUP wurde jeweils von einer befragten Person gelesen und ohne Korrekturen als zutreffend eingeschätzt. Dieses Verfahren, um die Güte der qualitativen Sozialforschung zu steigern, wird »kommunikative Validierung« (Steinke 2012: 320) genannt.

Ab Februar 2020 wurde ein Forschungstagebuch geführt, wo ich sowohl theoretische als auch empirische Notizen (Überlegungen, Beobachtungen etc.) gesammelt habe. Ebenso wurden die Interview-Situationen protokolliert und in einer Datei festgehalten (inkl. Auffälligkeiten und offene Fragen).

3.3. Kodierung, Kategorisierung und Fallvergleich

Es wurde mit verschiedenen Kodierungsformen experimentiert. Ganz zu Beginn wurde line-by-line Coding (vgl. Charmaz 2014: 343) probiert, was zu viele Kategorien hervorbrachte.

Friese (2016) weist darauf hin, dass es Corbin und Strauss darum ging, dass Codes nicht nur Dinge beschreiben sollen, sondern eine Interpretation und Gruppierung auf höherer Ebene darstellen (vgl. ebd. 487). Dadurch entstehen nicht so viele Kategorien. Sie meint zudem, statt »kodieren« wäre »taggen« als Begriff besser, weil es mehr um Etikettieren oder Identifizieren geht (vgl. ebd. 490). Zentraler Teil ist neben dem »Taggen« verschiedener Interviewstellen für sie das analytische Schreiben (vgl. ebd.). Das ist in der vorliegenden Arbeit beim Text für die Fallstudien angewendet worden.

Dabei ist die Besonderheit von Expert:inneninterviews zu beachten:

> »Anders als bei der einzelfallinteressierten Interpretation orientiert sich die Auswertung von ExpertInneninterviews an thematischen Einheiten, an inhaltlich zusammengehörigen, über die Texte verstreuten Passagen – nicht an der Sequenzialität von Äußerungen je Interview« (Meuser/Nagel 2002: 81).

Es wurde das versucht, was die Grounded Theory ausmacht: offen zu bleiben, nahe an den Daten zu bleiben, Codes einfach, präzise und kurz zu halten, Handlungen in den Fokus der Codes zu nehmen, Daten untereinander zu vergleichen (vgl. Charmaz 2014: 120). Aus der Codierung entstand ein Kategorienbaum in MAXQDA. Er besteht aus fünf übergeordneten Kategorien:

- »Verteilte Kompetenzen und Funktionen/Arbeitsteilung/Spezialisierung«
- »Verhandlung generisch/individuell, integriert/segregiert«
- »Kooperation durch gemeinsames Wissen + etablierte Kommunikation«
- »Rahmen zur Integration von Software(-entwicklung)«
- »Subjektivierung (Möglichkeiten und Erwartungen des Einzelnen)«

Die Unterkategorien und -codes dienten dazu, für jeden Fall die zugeordneten Textabschnitte in Excel zu exportieren, dort zu paraphrasieren und entsprechend den (übergeordneten) Kategorien in eine Textdatei für die Fallbeschreibung zu kopieren. Letztendlich wurden die Kategorien, wie sie nun im Analyserahmen festgehalten sind, erst ausgearbeitet durch die Paraphrasierung aller als wichtig erachteten Interviewstellen, deren Zusammenführung in einen Fall und durch die das im Fall entwickelte fallspezifische Narrativ.

> »In reading each case, the researcher will develop an idea of the important concepts and their linkages in the data— provisional answers to the ›how‹ and ›why‹ questions at the center of the research.« (Deterding/Waters 2021: 727)

Es war ein Prozess, in dem immer wieder zwischen den Kategorien in MAXQDA, einzelnen Fallstudien und Theorien gedanklich hin und her gesprungen wurde.

> »When passing through multiple cases in the first round of reading and coding, we recommend compiling a list of concepts and relationships between them that appear to describe multiple cases and begin describing the contours of relationships in thematic memos.« (Ebd.)

Das wurde für mehrere Fallstudien durchgeführt. Peu à peu wurden über den Vergleich verschiedener Fallstudien die Reflexion der Theorie und unter Berücksichtigung der Fragestellung die Kategorien weiterentwickelt. Ergebnis dieses Prozesses sind drei Oberkategorien (soziotechnische Konstellation, Arbeitsprozess Softwaregestaltung, Folgen) mit den jeweiligen Unterkategorien und der Analyserahmen in seiner jetzigen Form. Auf diese Weise konnte ein Rahmen gewonnen werden, um die Fallstudien zu strukturieren und den Arbeitsprozess der Softwaregestaltung von seinen Ausgangspunkten und seinen Folgen klar analytisch unterscheiden zu können.

Als Erstes wurde für INTERN2 und KOOP1 versucht, einen möglichst kohärenten Aufbau der Fälle zu erstellen. Dann kamen die anderen Fälle, wobei auch INTERN2 und KOOP1 immer wieder leicht angepasst wurden. Zuletzt wurde STARTUP angefertigt. Abschließend wurde weiter an den jeweiligen Besonderheiten der Fälle gearbeitet. Im weiteren Verlauf (Überarbeitung der einzelnen Kapitel der Dissertation, wiederholtes Lesen der Fallstudien) kam nichts mehr zum Vorschein, was gegen den Analyserahmen gesprochen hätte. Zuletzt wurden die Unterschiede zwischen den Fällen herausgearbeitet und aus diesen Idealtypen konstruiert. Im Sinne Webers bezieht sich ein Idealtyp zwar »auf empirische Phänomene, beschreibt sich aber nicht einfach, sondern übersteigert einige ihrer Merkmale, um zu einem Modell sozialer Wirklichkeit zu gelangen.« (Kelle/Kluge 2010: 83)

3.4. Grenzen der Untersuchung

Erstens kann nicht ausgeschlossen werden, dass es ein Bias hinsichtlich funktionierender und aus der Perspektive der Beschäftigten positiver Arbeitsbedingungen gibt. Im Laufe der Feldforschung fiel auf, dass sich vor allem jene Personen und Organisationen zur Verfügung stellten, bei denen die Softwareentwicklung und der Softwareeinsatz positiv wahrgenommen werden. Nur vereinzelt gab es Gespräche, bei denen über IT-Projekte gesprochen wurde, die schiefgelaufen sind. Es blieb in diesen Fällen dann bei Einzelinterviews. Da viele Fallstudien über einzelne Personen gewonnen wurden, war es von deren Offenheit und Bereitschaft abhängig, sich Zeit für Interviews zu nehmen (nur bei INTERN1 und INTERN2 kamen die Kontakte innerhalb der Organisation über den Betriebsrat zustande). Der Verdacht besteht, dass damit immer auch eher offene persönliche Netzwerke genutzt wurden. Womöglich sind dadurch Arbeitsumfelder mit schlechteren Arbeitsbedingungen aussortiert worden. Das heißt aber nicht, dass die Befragten nicht einzelne Aspekte ihrer Arbeit kritisiert hätten (bspw. die Zusammenarbeit mit den IT-DL oder Softwarefirmen).

Zweitens kann die Professionalität der Softwareentwicklung nicht gemessen werden. Es werden zwar in einzelnen Fallstudien aktuelle Methoden wie Scrum angewendet oder sogar spezifische Entwicklungsmethoden wie Domain-driven Design[3]. Gleichwohl wurde nicht der Quellcode geprüft, wie die Anforderungen geschrieben sind, ob sie klar und verständlich sind, ob sauber dokumentiert wurde oder inwiefern objektorientiert programmiert wurde.

Drittens wurde das Wissen der Befragten nicht differenziert erfasst. Es ist immer noch eine offene Forschungsfrage, welche Wissensbestände entscheidend für die softwaretechnische Interdisziplinarität sind. Nicht in allen Fällen wurde genau erhoben, vor allem wegen der geringen Anzahl der Befragten, wer wie viel von welchem Wissen hat. Meist wurde das Verhältnis von IT- zu Fachwissen abgefragt, welche Schulungen besucht

3 Dabei wird die Wissensdomäne des Anwendungsbereichs in der Programmierung berücksichtigt, z.B. indem Fachbegriffe aus dem Anwendungsbereich in den Quellcode einfließen. Dadurch wird die Kommunikation zwischen Programmierenden und Anwendenden einfacher, weil sie die gleichen Begriffe verwenden.

wurden und welche Ausbildung oder welches Studium die befragte Person absolviert hat. Das gilt auch für den Wissensaustausch: Weder wurden Meetings beobachtet noch Chats oder E-Mails analysiert, wer mit wem welches Wissen austauscht und welches eigene Wissen einbringt.

Letztendlich erhebt die Arbeit keine quantitativen Zahlen und kann keine Aussagen darüber treffen, wie viele EVU intern Software programmieren oder gestalten – auch nicht für die untersuchten Firmen, weil immer nur ein Teil davon untersucht wurde. Die Generalisierungen in Form konzeptioneller Ergebnisse wie die soziotechnische Netzwerkarbeit oder der Analyserahmen müssen sich erst noch in Untersuchungen anderer Branchen beweisen.

3.5. Forschungsethik und Datenschutz

Bei der Befragung wurden die forschungsethischen Grundsätze berücksichtigt. Es gilt, Schaden für die Untersuchten zu vermeiden, deren freiwillige, informierte Einwilligung und der Datenschutz (vgl. Gläser/Laudel 2006: 48). Die Befragten wurden mit »möglichst ausführlichen Information[en] über Ziele und Methoden des entsprechenden Forschungsvorhabens« (DGS & BDS 2017: 2) und zudem über Gefahren und Risiken aufgeklärt (ebd.: 2f.). Dies schließt auch eine Anonymisierung ein (vgl. ebd.: 3). Zudem wurden der eigene Wissensstand und die Vorgehensweise offengelegt (vgl. ebd.: 1).

4. Softwaregestaltung als Teil der Digitalisierung

Vom Werkzeug der Forschung zum Primat der Softwareentwicklung bei Nicht-IT-Unternehmen

Warum ist Softwaregestaltung überhaupt relevant für Organisationen? Dieser Frage geht das Kapitel in einem kurzen historischen Abriss nach. Softwareentwicklung war und ist zentral, um Digitalisierung zu verstehen, und nicht nur die immer größere Bedeutung von Daten oder die reine Softwareanwendung. Software ist zentral für die Wettbewerbsfähigkeit von Firmen, und einige setzen mittlerweile von Anfang an auf Softwareentwicklung als den Kern ihrer Organisation, auch wenn sie keine Software, sondern andere Dienstleistungen oder Produkte verkaufen (vor allem sogenannte digitale Start-ups). Bei ihnen gilt der Primat der Softwareentwicklung. Das ist nicht bei allen Unternehmen der Fall und es gibt eine große Vielfalt an Mischformen. Wie die Softwaregestaltung jeweils organisiert ist, stellt der spätere Empirie-Teil dar. Neben dem Begriff des Primats der Softwareentwicklung führt das Kapitel noch zwei weitere wichtige Begriffe für die Analyse der Softwaregestaltung ein: die softwaretechnische Interdisziplinarität und die softwaretechnischen Gestaltungsmöglichkeiten. Sie stellen die beiden Kernprobleme der Softwaregestaltung dar, tauchen in der weiteren Analyse immer wieder auf und sind Teil des soziotechnischen Begriffsapparates, der zur Untersuchung der Formen und Folgen der Softwaregestaltung dient.

4.1. Primat der Softwareentwicklung in Nicht-IT-Branchen und -Betrieben

Software: Ihre Programmierung, Gestaltung und Anwendung ist Teil der Geschichte der IT-Nutzung. Von der vorwiegend wissenschaftlichen Nutzung in den 50er Jahren von IT zur Automatisierung von Routinetätigkeiten in den 60ern und den ersten PCs in den 80ern. Seit den 90ern nimmt die Netzwerk-Integration stetig zu und die Informationstechnologie hat auch verstärkt restrukturierende Effekte auf gesamte Unternehmen – und nicht nur auf einzelne Tätigkeiten. Informationen können vielen zugänglich gemacht werden und gleichzeitig ist eine zentrale Kontrolle via Daten möglich (vgl. Schwarz/Brock 1998: 70ff.). Autoren sprechen von einer »open network organization«

(Schwarz/Brock 1998: 67ff.) und von »ubiquitous computing«, da der Zugriff und die Kontrolle der Umwelt von überall und zu jeder Zeit möglich sein soll (wearables, collaborations, crowdworking, realtime) (vgl. Cascio/Montealegre 2016, vgl. Hirschheim/Klein 2012).

Software nimmt dabei eine immer größere Rolle ein. Friedman/Cornford (1993) machen drei Phasen der Entwicklung von Computersystemen aus. In der ersten Phase bis Mitte der 60er waren Beschränkungen durch die Hardware prägend und in der zweiten Phase bis in die frühen 80er solche durch Software. Ab der Phase 3 waren die Beziehungen zu den Nutzenden ein Hemmnis (»User Relations Constraints«). Die Nutzenden wurden wichtiger und die Softwareentwicklung musste sie einbeziehen, vor allem weil deren Anforderungen schwieriger zu spezifizieren waren (vgl. Friedmann/Cornford 1993: 325). Bereits Mitte der 70er waren Fehler in der Programmierung selbst nicht mehr das zentrale Problem: »Analysis and design errors were revealed to be far more common than coding errors« (Friedmann/Cornford 1993: 204).

Erst im Laufe der Zeit entstanden Firmen, die sich auf die Softwareentwicklung konzentrierten. Zu Zeiten der Großrechner bis Ende der 80er war Rechen- und Speicherkapazität begrenzt und kostbar. Mit Software wurde nicht das große Geld verdient. Erst mit Firmen wie Microsoft im Konsumentenbereich oder SAP und Oracle im Industriebereich entstanden große Konzerne, deren alleiniges Geschäft im Programmieren von Anwendungen (und Datenbanken) bestand. Für die Industriefirmen bedeutete das, dass Standardpakete zur Verfügung standen, die sie zentral implementieren konnten. Es wurde üblich, abteilungs- und standortübergreifend zu arbeiten – an einem digitalen Prozess, auf einer gemeinsamen, zentralen Datenbank.

In der Geschichte der Stadtwerke München (SWM) zeigt sich, dass sich unabhängig von der Liberalisierung der Energiewirtschaft die IT stetig zu einem großen, integrierten System entwickelte. Im Frühjahr 1979 waren noch 1206 verschiedene, meist selbst gestrickte, nicht miteinander verbundene Programme im Einsatz (vgl. Bähr/Erker 2017: 280). 1995 wurde für 13,5 Millionen D-Mark das Standard-R/2 ERP-System von SAP eingeführt (Bähr/Erker 2017: 327). In der Folge gab es ab Ende 1997 dann eine gezielte IT-Strategie bei den SWM, »die die datentechnische Durchdringung sämtlicher Arbeits- und Geschäftsprozesse umfasste und als integraler Bestandteil des Transformationsprozesses begriffen wurde« (Bähr und Erker 2017: 326).

Die Softwareentwicklung selbst hat sich seither nicht in einigen wenigen Softwarefirmen konzentriert. Das Gegenteil ist passiert. Sie wird immer einfacher: Mehr Rechnerleistung, mehr Speicher und mehr Übertragungskapazität macht mittlerweile auch das Entwickeln in der Cloud möglich und damit nicht nur neue Formen des kollaborativen Entwickelns, sondern auch das Nutzen von Entwicklungsumgebungen und Werkzeugen wie von Amazon (mithilfe von deren Services auf AWS) oder Microsoft (dem Cloud-Angebot Azure). Durch das Internet, mobile Geräte und immer fortschreitende Softwareentwicklungstechnologien (Entwicklungsumgebungen, Bibliotheken, Entwicklungsmethoden) und immer mehr Arbeitskräfte mit Know-how in der Softwareentwicklung (z.B. viele Studiengänge mit Informatikanteil, mehrere Ausbildungsberufe, mehr Absolvent:innen national und international) verbreitet sich der Einsatz von Software: von einzelnen, einfachen Anwendungen für Großrechner zu einer Vielzahl von Anwendungen für Heimcomputer, Geschäftsprozesse, mobile

Endgeräte und Industrieanlagen – ob im Hintergrund laufend oder in Form von Interfaces mit den Nutzenden interagierend. Dabei werden die Softwarepakete und die IT-Landschaften komplexer, mit vielen Schnittstellen, Einstellungs-, Anpassungs- und Erweiterungsmöglichkeiten. Arbeitsmethoden aus der Softwareentwicklung wie Scrum oder IT-Projekte zur Implementierung oder Entwicklung von Software sind fester Bestandteil vieler moderner Organisationen. Dabei ist der Kern der Softwaregestaltung die Datenverarbeitung nach bestimmten Regeln (Algorithmen) und nicht die Daten selbst.

Nun ist für die meisten Organisationen die Software nicht das Produkt, mit dem sie ihr Geld verdienen, bzw. ihr Organisationszweck. Sie ist das Werkzeug, um Geld zu verdienen oder Leistungen zur Verfügung zu stellen. Diese Firmen stellen sich die Fragen: Selbst entwickeln oder entwickeln lassen? Standard[1] anwenden oder etwas Eigenes entwickeln, was einen von der Konkurrenz abhebt? Ist Software ein reiner Kostenfaktor oder gar Teil des Kerngeschäfts?

Ob Allianz, Siemens oder Volkswagen: Sie sind alle keine reinen Softwareentwicklungsfirmen. Doch hat die Allianz erst vor kurzem ihre Strategie wieder aufgegeben, eine Softwarefirma zu werden und ihre selbst entwickelte Software auch anderen Versicherern anzubieten (vgl. Fromme 2022). Siemens erweitert sein Angebot an Software immer mehr, indem es bspw. eine Softwarefirma für 1,6 Mrd. Euro gekauft hat (vgl. Kopplin 2022). Volkswagen versucht mit Tesla Schritt zu halten, das Software als Kern des Autos sieht, und hat seine Softwarekompetenz in der Tochterfirma Cariad konzentriert (vgl. Hägler 2022). Das heißt, für Firmen wird Software nicht nur zum Kern ihrer Organisation (wie ERP-Systeme), sie bilden sich nicht nur um Softwarelösungen herum. Sie nutzen die Software nicht nur als Plattform für die Organisation ihrer Wertschöpfung (vor allem bei den sogenannten Plattformfirmen der Gig Economy, Amazon etc.). Es wird der Kern strategischer Fragen, weil unabhängig vom eigentlichen Kerngeschäft (Versicherungen, Autos, Industrieanlagen etc.) die Softwareentwicklung entscheidend wird, um überhaupt eine konkurrenzfähige Wertschöpfung oder Produkte zu haben.

James Bessen spricht von »competing on complexity« (2022): In softwareintensiven Industrien, in denen Software nicht das Produkt ist, ermöglichen es große, proprietäre Softwaresysteme, sich gegen die Konkurrenz durchzusetzen. Weil Software leicht erweiterbar ist, kann sie immer komplexer werden. Diese Komplexität können die Unternehmen unterschiedlich nutzen: um die Qualität von Produkten und Dienstleistungen zu steigern. Um durch zusätzliche Features von Produkten und Dienstleistungen mehr heterogene Bedürfnisse der Kund:innen zu befriedigen. Mehr Varianten, mehr Auswahl und auch individualisierte Produkte anzubieten (vgl. ebd. 25f.).

> »[F]irms make large investements in systems that combine the advantages of large scale with the advantages of mass customization« (ebd. 45).

Dafür müssten die Firmen allerdings in die Entwicklung eigener, proprietärer Software investieren. Als Vorreiter sieht Bessen Walmart an, die dank ihrer eigenen Software ei-

1 Zum Standard einer Software gehört alles, was nicht für eine einzelne Anwenderorganisation bzw. einzelne Kund:innen einer Softwarefirma programmiert wurde.

ne immer größere Bandbreite an Produkten günstig anbieten konnten. Das Prinzip des »competing on complexity« gelte aber mittlerweile für sämtliche Branchen. 2019 investierten Firmen in den USA 239 Milliarden Dollar in proprietäre Software (vgl. ebd. 29).

Softwareentwicklung kann den Unterschied ausmachen. Um diesen Unterschied hinzubekommen, ist eine entsprechende Organisation notwendig. Es ist nicht nur eine Frage des Geldes. Mit wenig (Personal-)Aufwand können Firmen bereits entwickeln. Es ist mehr eine Frage davon, ob eine Organisation, die nicht auf Softwareentwicklung spezialisiert ist, fähig ist, Teams von Programmierenden einzubinden oder mit größeren Softwarefirmen oder -dienstleistungsunternehmen zu kooperieren.

(Vormals) digitale Start-ups sind ein Beispiel dafür, dass Softwareentwicklung nicht nur eine fertige Anwendung ist, sondern die gesamte Organisation und Strategie betrifft. Ob Fintecs, Amazon, Facebook, Firmen der Gig Economy: Es sind Firmen, die Software für eine bestimmte Anwendung entwickeln und die gesamte Organisation auf den Softwareentwicklungsprozess ausrichten. Sie bilden nicht einfach bestehende Strukturen in Software ab. Es geht darum, kontinuierlich den Zugriff und die Konstruktion der Wirklichkeit via Software zu optimieren (bereits 1992 spricht Christiane Floyd von »Software Development as Reality Construction«). Digitale Start-ups stellen sich die Frage, wie sie das Potenzial der neuen Technologien in einem bestimmten Geschäftsumfeld nutzen können, und denken damit softwaretechnisches und branchenspezifisches Know-how von Anfang an zusammen, um daraus Software entwickeln zu können. Wie Amazon, Google & Co. zeigen, hören sie auch nicht mehr auf damit. Letztendlich können digitale Start-ups als das Ende einer Entwicklung betrachtet werden, in der in ursprünglichen Nicht-IT-Industrien die Softwareentwicklung immer mehr in den Kern rückt: Zuerst auf die IT-Abteilung beschränkt, die einzelne, industriespezifische Anwendungen schreibt. Dann entstehen immer mehr professionelle Softwarefirmen, die kostengünstige Standardsoftware anbieten. Bis letztendlich zum **Primat der Softwareentwicklung** einer Organisation, wo sich seit der Gründung die Wertschöpfung um eine Software herum bildet, die kontinuierlich weiterentwickelt wird. Oder anders ausgedrückt: »Sachverhalte werden von vorneherein als Informationsprozess verstanden, formuliert und modelliert« (Schmiede 2006: 465). Zusätzlich bieten solche Firmen die intern entwickelte Software anderen Unternehmen an. Ein Ingenieur von Amazon formuliert es so:

> »Amazon is a technology company, but its warehouses are a huge laboratory where we develop new technologies to sell to third parties.« (Massimo 2022)

Organisationen die auf dem Primat der Softwareentwicklung basieren, wie digitale Start-ups, nutzen keinen rein technischen Vorteil. Sie nutzen vielmehr den Vorteil, neue soziale Strukturen schaffen zu können, welche die Potenziale der Technik ausschöpfen helfen. Dabei soll der softwaretechnische Vorsprung von Google et al. nicht unterschlagen werden. Aber nur wenige Organisationen wurden von Beginn an unter der Prämisse aufgebaut, alle Tätigkeiten zu prüfen, inwiefern sie mit einer Software bearbeitet und in einer solchen abgebildet werden können. Die meisten (ob in der öffentlichen Verwaltung oder der Wirtschaft) existieren bereits seit längerem. Bestehende Strukturen wurden noch unter anderen technischen Vorzeichen rationalisiert. Sie stammen in ihrer

Grundstruktur aus einer Zeit vor Computer und Internet. So ist die Organisation auf die Produktion von Autos oder die Wartung von Stromnetzen ausgelegt. Die IT-Abteilung ist eine neben anderen indirekten Bereichen wie Personal oder Buchhaltung.

Die Fallstudien werden unterschiedliche Wege zeigen, wie Organisationen, die nicht primär auf die Softwareentwicklung ausgerichtet sind, sich mal mehr und mal weniger auf dem Weg machen, selbst Software zu programmieren.

4.2. Die zwei Kernprobleme der Softwaregestaltung

Traditionelle Nicht-IT-Unternehmen können nicht einfach Start-ups werden und werden es auch nicht. Sie müssen aber für sich die Frage beantworten, wie sie mit ihren bestehenden Strukturen umgehen. Die Untersuchung der Fallstudien im 8. Kapitel und die Aufarbeitung des Forschungsstandes ergaben zwei Kernprobleme der Softwaregestaltung – egal ob die Organisationen ihre bestehenden sozialen Strukturen (ob z.B. zur Bearbeitung von Kund:innenrechnungen oder Anträgen auf Bauförderung) abbilden, verändern oder ganz neu gestalten: die softwaretechnische Interdisziplinarität und die softwaretechnischen Gestaltungsmöglichkeiten. Die beiden Probleme sind deshalb relevant, weil sie die Besonderheit des Arbeitsprozesses der Softwaregestaltung ausmachen und die hier vorliegende Untersuchung darum kreist, wie Organisationen diese beiden Probleme in unterschiedlichen Kontexten lösen und was das für Folgen hat.

4.2.1. Softwaretechnische Interdisziplinarität

Um Software für eine bestimmte Industrie zu entwickeln, ist Wissen über die entsprechende Industrie notwendig. SAP (bekannt für industriespezifische Standardsoftware) ist seit der Gründung 1972 groß geworden damit, in gemeinsamen Projekten mit Industriefirmen das industriespezifische Wissen in Konzepte und in Software umzusetzen (vgl. Siegele/Zepelin 2009: 24, 52ff.). Wissen aus dem jeweiligen branchenspezifischen (zukünftigen) Anwendungsbereich der Software und softwaretechnisches Know-how müssen zusammen gedacht werden, um die Möglichkeiten der Technologie auszuschöpfen. Vom allgemeinen Branchenfachwissen über einzelne industriespezifische Prozesse, über Firmenwissen zu firmeninternen Abläufen bis hin zum individuellen Anwendungswissen eines Arbeitsplatzes mit seinen spezifischen Besonderheiten, an dem Spezialist:innen arbeiten: Die Programmierenden müssen wissen, was sie zu programmieren haben. Wer verantwortlich für die Interdisziplinarität ist, d.h. ob die Fertigkeiten der Softwareentwickelnden ergänzt werden (vgl. Baukrowitz/Boes/Eckhardt 1994) oder andere Fächer am Zug sind (Management, Behavioral Science, VWL) (vgl. Boehm 2006: 25), ist eine Frage, welche die Fallstudien erhellen werden: Es geht nicht nur um das Wissen Einzelner, sondern vor allem darum, durch den Arbeitsprozess der Softwaregestaltung die verschiedenen Wissensträger:innen zusammenzubringen. So oder so nimmt die Herausforderung mit der Spezialisierung der Softwarelandschaft zu: was die Vielzahl an Anwendungen, aber auch deren Umfang anbelangt. Die immer komplexer werdenden Softwarepakete haben ihre eigene Biografie und Pfadabhängigkeiten (vgl. Pollock/Williams 2009: 80ff.), deren Kenntnis oftmals Voraussetzung ist, um sie weiter-

entwickeln zu können. Um all dieses Wissen für die Softwaregestaltung zu mobilisieren, können unterschiedlichste soziale Einheiten zusammenarbeiten: Softwarefirma und Verwaltung, IT-Abteilung und Fachbereich, Start-up und Industriekonzern, Scrum-Entwicklungs-Team und ein Team für Stromhandel. Für viele Nicht-IT-Firmen ist das eine Herausforderung, weil der offene, kommunikative Austausch ebenso wenig zum Organisations- und Arbeitsrepertoire gehört wie die Softwareentwicklung.

Wegen immer komplizierterer Softwarelandschaften und -lösungen und immer komplexerer Industrieprozesse gibt es immer mehr Spezialist:innen, von denen eine Abhängigkeit besteht und auf deren Partizipation man angewiesen ist (das Heer an SAP-Beratenden ist nur ein Beispiel). Auch wenn schon früh in der Forschung darauf hingewiesen wurde, wie wichtig Partizipation bei der Einführung von IT-Systemen ist (vgl. Mann/Williams 1960: 225f.), zeigen die Fallstudien dieser Arbeit, dass in den Unternehmen weniger der Wille ausschlagend ist, die Beschäftigten an der Gestaltung zu beteiligen. Vielmehr geht es darum, an das für die Gestaltung der Software notwendige Wissen bestimmter Beschäftigter zu gelangen. Der Gestaltungsprozess verteilt sich auf all diejenigen, die zwischen Anwendenden und Programmierenden Anforderungen aufnehmen, Softwarepakte anpassen, Verhandlungen über neue Features führen. Hohlmann (2007) spricht in ihrer Untersuchung von Netzwerken der Gestaltung, die über das, wie sie es nennt, Integrationswissen aus Organisations-, Prozess- und Technologiewissen verfügen. Andere Autoren sprechen von einem »institutionalisierten Informationsbruch« (Remer 2008: 162) zwischen IT- und Fachabteilung, den es zu überwinden gilt. Durch die Konzentration der IT-Fachkräfte und -Kompetenzen in einer Abteilung besteht eine Wissensgrenze zu den anderen. Mit dem IT-Alignment gibt es eine eigene Disziplin in der (Wirtschafts-)Informatik, die erforscht, wie IT- und Fachabteilung besser zusammenarbeiten können (darauf geht 6.4.1.1 genauer ein). Dabei geht es z.B. um das IT-Projektmanagement, das aufgrund der fachlichen und technischen Unsicherheiten in Projekten am besten sowohl über fachliches als auch über technisches Wissen verfügen sollte. Gefragt ist die oder der »*hybrid* IT PM with one foot in the IT domain and the other foot in the business domain« (Ko/Kirsch 2017: 316).

Zentral für die Untersuchung ist, wie in unterschiedlichen Kontexten diese Wissensgrenzen überwunden werden. Die dafür zuständigen Beschäftigten und Arbeitsabläufe der Gestaltung stehen dabei im Mittelpunkt. Es handelt sich um eine neue Sorte von Wissensarbeit zwischen Anwendung und Programmierung. Die Wissensarbeitenden verfügen über spezifische Qualifikationen, Arbeitsmethoden und Rollen. Sie sorgen für die kontinuierliche Weiterentwicklung der Software. Sie sorgen für die Konzeptionslogistik, dass Programmierende immer wieder neue Konzepte bekommen, die zu programmieren sind. Die Fallstudien machen deutlich, dass diese interdisziplinäre Wissensarbeit im Kontext der Konkurrenz auf der Ebene von Individuen und Organisationen geschieht. Wirtschaftliche Indikatoren bestimmen Entscheidungen über Einsatz und Entwicklung von Software. Das hat u.a. die Folge, dass es nicht ohne weiteres möglich ist, sich frei von jeglichen Zwängen über die optimale Software auszutauschen. Kooperatives Verhalten und damit eine Basis für einen offenen Wissensaustausch ist zwischen und innerhalb von Firmen nicht selbstverständlich.

4.2.2. Softwaretechnische Gestaltungsmöglichkeiten

Ein zweites zentrales und typisches Problem für den Arbeitsprozess der Softwaregestaltung ist jenes der **softwaretechnischen Gestaltungsmöglichkeiten**[2]. Worin liegt das Problem? Organisationen müssen zwei wesentliche Entscheidungen treffen und den Arbeitsprozess der Softwaregestaltung entsprechend organisieren:

A) Gestaltung der Software: Ein Quellcode kann nur auf einem Computer existieren oder auf einem vernetzten Server, auf den die ganze Welt Zugriff hat. Er kann Teil eines Standards sein, den viele Organisationen nutzen, oder rein individuell und es nutzt ihn nur eine Organisation. Die Softwaregestaltung steht nun vor dem Problem, den jeweiligen **softwaretechnischen Zuschnitt** zu erarbeiten: Soll sie etwas individuell oder als Standard gestalten? Soll sie eine Standardsoftware erweitern oder anpassen?
B) Gestaltung von Arbeit und Organisation: Die anwendende Organisation kann die Software selbst gestalten oder durch andere wie IT-Dienstleistungsunternehmen (IT-DL) oder Softwarefirmen gestalten lassen. Beim Primat der Softwareentwicklung hat sich eine Organisation dafür entschieden, sich organisatorisch auf die Softwareentwicklung auszurichten. Das Problem, sich organisatorisch auf die Anwendung einer (Standard-)Software oder auf die Softwaregestaltung auszurichten, wird hier als **softwaretechnische Ausrichtung einer Organisation** bezeichnet.

Die Fallstudien des 8. Kapitels zeigen konkret, dass es für die Softwaregestaltung entscheidend ist, welche softwaretechnischen Gestaltungsmöglichkeiten in puncto Zuschnitt (Standard oder individuell) und organisatorische Ausrichtung (Anwendung oder Gestaltung) die Organisationen nutzen.

Theoretisch könnte A) bedeuten, dass man weltweit nur noch eine (Softwarebaustein-)Bibliothek mit allen möglichen Funktionalitäten braucht. Dadurch wäre die Softwaregestaltung deutlich weniger aufwendig. Tatsächlich gibt es solche Bibliotheken mit industrieunspezifischen, grundlegenden Softwarebausteinen[3]. Auch eine Standardsoftware wie SAP kann als Versuch gesehen werden, bestimmte Funktionalitäten (bspw. Arbeitsabläufe oder Datenverarbeitungsprozesse) durch identischen Quellcode für alle Organisationen gleich abzubilden – selbst die industriespezifischen. Der softwaretechnische Zuschnitt kann sich aus unterschiedlichen Ursachen ergeben:

- Synergien: Durch generische Teile einer Software oder Standardbausteine für mehrere Anwendungsbereiche (verschiedene Abteilungen, Firmen etc.) oder Programmierungen (einzelne Bausteine) sollen Synergien gehoben werden.

2 Es gibt Ähnlichkeiten zur »Bezugsebene« des Informationsraums (vgl. Boes et al., 2016: 34). Auch wenn die Firmen diese Bezugsebene nutzen, um die softwaretechnischen Gestaltungsmöglichkeiten zu verwirklichen: Der Begriff der Bezugsebene allein würde nicht verdeutlichen, um welche spezifischen Eigenschaften von Softwareentwicklung es geht, die für die Softwaregestaltung besonders relevant sind.

3 Zum Beispiel Java Class Library, C++ Standard Library, React.js, Node.js.

- Institutionell: Die Software wird auf Standards ausgerichtet (bspw. technische oder regulatorische).
- Abgrenzung von Wettbewerbern: Die individuelle Software soll es ermöglichen, sich von der Konkurrenz abzuheben – ob durch mehr Effizienz oder individuelle Angebote für die Kundschaft.
- Anwendungsperspektive: Die Software soll auf die individuelle Perspektive der anwendenden Organisation zugeschnitten werden, bspw. durch die Erweiterung einer Standardsoftware.
- Prioritäten: Die Software bildet nur die wichtigsten Funktionalitäten ab. Das kann bedeuten, dass eine Standardsoftware nur einen bestimmten Umfang hat. Die anwendenden Unternehmen müssen diese dann noch individuell erweitern, um ihre firmeninternen Prozesse darüber hinaus mit Software abarbeiten zu können.

Egal wie der Zuschnitt zustande kommt: Sobald eine Arbeitsteilung existiert und je mehr Organisationen mitgestalten, desto höher wird der Koordinations- und Kommunikationsaufwand, um bspw. festzulegen, was in einen Standard hineinkommt oder ob und wie Synergien gehoben werden oder nicht. Eine Möglichkeit, diesen Aufwand zu umgehen, ist, sich einfach einer Standardsoftware unterzuordnen und seine Organisation und die Arbeit jedes Einzelnen auf diese auszurichten. Das bedeutet aber, potenzielle Möglichkeiten der individuellen Softwaregestaltung und unter Umständen Wettbewerbsvorteile auszuschlagen.

Solche Synergien durch Standardbausteine schaffen im Fall von ERP-Systemen Softwarefirmen. Sie haben es übernommen, generische, für einen Standard relevante Arbeitsschritte und Prozesse zu entdecken und daraus eine Standardsoftware zu entwickeln – was eine aufwendige Verhandlungsarbeit ist (vgl. Pollock/Williams/D'Adderio 2007). Laut Mormann (2016) befördern Softwarehersteller den Glauben, dass Organisationen viel gemeinsam haben, vor allem wenn sie Standardsoftware verkaufen wollen (vgl. ebd.: 110). Es bestünde eine »Gleichheitsunterstellung« von SAP und den SAP-Beratenden: Prozesse in verschiedenen Industrien unterscheiden sich nicht (vgl. ebd.: 158). Eine Standardsoftware will die Softwarefirma möglichst oft verkaufen (»economies of scale«). Wie ein Fall in der Untersuchung hier zeigt (KOOP1), gibt es aber auch die Möglichkeit, dass mehrere Organisationen sich in kooperativen Projekten die Frage stellen, was sie denn gemeinsam haben, um dann zusammen eine Software zu entwickeln. Daraus kann eine umfassende Standardlösung entstehen oder einzelne Funktionalitäten, die über die Cloud abrufbar sind. Auch intern können Organisationen für Prozesse Standardsoftwarebausteine entwickeln, die mehrere Abteilungen mit den gleichen Arbeitsschritten betreffen.

Nicht nur Softwarefirmen oder Nicht-IT-Firmen tun sich schwer, solche Synergien zu erkennen. Auch auf Branchenebene ist es schwierig: Ein Beispiel dafür ist das Erneuerbare-Energien-Gesetz (EEG). Nachdem es die Bundesregierung verabschiedet hat, haben sich über die Jahre hinweg regelmäßig die Einspeisevergütungen für Erneuerbare-Energie-Anlagen geändert. Viele unterschiedliche Energieversorger, Softwarefirmen und IT-Beratungen haben erst eigene Lösungen entwickelt, um diese Anlagen abzurechnen, und diese dann immer wieder angepasst, anstatt zentral eine Lösung zu programmieren. Im Laufe der Zeit stellten einzelne Softwarefirmen Standardlösungen zur Ver-

fügung. Aber ob der Weg über den Markt der effizienteste für die Branche war, ist fragwürdig.

Das zweite praktische Problem der softwaretechnischen Gestaltungsmöglichkeiten B) betrifft die Gestaltung von Arbeit und Organisation: wie sich Organisationen dazu verhalten, dass sie sich so gestalten könnten, wie es optimal aus Sicht der Softwareentwicklung ist, und nicht so, wie es z. B. EVU gewohnt sind: nach Fachabteilungen getrennt für Instandhaltung, Abrechnung, Einkauf, IT-Abteilung, Stromhandel etc. Sich zusammen mit vielen anderen Abteilungen auf eine Software einigen oder eigenständig eine auswählen? Die IT-Landschaft von der bestehenden Organisation aus gestalten oder die bestehende Organisation als softwarebasierte Organisation betrachten und davon ausgehend nach Optimierungen oder Synergien auf Organisationsebene suchen? Sich an einer Standardsoftware ausrichten oder diese anpassen? Wenn Amazon seine Logistiksoftware auch anderen Firmen anbietet, setzt sich dann nicht nur eine zentrale Softwarelösung, sondern auch eine Standardorganisation und -arbeit durch?

Selbst bei einer Standardsoftware stehen die Unternehmen vor der Frage, ob sie sich rein auf die Anwendung konzentrieren oder intern selbst individuelle Anpassungen und Erweiterungen an der Standardsoftware vornehmen: Der Bedarf an betriebsindividuellen Anpassungen ist groß. Die Anzahl an kooperierenden Firmen der Standardsoftwarefirma SAP, die unter die Kategorien »Solution Building« und »Consulting Services« fallen, beträgt 457 in Deutschland und 2796 weltweit[4]. Das ist möglich, weil SAP sich für eine Softwarearchitektur entschieden hat, die neben individuellen Einstellungen auch Veränderungen am Quellcode und das Programmieren von Erweiterungen zulässt.

In vielen Firmen existiert die Mischung aus Standardsoftware und selbst entwickelten Erweiterungen oder Anpassungen. Wie es zu dieser Mischung kommt und sie konkret aussieht, wäre eine weitere Frage. Es mögen Pfadabhängigkeiten sein oder strategische Entscheidungen, die Softwaregestaltung in verschiedene Bestandteile aufzuteilen: einen Teil für Tätigkeitsbereiche der Firmen, in denen durch individuelle Softwareentwicklung ein Vorteil gegenüber konkurrierenden Firmen erzielt werden kann, und einen anderen Teil, bei dem das nicht der Fall ist und daher eine Standardsoftware ausreicht, die viele andere auch verwenden (es bleibt die Möglichkeit, durch eine effizientere Anwendung Wettbewerbsvorteile zu erzielen). Es ist eine wichtige strategische Frage, welche internen Prozesse sich an einer Standardsoftware ausrichten (können) und dadurch »das Differenzierungsmerkmal der Organisation gegenüber möglichen Konkurrenten am Markt verloren geht« (Masak 2006: 245). Grundsätzlich IT nicht als strategisch relevant und austauschbar wie Bürostühle zu erachten, scheint da wenig plausibel.[5]

Die Herausforderungen der softwaretechnischen Gestaltungsmöglichkeiten und Interdisziplinarität sind geringer bei Start-ups bzw. bei Firmen, die von Anfang an auf Softwareentwicklung als Basis ihrer Leistungserbringung zurückgreifen. Wenn der

4 Abgerufen von www.sap.com/partners/find.html am 26.04.2023

5 2003 hat Nicholas Carr den Artikel »IT Doesn't Matter« geschrieben, in dem er argumentiert, dass IT den Unternehmen keine strategischen Vorteile brächte. Sie tendiere dazu, eine austauschbare Standardware (»Commodity«) wie bspw. Seife zu sein. Dies wäre dann der Fall, wenn die Software einfach zu bedienen ist, ohne große organisatorische Veränderungen ausgetauscht werden kann und kein Lock-in-Effekt besteht (vgl. ZDNet Staff 2004).

oben erwähnte Primat der Softwareentwicklung gilt, ist die Organisation bereits auf deren Anforderungen ausgerichtet. Die industriespezifischen Prozesse werden den softwaretechnischen Entwicklungsprozessen untergeordnet und damit die Arbeitsgestaltung komplett selbst in die Hand genommen.

Die beiden Kernprobleme verdeutlichen, wie wichtig es ist, verschiedene Konstellationen der Softwaregestaltung zu untersuchen. Die Firmen der Fallstudien des Empirie-Kapitels wählen unterschiedliche Softwarearchitekturen oder Möglichkeiten von Synergien, Wissensgrenzen verlaufen jeweils anders und haben entsprechend Auswirkungen auf Arbeit und Organisation. Genauso wird sich zeigen, dass es einer besonderen Form der Kontrolle der Softwaregestaltung bedarf, um diese beiden Kernprobleme der Softwaregestaltung zu adressieren.

Exkurs: Entwicklungsplattform von SAP

Die ersten vier Fallstudien (INTERN1, INTERN2, KOOP1, KOOP2) des Empirie-Kapitels verwenden die Entwicklungsplattform von SAP. Es handelt sich um jene der ERP-Version R/3, die SAP in den 90ern entwickelt hat. Für die Entwicklungsplattform ist SAP NetWeaver die technische Basis. Sie ist als offene Plattform konzipiert, d.h. nicht nur für den SAP-Konzern intern, sondern auch für kooperierende Firmen und die Kundschaft (vgl. Siegele/Zepelin 2009: 191). Teil des SAP NetWeaver ist eine eigene, SAP-spezifische Programmiersprache (ABAP) und Werkzeuge, um selbst Änderungen und Erweiterung am Standard vorzunehmen: Entwicklungsumgebung (inkl. Debugger), Ticketsystem (Solution Manager) und Transportwesen, um zwischen Entwicklungs-, Test- und Produktivsystem Softwareänderungen zu transportieren (vgl. Frederick/Zierau 2011: 29ff.). Die Architektur der ERP-Software hat Folgen für die Arbeitsteilung: Außerhalb der ERP-Firma programmieren EVU und IT-DL selbst. Zur betrieblichen Realität der EVU gehört deshalb nicht nur die angewendete Software, sondern eine Test- und Entwicklungsumgebung. Einerseits müssen sich die EVU und IT-DL an die Möglichkeiten halten, die ihnen SAP bietet, und es sich gut überlegen, wie weitgehend sie Anpassungen vornehmen, weil das für sie mehr Aufwand bedeutet. Andererseits eröffnet es Spielräume für die interne IT-Abteilung und externe IT-DL, die Software zu ergänzen und ihre eigenen Softwarelösungen an das SAP-System anzudocken.

Weil Organisationen unterschiedlichster Branchen SAP anwenden, anpassen und erweitern, hat sich ein SAP-Ökosystem aus kooperierenden Firmen und SAP-Beratenden entwickelt. Für die Versorgungswirtschaft (zu der die Energiewirtschaft gehört) gibt es insgesamt 202 kooperierende Firmen in Deutschland (siehe Abbildung unten). 150 davon bieten »Consulting Services« an, zu denen die Programmierung gehört. Zusätzliche Lösungen, die dann über Schnittstellen mit dem ERP-System von SAP verbunden werden, bieten 104 kooperierende Firmen an (»Solution Building«).

Zu dem SAP-Ökosystem gehören u.a. umfangreiche Hilfe-Seiten und Communities im Internet zur ABAP-Entwicklung (z.B. https://community.sap.com/topics/abap).

Tabelle 2: Anzahl kooperierende Firmen SAP allgemein und Versorgungswirtschaft

Kategorie	Anzahl
Kooperierende Firmen mit SAP für die Versorgungswirtschaft in Deutschland	202
- Solution Sales: SAP product and technology advisory and support services	84
- Solution Building: Build solutions on top of, or integrate with, SAP technology	104
- Consulting Services: SAP solution design, development, implementation, and integration guidance	150
- Outsourced Solution Management: Hosting, managing, and running your SAP solutions and IT infrastructure	34
- Global Technology; Global vendors of hardware, databases, storage systems, networks, and mobile computing technology	2
- Education: Learning needs assessment and enablement services	10

(Quelle: SAP https://www.sap.com/partners/find.html, abgerufen am 28.04.2023)

5. Softwaregestaltung basiert auf Wissen und Kommunikation

Um die beiden Forschungsfragen der Formen und Folgen von industriespezifischer Softwaregestaltung zu beantworten, sind nicht nur die beiden oben genannten Kernprobleme der Softwaregestaltung entscheidend (softwaretechnische Interdisziplinarität und Gestaltungsmöglichkeiten). Genauso wichtig für die Praxis sind Kommunikation und Wissen: Software besteht vom Quellcode über die Softwarearchitektur, ihren Einstellungsmöglichkeiten bis zur Bedienoberfläche aus mehreren Schichten. Weil für die beteiligten Menschen eine dieser Ebenen der primäre Arbeitsgegenstand ist (z.B. für die Programmierenden der Quellcode, für die Anwendenden die Bedienoberflächen), müssen sie sich über ihre unterschiedlichen Perspektiven verständigen und ihr Wissen einbringen. Das Wissen über die komplexe Welt aus Algorithmen, Daten, Einstellungsmöglichkeiten, Schnittstellen, fachlichen Prozessen etc. vor und hinter der Softwareoberfläche reicht aber nicht aus. Die Beteiligten müssen sich im Zuge der Softwaregestaltung kommunikativ austauschen können. Für die Softwaregestaltung ist neben der verbalen Kommunikation nonverbale Kommunikation als wissensbasierte Textarbeit in Form von Arbeit an Quelltext, Spezifikationen, Anforderungen, Konzepten oder Dokumentationen notwendig. Wie das Kapitel zeigt, wurde Softwareentwicklung selbst im Laufe ihrer Geschichte in Forschung und betrieblicher Praxis mehr und mehr als Kommunikationsprozess verstanden. Somit ist nicht nur die Anwendung von Software sozial bedingt. Ihr gesamter Gestaltungsprozess ist es. Das Kapitel erläutert die Prämisse der vorliegenden Arbeit, dass anders als bei anderen sozialwissenschaftlichen Ansätzen zum Verhältnis von Mensch und Technik eine Arbeitsteilung zwischen Mensch und Software(entwicklung) besteht: Der Mensch verfügt über das Wissen, kann kommunizieren und damit Software gestalten. Softwaregestaltung ist etwas genuin Menschliches, weil dafür ein sinnhafter Bezug zu Objekten notwendig ist. Damit legt dieses Kapitel die Basis für die darauffolgenden Ausführungen, auch um zu verstehen, warum Wissen und Kommunikation zwei zentrale Begriffe und Elemente der Softwaregestaltung sind. Es geht um die materielle Basis des Arbeitsprozesses.

5.1. Technische Grundlagen: Software als Ergebnis menschlicher Textarbeit

5.1.1. Verarbeiten und verstehen: Arbeitsteilung zwischen Menschen und Maschinen

Was die theoretische Sicht auf Technik anbelangt, geht diese Arbeit von einer klaren Arbeitsteilung zwischen Menschen und Computern aus: 1. Nur der Mensch kann informiert sein, etwas wissen und kommunizieren. 2. Der Mensch ist das soziale Wesen und 3. entwickelt die Software. Der Computer weiß weder etwas, noch ist er informiert, noch stellt er soziale Beziehungen her oder schreibt autonom industriespezifische Software.

Zu 1.: Für die Person, die Daten eingibt, sind die Daten Informationen und sie braucht ein bestimmtes Wissen, um die Eingabe korrekt auszuführen. Informationen sind immer soziale Interpretationen von Daten, sie haben eine Bedeutung, sie haben Sinn. Wobei aus Informationen Wissen wird, wenn sie in einen bestimmten Erfahrungskontext eingebunden sind (vgl. Willke 1998: 162). Das ist etwas, dass ein Computer nicht kann, weil interpretieren nur Menschen können. Wie von Brödner (2014) analysiert, ist der Interpretationsmoment hervorzuheben, der zwischen maschinell ausgeführten Operationen und sozialen Handlungen eingebettet ist, zwischen zu interpretierenden Daten und maschinellen Verarbeitungen (Algorithmen). Wenn Maschinen Daten liefern und diese dann zur Steuerung und Kontrolle dieser Maschinen dienen, dann müssen diese Daten von den Beschäftigten interpretiert werden (wie bspw. von Zuboff 1988 ausführlich beschrieben).

Der Begriff des Wissens markiert den Übergang zum Handeln. Mit Wissen können Menschen Probleme lösen. Mit Wissen können sie in einem bestimmten Kontext handeln und entscheiden. Somit ist es irreführend, davon zu reden, dass Computer handeln, entscheiden oder etwas wissen. Das können nur Menschen, die einem Zeichen eine Bedeutung beimessen können (s.o.) und dann der Bedeutung gemäß handeln.

»Wissen als Erklärungszusammenhang für Informationen, als eine mit Erfahrung, Kontext, Interpretation und Reflexion angereicherte Form der Information, geeignet, Arbeitshandeln und Entscheidungen anzuleiten« (Jürgens 1999 nach Wilkesmann 2005: 56).

Also: Wenn ein Mensch Daten Sinn geben kann, sind es Informationen. Erst wenn dieser Mensch daraus Handlungen und Entscheidungen ableiten kann, wird es zu Wissen. Genau dieser Prozess, den eine Autorin als De- und Rekontextualisierung beschreibt (vgl. Degele 2000: 69), passiert täglich in den softwaregestützten Organisationen. So z. B. wenn jemand vor einer Eingabemaske steht, die er nicht versteht, weil ihm das Wissen fehlt, oder wenn der Rechnungsbeleg alle notwendigen Informationen enthält, die/der neue Sachbearbeitende aber nicht genau weiß, warum die Rechnung nun so und nicht anders aufgebaut ist. Für die Organisation ist es wichtig, dass die Software das richtige Rechnungsformular erzeugt. Womöglich ist die Umsetzung auch dokumentiert. Für neue Sachbearbeitende wäre es jetzt wichtig zu wissen, wo sie diese Dokumente finden oder wer ihnen sagen kann, warum etwas wie auf dem Rechnungsformular gestaltet ist. Selbst bei programmiertem Quellcode ist es wichtig zu wissen, warum etwas wie entwickelt wurde. Die Bedeutung ist sonst nicht ersichtlich. Selbst eine umfangreiche Dokumentation reicht oft nicht aus, um das gesamte Wissen zu hinterlegen (vgl. D'Adderio

2003: 326). Der Quellcode kann der Maschine eindeutige Befehle geben, liefert aber nicht automatisch seine Entstehungsgeschichte und seinen Sinn für den Kontext, für den er existiert.

Um Daten, Informationen und Wissen unter Menschen auszutauschen, ist Kommunikation notwendig. Dadurch, dass der Computer Kommunikation von ihrem Kontext entkoppeln kann und wie ein Buch, ein Brief oder ein anderes Schriftstück vom Sendenden abstrahiert, verändert sich das Verhältnis von Information, Mitteilung und Verständnis durch Arbeiten via Software (vgl. Degele 2000: 65). Der Sinn der Kommunikation ergibt sich nicht mehr direkt aus der Mitteilung, sondern der/die Empfänger:in kann unabhängig davon interpretieren und der Mitteilung einen Sinn geben (vgl. Esposito 1993: 351f.). Sie/er kann aber auch daran scheitern, weil er/sie z.B. einen Begriff nicht versteht. Das alles macht menschliche Kommunikation zum wesentlichen Bestandteil der Softwaregestaltung, die der Computer nicht vollständig ersetzen kann. Das zeigt sich, wie im Folgenden dargestellt, vor allem im Anforderungsmanagement. Letztlich sorgen die Beschäftigten in einem stetigen Kreislauf aus Daten, Information, Wissen und Kommunikation dafür, dass Organisationen Software anwenden, programmieren und gestalten.

Zu 2.: Neben der interpretierenden Funktion des Menschen sind die von ihm verwendeten Zeichen Teil einer sozialen Welt. Der vorliegenden Arbeit liegt eine klare Unterscheidung zwischen Sozialem und Technischem zugrunde. Sie folgte dabei ausgehend von C. S. Peirce und Jürgen Habermas den Autor:innen Mingers und Willcocks (2014), die von drei Welten ausgehen:

A) Der Welt der Person, welche Zeichen und Nachrichten erzeugt und interpretiert (Softwaregestaltende, -anwendende, -programmierende).
B) Der materiellen Welt, in der die Zeichen verkörpert sind und übertragen werden (Software, Hardware).
C) Der sozialen Welt, weil die individuelle Nutzung des Zeichens nicht über das Soziale hinausgehen kann (z.B. der kollektive Arbeitsprozess der Softwaregestaltung, die Arbeitsteilung zwischen Anwendung und Entwicklung).

Für Mingers/Willcocks sind die oben aufgeführten drei Welten ontologisch und epistemologisch getrennt. Wobei für sie das Individuum im Mittelpunkt steht: »communications and information systems rest on individuals who create and send, or have sent, messages and data; then receive and interpret them; then act (or not act) upon them« (Mingers/Willcocks 2014: 50). Das Subjekt vermittelt zwischen materieller bzw. technischer und sozialer Welt, indem es Zeichen deutet. Damit grenzen sie sich von Ansätzen wie jenem der *Sociomateriality* ab, für den Soziales und Technisches nicht trennbar sind. Einer dieser Ansätze ist die Actor-Network-Theorie: Diese vernachlässigt für Mingers/Willcocks sowohl die vermittelnde Funktion des Einzelnen als auch die ontologischen Unterschiede zwischen Technik (Software) und der sozialen Welt (bspw. einer Organisation). Wie sehr die oder der Einzelne als Teil einer sozialen Welt bei der Softwaregestaltung agiert, führt 5.2 weiter aus und ist zentraler Bestandteil dieser Untersuchung (vor allem beim Arbeitsprozess der Softwaregestaltung selbst).

Zu 3.: Wenn eine Verwaltung bestehende Formulare in Software übersetzt und nun papierlos arbeitet, ohne dass sich etwas am bürokratischen Ablauf oder den Formularen wesentlich verändert: Entstehen hier neue Informationen oder wird hier nur etwas in Software überführt? Informatisierung, wie den Begriff unterschiedliche Autor:innen verwenden (vgl. Baukrowitz et al. 2006, Boes et al. 2018, Ziegler 2020), unterscheidet die vorliegende Arbeit von jener Tätigkeit, bei der Menschen die reale, analoge Welt in digitale umwandeln: der Softwareentwicklung.

5.1.2. Konkret und abstrakt: mehrere Schichten, sprachliche Strukturierung

Das Besondere an der Softwaregestaltung ist, dass die Beschäftigten während des Arbeitsprozesses mit den verschiedenen **technischen Schichten und sprachlichen Strukturierungen** der Software arbeiten müssen. Daraus erklärt sich auch die große Bedeutung von Wissen und Kommunikation, weil sich die Beschäftigten über diese unterschiedlichen Schichten und Begriffe verständigen müssen. Im Unterschied zu anderen Technologien besteht Software komplett aus Zeichen. Mit den zugrundeliegenden 0en und 1en beschäftigt sich in den EVU niemand. An ihren unterschiedlichen Erscheinungsformen kommt aber keiner mehr vorbei. Aus Arbeitssicht sind vier Aspekte zentral: Die Programmierung von (1.) Algorithmen verlangt je nach (2.) Programmiersprache unterschiedliche Fertigkeiten. Dazu gehört (3.) Softwarearbeit mit dem Medium der Sprache und Begriffen wie Architektur, Modelle oder Schnittstellen zu strukturieren. (4.) Es existiert eine Oberfläche als Medium zwischen Anwendenden, Daten und Algorithmen.

Zu 1.: Da sind zum einen die in der Software eingeschriebenen Anleitungen zur Datenverarbeitung: die Algorithmen. Sie stellen klare Vorschriften dar. Jeden formalisierbaren Sachverhalt kann die symbolische Maschine Computer ausführen. Dabei gibt es keinen Interpretationsspielraum und die Algorithmen sind durch ihre Schriftlichkeit klar definiert. Sie sind eindeutig, determiniert, unterscheidbar und allgemein (vgl. Degele 2000: 62f.). In einem Programm können Tausende solcher Vorschriften enthalten sein. Es ist dann eine Frage des Fokus, ob man eine relevante Funktionsweise (bspw. den Suchalgorithmus von Google) oder die Struktur einer Software (Methoden, Klassen, Funktionen, Befehle etc.) zugrunde legt, wenn man von Algorithmus spricht.

Zu 2.: Gebaut werden diese Vorschriften mithilfe von Programmiersprachen. Der Begriff »Sprache« sollte nicht in die Irre führen. Sie werden nicht wie menschliche Sprachen verwendet. Sie wurden als Medien entwickelt, um es den Menschen einfacher zu machen, der Maschine Befehle zu geben (anderes als bei der menschlichen Sprache gibt es keine Ambivalenz, Ironie oder Ambiguität). Softwarespezifische Sprachen wie ABAP (für SAP), funktionsspezifische wie R, objektorientierte wie C++ oder *Low-Code*-Ansätze zeigen, dass es genau darum geht: ABAP soll möglichst auch für Nicht-Programmierende leicht erlernbar sein. Einfache Abfragen und Ausgaben von Datenbanktabellen sollen bspw. auch für die in den Wirtschaftsorganisationen weitverbreiteten Betriebswirtschaftlern möglich sein. R wird für statistische Aufgaben verwendet und verfügt über die entsprechenden Befehle. *Low-Code*-Software (im Sinne von wenig programmieren) anbietende Unternehmen versprechen, dass jedes Mitglied einer Organisation eine Software entwickeln kann, weil keine komplizierte Programmiersprache gelernt wer-

den muss (bspw. FrontPage von Microsoft, um Webseiten zu erstellen). Objektorientierte Sprachen wie C++ ermöglichen es Programmierenden, Bibliotheken mit Quellcodes aufzubauen, die sie wie Bausteine in unterschiedlichen Kontexten verwenden können. Das erleichtert den für den Menschen sinnhaften Aufbau von Software und die sinnhafte Aufteilung der einzelnen Bestandteile. Es gibt immer noch sogenannten »Spaghetti«-Code, bei dem Befehl an Befehl aneinandergereiht ist, bis mehrere Tausend Zeilen dastehen, die schwer wartbar sind. Der Maschine ist das egal, für den Menschen eine Qual.

Zu 3.: Hier wurden schon einige analytische Sprünge gemacht, die für Software typisch sind: von Programmiersprachen über deren Eigenarten und deren Folgen für größere Mengen an Quellcode und Methoden, diesen Quellcode zu organisieren (bspw. in Klassen, Funktionen etc.). Wie im weiteren Verlauf mehr und mehr klar wird, gibt es eine Vielzahl von Konzepten, Begriffen und Methoden, um die Arbeit mit und am Quellcode, aber auch den Quellcode selbst zu organisieren. Die Vielfalt an Entwicklungsmöglichkeiten kontrollieren Modelle, damit die Programmierung nicht im Chaos endet. Bestimmte Formen der Programmierung, die den Quellcode strukturieren, stellen bereits eine Form der Modellierung dar[1]. Sie machen Modellierung alltäglich (vgl. Mahr 2009: 230f.). Modelle sind Ressourcen zum Speichern und Transportieren und sie sind Agenten »zur Konstruktion und Gestaltung neuer Realitäten« (ebd.). Sie spielen eine wichtige Rolle bei Erkenntnis- und Meinungsbildungsprozessen. Unterkategorien von Modellen sind bspw. Architekturen, Prinzipien der Systemgestaltung, Techniken der Abstraktion oder Prinzipien der Usability (vgl. ebd. 248). Vor allem der Modellbegriff der Architektur[2] ist mittlerweile weitverbreitet. Es gibt viele Definitionen von Architektur und laut einigen Autoren ist eine richtige Definition auch nicht möglich (vgl. Vogel et al. 2009: 49). Sie schlagen trotzdem eine vor:

> »Die Software-Architektur eines Systems beschreibt dessen Software-Struktur respektive dessen -Strukturen, dessen Software-Bausteine sowie deren sichtbaren Eigenschaften und Beziehungen zueinander und zu ihrer Umwelt« (ebd.: 49).

Für sie geht es darum, dass Software-Architektur »Komplexität überschaubar und handhabbar […] [macht] in dem sie nur wesentliche Aspekte eines Systems zeigt« (ebd. 10). Es geht um die Fundamente und tragenden Säulen einer Software (vgl. ebd.). Ob eine Firma intern etwas programmiert, es externen Programmierenden überlässt oder Bausteine aus der Cloud verwendet: Das wird schnell zu einer Frage der Architektur, weswegen auch Nicht-ITler außerhalb von IT-Abteilungen und -Unternehmen über sie sprechen. Weitere mittlerweile geläufige Begriffe wie Schnittstellen[3] oder Softwarepakete zeigen, wie strukturierungsbedürftig die Sprache bei der Arbeit mit Software ist.

---

1 Zum Beispiel die objektorientierte Programmierung.

2 Viel wichtiger als ihre Rolle bei der Modellierung ist die Softwarearchitektur bei der Untersuchung der Fallstudien im Empirie-Teil, weil sie die Organisation der Softwaregestaltung prägt. Darauf geht das nächste Kapitel gesondert ein und zeigt, welche konkreten Eigenschaften der Softwarearchitektur für die Analyse der Formen und Folgen der Softwaregestaltung relevant sind (6.4.2.2).

3 Meist nur noch APIs (Application Programming Interface) genannt.

Zu 4.: Neben den Medien zum Programmieren der Maschine gibt es noch die Medien zur Ein- und Ausgabe von Daten und Algorithmen: ob Web-Oberflächen, Eingabemasken, Computerspiele oder Textverarbeitungsprogramme. Anders als mechanische Maschinen wie Verbrennungsmotoren, Leichtbauroboter o. ä. erfordern sie per se keine mechanische Reaktion. Natürlich kann ein Programm so programmiert sein, dass eine Eingabe erforderlich ist oder nur eine bestimmte Zeit zur Eingabe bleibt. Software kann aber auch einfach nur Daten darstellen. Sie kann Arbeitsabläufe als Fließband darstellen – muss sie aber nicht. So oder so bleibt Software ein Medium zur Darstellung oder Eingabe von Daten. Der Mediencharakter zeigt sich bei Webseiten wie Wikipedia oder einer digitalen Zeitung. Der Inhalt ist zwar der gleiche (die Daten), aber die Aufbereitung anders, was Folgen für das Leseverhalten oder die Verbreitungsmöglichkeiten hat.

Wie die Fallstudien zeigen werden, spiegeln sich die verschiedenen technischen Schichten und sprachlichen Strukturierungen in der Arbeitsteilung zwischen Programmierenden, IT-Projektleitenden, IT-Beratenden, Key User:innen etc. wieder. In ihrer Arbeit vermitteln sie zwischen verschiedenen technischen Schichten, Perspektiven und Begriffen, wobei jeder seine Schwerpunkte hat und sie letztendlich eine gemeinsame Sprache finden müssen. Es ist Kommunikation notwendig, um die jeweiligen Perspektiven auf die Software zu integrieren und sich zu verständigen. Damit geht es bei der softwaretechnischen Interdisziplinarität nicht nur um das jeweilige industriespezifische und softwaretechnische Domänen-Wissen.

5.1.3. Zwischen Text und Blackbox: Grenzen der Gestaltung und des Verstehens

Für die Softwaregestaltung spielt es eine besondere Rolle, dass unterschiedliche Personen und Organisationen unterschiedlichen Zugriff und Gestaltungsmöglichkeiten bezüglich der Software haben und sich die Software stetig ändert. Nicht jede:r kann den Quellcode oder eine Datenbank einsehen oder verstehen und verändern. Im Verlauf der Entwicklung einer Software verändert sich, was die Beschäftigten noch gestalten oder worüber sie noch reden können (vor allem bei Standardsoftware). Das ist insofern wichtig, weil es Teil des Arbeitsprozesses der Softwaregestaltung ist, zu vermitteln: zwischen Teilen der Software, die als Blackbox erscheinen, und den analysierbaren; zwischen gestaltbaren und nicht mehr gestaltbaren Teilen der Software; zwischen Softwareoberflächen und einem Quellcode, die oder den man kennt oder einem fremd ist; zwischen einer Software und ihrem Umfeld, die sich beide stetig ändern und damit das Wissen über beide langfristig nicht gesichert ist.

Anders als bei anderen Maschinen oder Werkzeugen gibt es die Möglichkeit, den Zugriff auf Software genau festzulegen. Dies geschieht häufig durch differenzierte Berechtigungsstrukturen, die unterschiedliche Zugriffe auf Software und damit Daten, Funktionalitäten bis hin zum Quellcode erlauben (bspw. bei SAP, Windows oder diversen Online-Plattformen). Mitarbeitende in einem Call-Center müssen mit der Software arbeiten, die ihnen ihre Firma zur Verfügung stellt. Wenn die Software genaue Vorgaben macht, wie ein Anruf abzuwickeln ist, und bestimmte Daten anzeigt, können die Mitarbeitenden das nicht ändern. Es können auch einzelne Eingabefelder für Mitarbeitende freigeschaltet oder gesperrt sein. Andererseits gibt es formalisierte Wege für den Zugriff auf die Gestaltung von Software. Viele Firmen haben (formale) Wege, um an

bestimmte Berechtigungen zu kommen. Bei Schwierigkeiten oder Fehlern mit Software gibt es einen First-, Second-, Third-Level Support, der Anwendenden weiterhilft.

Für die unterschiedlichen Stakeholder einer Software gibt es unterschiedliche Möglichkeiten der Gestaltung und des Verstehens. So entscheiden meist wenige über die Softwarearchitektur, die langfristig weitreichende Folgen hat. Manche Softwarelösungen bieten Einstellungs- oder Anpassungsmöglichkeiten an, die vom Festlegen des Farbschemas bis hin zum Ersetzen einzelner Codestellen durch eigenen Quellcode reichen können. Neben der Architektur können die Anwendenden oftmals nichts mehr daran ändern, wie die Programmierenden den Anwendungskontext modelliert haben, auch wenn das ihre Arbeit beeinflusst (vgl. Mahr 2009: 230). In Software ist ein »objektiviertes Modell der organisatorischen Wirklichkeit« (Heidenreich/Kirch/Mattes 2008: 4) fixiert. Zudem ist das meiste Wissen, was in der Software steckt, nicht mehr außerhalb vorhanden oder kann nur durch Fachexpertise oder über öffentlich zugängliche Spezifikationen mühsam angeeignet werden. Der Computer ist für die meisten eine Blackbox (vgl. Zuboff 1988: 166). Das kann bedeuten, dass die Software Dinge tut, von denen die Anwendenden nichts wissen – wie z.B. unentdeckt überwacht zu werden, wie dies durch Software von Google oder Amazon passiert (vgl. Zuboff 2018).

> »Je umfassender und komplexer Maschinen werden, wandern Praktiken und Normen in die materielle Basis der Gesellschaft, allerdings black-boxed« (Joerges et al. 1998: 372).

Trotz beschränktem Zugriff auf eine Software und obwohl sie eine Blackbox sein kann, die nicht mehr änderbar ist, ist die oder der einzelne Beschäftigte für ihre/seine Arbeit auf das Wissen über die Software angewiesen. Das Wissen über den Anwendungskontext allein reicht nicht. Denn der Anwendungskontext existiert nur noch als einer, den die Software bereits verändert hat. Eine Autorin spricht von einem reflexiven Strukturierungsprozess: Beim Einsatz von Technik in organisationalen Netzwerken (in dem konkreten Fall geht es um Call-Center) bedeutet dies, dass sich das Verhältnis von organisationalem Netzwerk und Technikverwendung als eines der zunehmenden Durchdringung und wechselseitigen Gestaltung beschreiben lässt. Soziales und Technisches sind nur noch schwer zu trennen (vgl. Longen 2015: 120). In einer Studie zu einer ERP-Einführung ist von »durchwurstelt«, dem Eigenleben des Einführungsprojektes oder einer »unruly technology« die Rede: Es kann immer etwas Unvorhergesehenes passieren (vgl. Conrad 2017: 189f.). Das liegt für die Autorin daran, dass Organisation und Technik sich nicht mehr auseinanderhalten lassen.

> »Man hat es nicht mit zwei unterschiedlichen Entitäten zu tun – Organisation auf der einen Seite, Medien und Technologien auf der anderen –, sondern beide enthalten Elemente voneinander und haben sich in Abhängigkeit voneinander und in Abstimmung aufeinander ausgebildet.« (Conrad 2017: 12)

Die Beschäftigten denken immer nur noch im Angesicht der Software über ihre eigene Arbeit und Organisation nach. Über die Zeit (das können Jahrzehnte sein) findet eine Ko-Konstruktion von Organisation und Software durch die anwendenden Beschäftigten

und Softwaregestaltenden statt. Am Ende eines IT-Projektes existiert der Arbeitskontext nicht mehr, für den ursprünglich der Auftrag erteilt wurde, eine Software zu entwickeln:

> »Coevolution changes the context […] and building the system changes the context itself, a software development projects actively obsolesces its own contract« (Ralph 2015: 38).

Das Wissen über Anwendungskontext und Software ist nicht nur verschränkt. Es ändert sich auch stetig. Die softwareeinsetzende Organisation als solche hat mit einem permanenten Anpassungsbedarf zu rechnen. Bereits in den 80er Jahren schreibt Lehman (1980), dass sich Software permanent ändert. Seine ersten zwei Gesetze der Programmevolution beziehen sich darauf: 1. kontinuierlicher Wandel und 2. zunehmende Komplexität der Software. Mit dem Einsatz einer Software wird der oder die Anwendende Teil ihres Lebenszyklus. Dabei geht es nicht nur um einen allgemeinen Zyklus der Softwareevolution: initiale Software, Entwicklung, Betreuung, Ausphasung, Abschaltung (vgl. Masak 2006: 222). Wenn SAP auf die Cloud und die neue Version seiner ERP-Software S/4 umstellt und die Wartung für die alte Version R/3 ausläuft, entsteht der Zwang, die Software auszutauschen. Dabei gilt besonders bei individuell entwickelter Software: Wenn Anwendende, Gestaltende oder Programmierende neu in einen Anwendungskontext kommen, kennen sie die Vorgeschichte der nur für eine Organisation entwickelten Software nicht[4].

Letztendlich sind Anwendende, Gestaltende und Programmierende nicht nur Teil einer modellierten Welt. Sie werden auch Teil eines Produktzyklus, auf den sie wenig Einfluss haben – und damit wenig Einfluss auf einen Teil des Wissens, den sie für ihre alltägliche Arbeit brauchen und das sich stetig ändert.

5.2. Softwareentwicklung: vom einsamen Nerd zum kollektiven Kommunikationsprozess

Im Laufe der Zeit wurde Softwareentwicklung immer weniger zu einem rein technischen Problem, das Techniker:innen lösen. Wie bereits oben erwähnt, wurde es zu einer großen Herausforderung, die für den fremden Anwendungskontext nützliche Software zu programmieren. Dafür sind Methoden wie Scrum nützlich (Näheres weiter unten unter 5.2.4), die den kontinuierlichen, geregelten sozialen Austausch mit klaren Rollen in den Mittelpunkt der Softwareentwicklung stellen. Trotzdem konnte in der Forschung kein Konsens hinsichtlich einer Best Practice gefunden werden, die als Orientierung für die Kontrolle von Softwaregestaltung in unterschiedlichen Kontexten der Energiewirtschaft nützlich sein könnte. Vielmehr scheinen unterschiedliche Methoden Softwaregestaltung

4 Man spricht auch von Legacy einer Software (vgl. dazu Fischbach 2016: 395ff.). Manche individuell entwickelten Altsysteme von Firmen sind kompliziert, nicht wartungsfreundlich programmiert und man befürchtet unvorhersehbare Fehler bei Änderungen an ihnen. Verlassen Mitarbeitende das Unternehmen, die mit dem Altsystem gut vertraut waren (z.B. weil sie es selbst entwickelt haben), kann das der Anlass sein, stattdessen eine Standardlösung einzuführen.

zu ermöglichen, solange sie die zentrale Rolle von Wissen und Kommunikation für den Arbeitsprozess berücksichtigen.

5.2.1. Vom schnellen Reparieren zum iterativen, kollektiven Kommunikationsprozess

Einerseits würde man aus dem bisher Gesagten vermuten, dass Kommunikation wichtig in der Softwareentwicklung ist. Andererseits ist nicht verwunderlich, dass es bei einer neuen Technologie, die aus der Ingenieurs- und Mathematik-Tradition kommt, erst einmal um deren Erforschung und Entwicklung ging. Dafür waren komplexe Kommunikationsprozesse und kommunikative Fertigkeiten nicht entscheidend. Für die Programmierer hieß es in den 50ern noch »Engineer software like you engineer hardware.« (Boehm 2006: 13) Oftmals fand IT-Arbeit damals noch in Forschungs- und Entwicklungsabteilungen statt, wo die Techniker:innen unter sich waren. Unter seines/ihresgleichen sind die Wissensgrenzen geringer. Als einen extremen Typus sieht Weizenbaum die zwanghaft Programmierenden an, für die Programmieren ein Selbstzweck ist. Ihnen geht es vor allem darum, mit der Maschine zu interagieren (vgl. Weizenbaum 1978: 161). Sollen sie dann Software schreiben, die in anderen Kontexten als der Werkstatt oder dem Labor funktionieren soll, ändern sich die Anforderungen. Die Erfahrung der Beherrschbarkeit der Maschine durch Erteilen eindeutiger Befehle via Programmiersprache wird unreflektiert auf soziale Zusammenhänge übertragen, in der diese Software entsteht oder in der sie wirken soll (vgl. Klischewski 1996: 78). Die Widerständigkeit des Sozialen fand erst über die Jahrzehnte hinweg in den Methoden der Softwareentwicklung mehr und mehr Berücksichtigung.

Hieß es in den 60ern »code-and-fix«, also einfach zu programmieren, schauen, ob es funktioniert, und dann zu verbessern (vgl. Boehm 2006: 14), wurden in den 70ern die getrennten Aufgabenschritte der Anforderungsanalyse und des Designs eingeführt (siehe auch Friedman/Cornford 1989). Das ursprünglich entwickelte Wasserfallmodell sah die erst mit Scrum weitverbreiteten Mechanismen der Iterationen, Prototypen und Feedbacks zwischen den Entwicklungsschritten vor. In der Praxis wurde das Wasserfallmodell aber erst einmal als rein sequenzieller Prozess ausgelegt (vgl. Boehm 2006: 15). Softwareentwicklung wurde zum Arbeitsvorgang, in dem streng abgetrennte Phasen der Spezifikation, Programmierung, Tests und Implementierung aufeinander folgen. Kritisch wird diese strikte Trennung vor allem, weil bei komplexen Anforderungen fehlerfreies Arbeiten unmöglich ist. Eine vollständige Konzeption oder Spezifikation ist nicht möglich, weil sich u.a. die Anforderungen der Anwendenden im Projektverlauf ändern. Das kann an einem veränderten Umfeld liegen (Konkurrenzdruck, Markt verändert sich) oder daran, dass technische Möglichkeiten erst bewusst werden, dass es Kommunikationsprobleme gab oder dass erst in der Anwendung neue Ideen auftauchen (vgl. Funken 2001: 30). In einem bekannten Artikel von 1980 schreibt Lehman, dass ein Programm nie korrekt sein kann, weil es die Umwelt nicht komplett beschreiben kann. Software ist immer nur ein Modell der Welt. Für ihn kann es bei Software deshalb nicht um absolute Korrektheit gehen (was eine mathematische Herangehensweise bedeuten würde), sondern um die Relevanz der Ergebnisse oder die Anwendungsfreundlichkeit (vgl. Lehman 1980: 1064). Er führt auch eine Unterscheidung verschiedener Programmtypen ein. Das

ist insofern wichtig, weil es darauf hinweist, dass es jenseits der hier behandelten industriespezifischen Softwareentwicklung selbstverständlich weiterhin Programme gibt, die einige wenige oder sogar eine programmierende Person allein nach einer klaren Spezifikation entwickeln kann (bspw. ein:e Physiker:in, die eine Software für ein physikalisches Experiment programmiert).

Laut Lehman kann Programmierung zudem keine Fließbandarbeit sein, weil u.a. die Entwicklung nicht bereits im Vorhinein in einfach verbundene Untereinheiten zerteilt werden kann, ohne dass sie sich gegenseitig bei der Umsetzung beeinflussen (vgl. Lehman 1980: 1065). Das liegt auch an der Abstimmung zwischen der fachlichen Domäne, in der die Software laufen soll, und den Programmierenden. Um die Nutzendenpartizipation und damit die Kommunikation zur Programmierung zu verbessern, wurde seit Mitte der 70er die Methode des Prototyping entwickelt. Sie entlastet die Anforderungsaufnahme, weil das anschauliche Ergebnis als Kommunikationsgrundlage fungiert und Nutzende direkt an der Spezifikation beteiligt sind (vgl. Funken 2001: 32ff.). Wie weitgehend sich das in der Praxis mit der Zeit durchgesetzt hat, wäre zu untersuchen.

In den 80ern stellten Floyd/Keil eine Methode vor, die eine iterativ-inkrementelle Vorgehensweise und eine kontinuierliche Kommunikation zwischen programmierenden und anwendenden Beschäftigten vorsieht. Vorteilhaft ist dabei auch die geteilte Verantwortung für die Weiterentwicklung – anstatt dass sie nur bei den Programmierenden liegt, die gar nicht wissen, was die Anwendenden brauchen (vgl. Funken 2001: 36). Die/der Programmierer:in soll nicht mehr einfach Herstellende:r sein, sondern

> »Berater[:in] in Informationsangelegenheiten, welche Multiperspektivität anerkennt und umsetzt, Vielfalt und Rückkopplung sucht und zu Revisionen bereit ist« (Floyd/Keil 1983: 36 zitiert nach ebd. 37).

Das Agile Manifesto von 2001 (Scrum ist eine der agilen Methoden) führte diesen Ansatz weiter und stellte vier Kernforderungen auf:

- »Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation
- Responding to change over following a plan.« (Beck et al. 2001)

Letztendlich legt das Manifest einen klaren Fokus auf einen iterativen Arbeitsprozess mit direktem, regelmäßigem Feedback. Das Wasserfallmodell setzte sich auch deswegen nicht vollumfänglich durch, weil der Druck wuchs, Software möglichst schnell auf den Markt zu bringen, und es immer mehr Software gab, bei der die Benutzendeninteraktion im Vordergrund stand. Anforderungen waren schwerer im Vorhinein festzustellen. Sie wurden emergent und folgten dem IKIWISI-Syndrom – I know it when I see it: Die anwendende Person konnte erst sagen, ob die Software den Anforderungen genügt, wenn sie das Programm selbst gesehen hat und testen konnte (vgl. Boehm 2006: 18). Deshalb war schon die Verwendung von Prototypen ein Fortschritt. Die agilen Methoden bewerkstelligten das, indem in kurzen Zyklen (bspw. monatlich) ausführbare Software erstellt

wird, die dann die zukünftigen Nutzenden oder die Fachleute testen und dazu Feedback geben können.

5.2.2. Kommunikationskompetenz und -kern: Anforderungsmanagement

Ohne auf alle agilen Ansätze[5], geschweige denn alle anderen Methoden eingehen zu können, soll der obige kurze Abriss andeuten, dass die Softwareentwicklung sich selbst erst mit der Zeit methodisch mit ihrer sozialen Einbettung befasst hat. Umfangreiche Feldstudien in den 80ern haben festgestellt, dass nicht fehlende technische Fertigkeiten das Problem waren und sind. Die Softwareproduktivität und -qualität beeinflusst vor allem

> »zu geringe und zu wenig verbreitete Kenntnisse der Entwickler über das Anwendungsgebiet, sich verändernde und widersprüchliche Anforderungen an das Software-Design und Kommunikations- und Kooperationsprobleme zwischen Entwickler und Kunden« (Funken 2001: 46).

Es wurde abgerückt davon, sich allein auf technische Fertigkeiten zu konzentrieren:

> »Software-Entwicklung und -gestaltung muß also […] in wesentlichen Teilen als ein Lern-, Kommunikations- und Aushandlungsprozeß verstanden werden, der hohe Kooperations- und Kommunikationsanforderungen – mithin soziale Kompetenzen – an die Entwickler stellt.« (Funken 2001: 48)

Mehrere Autor:innen weisen in den 80ern und 90ern drauf hin, dass das auch in der Ausbildung von Informatiker:innen berücksichtigt werden sollte (vgl. Funken 2001: 46ff., Baukrowitz/Boes/Eckhardt 1994).

> »[D]rei Viertel ihrer Arbeitszeit benötigen Software-Entwickler für die Kommunikation mit verschiedenen Partnern: Auftraggebern, Benutzern, Kollegen, Management, Vertrieb usw.« (Funken 2001: 48)

Christiane Floyd sprach 1992 von »software development as an insight-building process in terms of multiperspectivity, self-organization and dialogue« (Floyd 1992: 86) und eben nicht davon, dass Anforderungen fix auszumachen sind wie technische Eigenschaften einer Maschine oder in einem kontrollierbaren, experimentellen Setting. Anders als bspw. bei einem Labor-Experiment ist der Entwicklungsprozess nicht durch innertechnische Rationalität vorgegeben, sondern ist ein Gestaltungsprozess, bei dem nicht nur ein technisches System, sondern auch »die sozialen Zusammenhänge seiner Verwendung modelliert werden müssen« (Schulz-Schaeffer 1996: 8).

Der kommunikationsintensivste Teil der Softwareentwicklung, das Requirements Engineering (auf Deutsch meist: Anforderungsmanagement), entwickelt sich seit den 70ern zu einem eigenständigen Forschungsfeld (vgl. Funken 2001: 52). Es stellt die korrekte und objektive Darstellung von Anforderungen in Frage. Es plädiert dafür, unterschiedliche Meinungen, Perspektiven und Sichten zu berücksichtigen. Unter anderem

5 Wie Extreme Programming, Kanban, Scrum etc.

sollen auch potenziell konfliktträchtige Perspektiven aufgenommen werden (vgl. Funken 2001: 56f.). Das Schreiben von Anforderungen, die dann die Programmierenden umsetzen, hat etwas von einer Sozialforschung: Wer mit wem wie interagiert und wie die Arbeitsabläufe sind, muss erfragt und beobachtet werden. Die Anforderungsaufnahme kann Methoden wie Interviews, Ethnografie, Perspektivenübernahme oder diskursive Anforderungsanalyse verwenden. Die Anforderungsstrukturierung nutzt Skizzen, Use Cases, Diagramme etc. (vgl. Kaminski 2012: 112ff.). Im Prozess der Anforderungsaufnahme treten IT-Fachkräfte als Kommunikations- und Übersetzungsexpert:innen auf, wobei die Programmiersprache einen Eindeutigkeitsdruck auf die Kommunikation des Anforderungsmanagements ausübt (vgl. Kaminski 2012: 89). Es muss Übersetzungsarbeit auf dem Weg zum Quellcode geleistet werden, weil Anwendende, Auftraggebende und Programmierende unterschiedliche Sprachen sprechen (vgl. Kaminski 2012: 91). So entscheidend ist die Sprache dabei, dass selbst sprachliches Framing relevant ist, um zu verstehen, wie Entwicklungsprozesse ablaufen und Expert:innen Autorität gewinnen (vgl. Alvarez 2002: 103).

Die Kommunikation muss es den Systemfachleuten ermöglichen, sich mit dem fachlichen Kontext vertraut zu machen, und den fachlichen Kontextexpert:innen, sich mit der Systemsprache vertraut zu machen. Nur so kann der Formalisierungs- und Systembildungsprozess funktionieren (vgl. Kaminski 2012: 121). Anforderungen aufzunehmen ist für einige Forschende vor allem ein Meinungsbildungs- und Verbalisierungsprozess:

> »Based on this vision, much of what occurs during the requirements process should be about opinion and will formation that is, the development of an understanding of, and the creation of meaning – about the organization and its goals and processes for achieving these goals, supported by new systems« (Ross/Chiasson 2011: 134).
> »[R]equirements elicitation takes on the form of a ›confessional‹ act where the individual verbalizes thoughts, intentions and consciousness« (Alvarez 2002: 85).
> »The RE process is a socio-technical activity. It requires intensive communication among stakeholders who have different backgrounds, skills, culture, knowledge, and behavior« (Alsanoosy et al. 2020: 356).

Erfolgreicher Wissenstransfer, gegenseitiges Verständnis (gemeinsame Konventionen und Sprache) und Kommunikation sind wesentliche Faktoren für eine erfolgreiche Softwareentwicklung (vgl. Corvera Charaf et al. 2013: 117).

Das gilt ebenso bei der Implementierung einer Standardsoftware. Es geht darum, inwiefern diese anzupassen oder wie sie einzustellen ist. Auch hier müssen die Anforderungen der Kundschaft erst aufgedeckt werden, weil sie für Beratende und Kundschaft nicht so klar auf der Hand liegen (vgl. Mormann 2016: 169). Dabei haben es die Beratenden in der Hand, welche Möglichkeiten der Software sie preisgeben oder bspw. aus Kostengründen die Gestaltungsmöglichkeiten einschränken (vgl. Mormann 2016: 186).

Wie bereits oben aufgezeigt, sind Begriffe wie Funktionen, Architekturen, Modelle oder Schnittstellen Hilfsmittel, um über Software zu reden. Dabei können im Prozess des Anforderungsmanagements nicht nur einzelne Funktionalitäten eine Rolle spielen, sondern auch wie die Software aufgebaut ist. Modelle dienen dazu, um über Software zu diskutieren und sie zu dokumentieren. Sie spielen in unterschiedlichen Entwicklungs-

methoden jedoch eine unterschiedliche Rolle. Manche Methoden[6] betrachten Modelle von vornherein als vorläufig und als fortlaufend anzupassen (vgl. Mahr 2009: 245). Agile Softwareentwicklung verwirft den »Gebrauch von Modellen zugunsten unmittelbarer Programmierung« (Mahr 2009: 246). Folglich wird das Programm selbst zur Referenz, um über die gedachten Modelle zu reden und sie anzupassen. Abhängig von der Methode unterscheidet sich dann die Kommunikation im Anforderungsmanagement und damit der Softwaregestaltung.

5.2.3. Kommunikation und Wissen organisieren: Local Practice statt Best Practice

Um die Softwareentwicklung so zu organisieren, damit sie »the right thing« (Friedman/Cornford 1989: 204) tut, hat die Prüfung der Forschungsliteratur keine Best Practice zutage gefördert. Vielmehr existieren lokale Praktiken und widersprüchliche Vorgehensweisen. Daraus ergeben sich Ansätze, aber noch keine Konzepte für die Beschreibung dessen, was bei industriespezifischer Softwareentwicklung in der Phase der Softwaregestaltung in unterschiedlichen Kontexten zwischen Anwendung und Programmierung passiert.

Unabhängig von einzelnen Methoden wie Scrum oder dem Anforderungsmanagement betrachten die Autor:innen der »general theory of software engineering« (Wohlin et al. 2015) bei der Softwareentwicklung Wissen und Kommunikation als zentral. Den Kern der Theorie bildet das intellektuelle Kapital, welches aus dem Wissen der Organisation (organisationales Kapital wie Dokumentationen, Anleitungen oder der Quellcode selbst), der Fähigkeit von Individuen (Humankapital) und den Beziehungen zu den Kund:innen und Anwendenden besteht (soziales Kapital). Wobei soziales Kapital hilft, die zwei Kapitalsorten (Human, organisational) miteinander zu verbinden (u.a. um implizites Wissen – »tacit knowledge« – auszutauschen und voneinander zu lernen). Zentral ist für die Autorenschaft letztendlich die Kommunikation:

> »Software system development is more of a communication problem than a technical problem« (Wohlin/Smite/Moe 2015: 231)

Wie eine Entwicklungsaufgabe umgesetzt wird, hängt vom intellektuellen Kapital und dem angestrebten Performance-Ziel ab. Das heißt, die Theorie sieht durchaus vor, dass bspw. das intellektuelle Kapital nicht ausreicht, um die Aufgabe umzusetzen. Aufgabe des Managements ist es dann, die Ziele zurückzuschrauben. Eine Best Practice oder spezifische Methode schlagen die Autor:innen nicht vor. Sie haben ein situatives Verständnis von Softwareentwicklung, wobei Wissen, die Kompetenzen der Mitarbeitenden, Kommunikation und gute Beziehungen eine zentrale Rolle spielen. Diese Abkehr von einzelnen Methoden und die Hinwendung zu abstrakteren Zusammenhängen vollzieht bereits ältere Literatur. In Bezug auf Managementstrategien zur Softwareentwicklung seien keine eindeutigen Best Practices auffindbar:

6 In diesem Fall RUP (Rational Unified Process).

> »Policies [of management strategies] pursued depended on the particular task at hand, and on the particular skills, experience levels and even personalities of the staff involved« (Friedman/Cornford 1989: 358).

Die Autoren lehnen bspw. Aussagen von anderen Forschenden ab, die Dequalifizierung (»deskilling«) und direkte Kontrolle oder eine Mischung von »Slack« und direkter Kontrolle allgemein als beste Strategie ansehen (vgl. ebd. 356). Andere Forscher stellen ebenso die

> »lokale Praxis einer inkrementellen Anpassung von Vorgaben, Zielen und Vorgehensschritten an sich wandelnde oder erst spät erkennbare Erfordernisse« (Schulz-Schaeffer 1996: 1)

fest. Ein anderer Autor spricht bei Softwareentwicklung von »Zonen iterativer und kommunikativer Verständigungsprozesse« (Peter 1993: 423), weil nicht vorweg geplant festgelegt werden kann, wann wer über was kommuniziert. Trotzdem seien Entwicklungsmethoden nicht überflüssig. Sie haben einen Wert für den Prozess, weil sie Sicherheit erzeugen. Sie helfen, den Planungsprozess erst einmal in Gang zu setzen und den Beteiligten eine gewisse Handlungssicherheit zu geben (vgl. Schulz-Schaeffer 1996: 15).

Es gibt Fälle, die bestimmten Managementstrategien klare Grenzen aufzeigen. Eine Studie zeigt, wie bei ausgelagerter Softwareentwicklung autoritäre Kontrolle verhindert, dass sich ein gemeinsames Verständnis entwickelt und eine ausreichende Kommunikation stattfindet. Beides wird erst wieder durch vertrauensvolle Beziehungen möglich (vgl. Gregory/Beck/Keil 2013: 1226). Auch frühe Autoren argumentieren, dass nicht einfach mehr Arbeitskräfte produktiver sind, sondern dass Kommunikation entscheidend ist und dass erst über diese nachgedacht werden sollte (vgl. Conway 1968: 31). Bei einer Untersuchung von Softwareprojekten stellt die Autorin fest, dass Demokratisierung die Produktivität einer individualisierten, wissensspezialisierten Belegschaft steigern kann (vgl. Müller 2010: 52f.). Kooperatives Arbeiten sei vielversprechender – ob durch kooperative Planung, Eigeninitiative etc. (ebd. 281f.). Eine Studie zu globalen Softwareprojekten kommt zu dem Schluss, dass nicht mehr das Management das Wissen zentralisiert oder verwaltet, sondern die eingesetzten »[K]oordinationsmechanismen zu Wissensmanagementinstrumenten« (Kotlarsky/Van Fenema/Willcocks 2008: 99) werden. Wenn die Fachleute der Anwendungsbereiche und der Programmierung einer Software zusammenarbeiten sollen, sind statt einer vertikalen Integration oder Märkten Netzwerke die optimale Organisationsform. Das zeigt eine Studie zur Entwicklung eines digitalen Kontrollsystems für Flugzeugtriebwerke (vgl. Brusoni/Prencipe/Pavitt 2001: 610). Setzt sich also das agile Arbeiten mit seinen Kernforderungen durch?

> »The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. […] The best architectures, requirements, and designs emerge from self-organizing teams.« (Beck et al. 2001)

Empirisch existiert ein gemischtes Bild, was die Organisation der Softwareentwicklung anbelangt. Selbstorganisation und direkte und offene Kommunikation sind in effizienz-

getriebenen Organisationen nicht immer vorzufinden: Autoren sehen Fälle, in denen Softwareentwicklung so strukturiert und organisiert werden kann, dass sie »dem Ideal eines strikt vorgeplant-arbeitsteiligen Arbeits- und Produktionsprozesses recht nahekommt« (Schulz-Schaeffer/Bottel 2018: 102). Zugleich räumen sie ein, dass das nicht immer so ist und Softwareentwicklung auch in teamförmigen Abstimmungsprozessen à la Scrum stattfinden kann (vgl. ebd.). Andere Untersuchungen zeigen, dass Scrum nicht automatisch zur Selbstorganisation führt (vgl. Pfeiffer/Sauer/Ritter 2014, vgl. Boes et al. 2018). Unterschiedliche Organisationsformen sind auch bei internationaler Softwareentwicklung zu finden: Zwei Fallstudien stellen einmal eine Industrialisierung und fabrikmäßige Arbeit fest und einmal eine wenig formalisierte, auf eigenverantwortliche Kommunikation setzende Arbeitsweise. Beides sind somit Beispiele für einmal direkte und einmal permissive Kontrolle in der Softwareentwicklung (vgl. Feuerstein 2012). Bei einer anderen Fallstudie hatte allein die Verlagerung der Softwareentwicklung innerhalb Deutschlands »ausgeprägte Tendenzen der Spezialisierung, Abschottung, Verlust an Aufgabenvielfalt und Zuname der Dokumentations- und Kontrollarbeiten« (Flecker/Holtgrewe 2008: 321) zur Folge. Die geringe Formalisierung der geografisch verteilten Arbeit wurde zum Problem. Sie wurde dann stärker standardisiert (vgl. ebd.). Ganz zu schweigen davon, dass es unterschiedliche Vertragsverhältnissen für Programmierende inkl. Selbstständige gab (vgl. ebd. 316f.). Daneben wirken sich die Projektphase oder die Teamgröße auf die Arbeit in der Softwareentwicklung aus: Spätere Phasen in einem Projekt sind strukturierter (Heidenreich/Kirch/Mattes 2008: 13) und bei größeren Teams werden formale Organisation und Dokumentation wichtig (vgl. Ralph 2015: 35). Allein der Typ der Software kann weitreichende Folgen für ihre Entstehung haben, wie eine Gegenüberstellung von Carmel und Sawyer (1998) zeigt: Ob eine Softwarefirma entwickelt oder intern in eine Firma selbst: Laut den Autoren besteht intern eine Matrixorganisation und es läuft bürokratischer ab. Die Softwarefirma arbeitet dahingehend u.a. selbstorganisierter und die Prozesse haben eine geringere Reife. Zudem ist die Realität vieler Softwareentwickelnde, dass sie bestehende Standardsoftware anpassen und deshalb viel Zeit damit verbringen, diese zu beurteilen, zu verändern und andere Lösungen zu integrieren (vgl. Boehm 2006: 21).

Womit wir wieder am Anfang dieses Absatzes angekommen sind: Der Ansatz von Wohlin et al. (2015) ist insofern zielführend, weil er nicht auf feste Methoden oder Managementstrategien wie Standardisierung oder Selbstorganisation setzt, wenn es um die Analyse von Softwareentwicklung geht. Zudem geht es bei der Arbeit hier um jene Phase der Softwareentwicklung, bei der die verschiedenen Kontexte eine noch größere Rolle spielen dürften und ein Wissensaustausch schwieriger zu standardisieren und kontrollieren ist. Um in der Scrum-Begrifflichkeit zu sprechen, geht es hier um die Rolle Product Owner. Sie ist für das Schreiben von Anforderungen zuständig und wie sie an ihre Infos über die Anwendung (in unterschiedlichen Industrien, Firmen oder Abteilungen) kommt. Sie findet sich in unterschiedlichsten Kontexten wieder.

Die Spezifikation einer Software kann auf unterschiedlichen Wegen gelingen. Austausch von Wissen und Kommunikation sind flexibel. Ein Meeting, eine gut formulierte E-Mail, ein klärendes, persönliches Gespräch oder klare Abläufe via Ticketsystem sind allesamt Wege, Anforderungen zu spezifizieren (mehr dazu siehe unter 6.4.4). Es bleibt die Spannung zwischen offenem und direktem Wissensaustausch und Effizienz und Wett-

bewerb, wie sie allgemein für Wissensarbeit typisch ist. Bei ihr basiert Effizienz darauf, »Wissen und Expertise als Rohstoff umzuformen« (Willke 1998: 166). Das Konzept des soziotechnischen Netzwerkes im nächsten Kapitel zeigt, wie Organisationen den Arbeitsprozess der Softwaregestaltung trotzdem kontrollieren können.

Exkurs: Was sind die Bestandteile von Scrum?

Scrum ist eine Methode zur Softwareentwicklung, bei der iterativ Konzepte an Programmierende übergeben und von diesen abgearbeitet werden. Der Scrum-Prozess besteht aus Rollen, Artefakten und Meetings (vgl. Gloger 2009: 11ff.):

Scrum-Prozess	Beschreibung
Rollen	
Product Owner:in	für die Softwarelösung (das Produkt) verantwortlich; pflegt Anforderungen (Items) in eine Liste (Product Backlog) und priorisiert sie für die Programmierenden
Scrum Team	Personen, die notwendig sind, um Anforderungen in Software zu verwandeln; managt sich selbst (inkl. Arbeitsmenge); den Standards und Prozessen von Scrum verpflichtet; für die Qualität verantwortlich
Scrum Master:in	beseitigt Schwierigkeiten, Blockaden und Probleme, die das Team aufhalten; nicht weisungsbefugt, sorgt für Einhaltung Scrum-Prozess; schult Teilnehmende in ihren Rollen
Management	für Ressourcen und Richtlinien zuständig, setzt Rahmen des Scrum-Prozesses, löst von Scrum Master:in identifizierte Probleme
Artefakte	
Product Backlog	Liste mit Anforderungen (Items), je Anforderung schätzt das Team den Aufwand
Sprint	ein Zyklus (z.B. 2 Wochen), in dem Team Items abarbeitet
Sprint Backlog	Liste mit abzuarbeitenden Aufgaben für einen Sprint, wird täglich überarbeitet und aktualisiert
Meetings	
Daily Scrum	Meeting (ca. 15 Minuten) im Team, bei dem Personen sagen: Was habe ich seit dem letzten Daily Scrum erreicht? Was will ich bis zum nächsten Daily Scrum erreichen? Welche Impediments (Hindernisse) stehen mir dabei im Weg?
Sprint Plannings	Treffen für anstehenden Sprint, über Anforderungen und Ziele des im Sprint entwickelten Softwareteils; wie wird Software aufgebaut und welche Architektur soll sie haben?
Sprint Review	Treffen, bei dem das Team Funktionalität am Ende des Sprints präsentiert; Fortschritt wird anhand von »usable Software« demonstriert
Retrospektive	Treffen, in dem das Team die eigenen Arbeitsprozesse optimiert

5.3. Zwischenfazit: Softwaregestaltung als soziologisches Problem

Für die Untersuchung der Formen und Folgen von industriespezifischer Softwaregestaltung hat Kapitel 4 dargestellt, warum Softwaregestaltung zum Kern der Digitalisierung von Wirtschaft und Gesellschaft gehört. Dies gilt auch für von Software durchdrungene Organisationen, die nicht mit Software ihr Geld verdienen. Softwaregestaltung kann in unterschiedlichen Kontexten stattfinden, die auch im weiteren Verlauf der Arbeit noch eine Rolle spielen werden: durch Softwarefirmen, in digitalen Start-ups, durch IT-DL, in den EVU selbst etc. Zudem hat Kapitel 4 für die weitere Analyse grundlegende Begriffe anhand eigener Überlegungen eingeführt, die für das hier vertretene soziotechnische Verständnis moderner Organisationen stehen: softwaretechnische Interdisziplinarität und softwaretechnische Gestaltungsmöglichkeiten (bestehend aus softwaretechnischer Ausrichtung und Zuschnitt). Eine Variante der Letzteren ist der Primat der Softwareentwicklung, bei dem eine Organisation von Anfang an auf die Softwaregestaltung ausgerichtet ist (softwaretechnische Ausrichtung) und für sich eine individuelle Software gestaltet (softwaretechnischer Zuschnitt). Eine andere Variante ist, dass sich eine Organisation auf die Anwendung einer Standardsoftware konzentriert.

Um begrifflich zu klären, was bei der Softwareentwicklung der Mensch macht und was die Maschine, hat Kapitel 5 zwischen den Begriffen Daten, Informationen, Wissen und Kommunikationen unterschieden. In Abgrenzung zu anderen Theorien über das Verhältnis von Menschen und Technik folgt die Untersuchung dem kritisch-realistischen Ansatz von Mingers/Willcocks (2014). Demnach unterscheiden sich die drei Welten von Person, Sozialem und Technik ontologisch und epistemologisch voneinander und zwischen Mensch und Technik besteht eine Arbeitsteilung: Der Mensch versteht, interpretiert und vermittelt zwischen Software und Umwelt und ist dabei in eine soziale Welt eingebunden. Kommunikation und Wissen sind seine Domänen.

Um die Softwaregestaltung als Arbeitsprozess zu verstehen, hat Kapitel 5 gezeigt, dass sie und warum sie wesentlich auf Wissen und Kommunikation basiert. Das hat technologische (u.a. zeichenbasierte Technologie, mehrere technische Schichten, sprachliche Strukturierung u.a. durch Begriffe) und organisatorische Gründe (u.a. verstärkte Einbindung von Anwendenden). Es zeigt sich am historischen Wandel der Softwareentwicklung und ihrer Methoden über die Jahrzehnte hin zu einem in vielen Kontexten weitverzweigten, vernetzten, kollektiven Kommunikationsprozess.

Die Mitarbeitenden an der Softwaregestaltung machen das, was der Computer nicht kann: Sie tragen ihr Wissen bei und kommunizieren. Das müssen sie tun, wenn sie die notwendige softwaretechnische Interdisziplinarität herstellen wollen. IT-Fachleute wie Programmierende und fachliche Expert:innen wie Anwendende müssen sich austauschen. Dabei sind sie damit konfrontiert, dass Software unterschiedliche Schichten hat: Beispielsweise kennen die Programmierenden in erster Linie den Quellcode, während die Anwendenden die Bedienungsoberfläche der Software aus ihrer täglichen Arbeit kennen. Nicht jede:r hat die gleiche Perspektive auf die Software-Oberflächen bzw. spielt die Software die gleiche Rolle im Arbeitsalltag. Die Beteiligten der Softwaregestaltung haben unterschiedlichen Einblick in die Algorithmen, verstehen nicht alle Programmiersprachen oder das Gleiche unter softwaretechnischen Begriffen wie Softwarearchitektur, Schnittstelle oder Modell. In der Softwaregestaltung kann es dazu

kommen, dass es für jede Perspektive eigene Spezialist:innen gibt: für die Programmierung, Softwarearchitektur, Datenbanken, den Anwendungsbereich Prozessteil A der zu gestaltenden Software und jenen von Prozessteil B usw.

Im Zuge der Softwaregestaltung kann Software einerseits keine Blackbox bleiben und andererseits kann sich nicht jeder intensiv mit ihrem Innenleben beschäftigen. Beides wirkt sich auf Kommunikation und Wissen in der Softwaregestaltung aus. So können Berechtigungsstrukturen, die den Zugang zur gesamten Software oder einzelne ihrer Funktionalitäten regeln, es erschweren, an das notwendige Wissen zu kommen, was in der Software steckt. Dann muss möglicherweise der/die Expert:in der Softwarefirma oder ein:e IT-Berater:in hinzugezogen werden. Zudem haben Beteiligte nicht immer die Möglichkeit, sich sämtliches Wissen über die Software und ihre Anwendung anzueignen, wenn sich die Software stetig verändert, der gewohnte Arbeitskontext sich durch eine neue Software oder ein Softwareupdate verändert hat oder die Software im Laufe der Zeit immer komplexer geworden ist. Software ändert sich oftmals stetig. Das Wissen über sie kann schnell veralten. Das liegt auch daran, dass eine Standardsoftware einen Lebenszyklus hat. Die anbietende Softwarefirma löst alte Versionen ab, ohne auf die Zustimmung sämtlicher Anwendenden zu warten. All das hat Folgen für den kommunikativen Austausch und Wissenstransfer: die verschiedenen Perspektiven und Wissensstände der Stakeholder:innen einer Software, welche Veränderungen durch wen überhaupt an der Software oder an der Organisation möglich sind, stetige Änderungen an der Software oder gar die Ablösung einer Software durch eine neue Version.

Historisch betrachtet wurde es über die Jahrzehnte immer wichtiger zu berücksichtigen, dass Softwareentwicklung kein rein technisches Problem ist (vgl. Friedman/Cornford 1989, Funken 2001, Boehm 2006). Statt sie wie Fließbandarbeit zu strukturieren, haben die Autoren des agilen Manifests 2001 einen Gegenentwurf zu dieser Arbeitsorganisation veröffentlicht (vgl. Beck et al. 2001). Das Forschungsfeld des Anforderungsmanagements der Informatik zeigt, dass Kooperation, Kommunikation und soziale Kompetenzen wichtig für die Softwareentwicklung sind. Sprache, Übersetzungsfähigkeit zwischen IT- und energiewirtschaftlichen Fachleuten, intensive Kommunikation und gar die Anwendung sozialwissenschaftlicher Methoden zur Anforderungsaufnahme stehen im Vordergrund (vgl. Alvarez 2002, Ross/Chiasson 2011, Kaminski 2012, Corvera Charaf/Rosenkranz/Holten 2013, Alsanoosy/Spichkova/Harland 2020). Diese Ergebnisse stellen eine erste Grundlage für den Arbeitsprozess der Softwaregestaltung dar. Doch berücksichtigen sie keine arbeits- und organisationssoziologische Literatur, die Softwareentwicklung in unterschiedlichen Kontexten untersucht. Eine kurze Aufarbeitung dieser Literatur konnte zeigen, dass es nicht die beste Methode oder Organisationsform für alle Fälle gibt. Stattdessen existieren lokale Praktiken und die Empirie zeigt unterschiedliche Organisationsformen der Softwareentwicklung. Ein allgemeines Konzept, um die Softwaregestaltung arbeitssoziologisch zu analysieren, wäre aber hilfreich. Die allgemeine Theorie der Softwareentwicklung ist der Versuch, ein solches allgemeines Konzept aufzustellen (vgl. Wohlin/Šmite/Moe 2015). Sie berücksichtigt die Kontextabhängigkeit von Softwareentwicklung und legt sich nicht auf bestimmte Organisationsstrukturen, Abläufe oder Managementmethoden fest. Indem sie aber organisationssoziologische und arbeitssoziologische Erkenntnisse unterschlägt,

fehlt die Berücksichtigung von unterschiedlichen Arbeits- und Organisationskontexten und wie sich diese auf die Softwareentwicklung auswirken.

Mit dem Konzept der softwaretechnischen Netzwerkarbeit, was das nächste Kapitel entwickelt, lässt sich der aus der Empirie der Fallstudien entwickelte Analyserahmen besser konzeptionell in die Forschungslandschaft einbetten. Das Konzept ist auf die Phase der Softwaregestaltung zugeschnitten, berücksichtigt die für die Softwaregestaltung wesentlichen Kontextfaktoren und ist zugleich so allgemein, dass es sowohl agile wie auch weniger agile Organisationsformen abdeckt.

6. Softwaregestaltung – konzeptionelle Grundlagen

Soziotechnische Netzwerkarbeit und soziotechnische Arbeitsgestaltung zwischen Anwendung und Programmierung

Um die Frage nach den Formen und Folgen der Softwaregestaltung zu untersuchen, haben die vorhergehenden Kapitel deren materielle Basis und die daraus resultierenden Besonderheiten im Unterschied zu anderen Arbeitsprozessen herausgearbeitet. Neben den Kernproblemen der softwaretechnischen Interdisziplinarität[1] und der softwaretechnischen Gestaltungsmöglichkeiten[2] (siehe 4.2) sind Wissen und Kommunikation zentral. Einerseits muss es Einzelnen möglich sein, sich auf komplexe technische Objekte wie Software mit ihren verschiedenen technischen Schichten und sprachlichen Strukturierungen einzulassen (siehe 5.1.2) und sich mit komplexeren (energie)fachlichen Themen oder Anwendungsbereichen auseinanderzusetzen. Sie müssen sich mit umfangreichen Wissensdomänen beschäftigen können. Andererseits muss es möglich sein, sich über beides mit anderen auszutauschen, Möglichkeiten der Softwareentwicklung und fachliche Bedarfe abzugleichen und sich auf eine Umsetzung zu einigen. Dabei ist diese Arbeit Einzelner wie auch die wissens- und kommunikationsintensive Zusammenarbeit untereinander nicht einfach hierarchisch vorstrukturier- und standardisierbar oder einfach monetär zu bewerten und dann über Marktbeziehungen zu erledigen. Diese Zusammenhänge und Voraussetzungen arbeitet dieses Kapitel anhand von Forschungsliteratur heraus. Dies zeigen aber auch die Fallstudien im 8. Kapitel, wo die Zusammenarbeit in sehr unterschiedlichen Konstellationen und teils auf mehrere Organisationen verteilt stattfindet.

In diesem Kapitel geht es nun darum, eine konzeptionelle Grundlage zu schaffen, um zu beschreiben, wie die Organisationen in den Fallstudien die beiden Kernprobleme lösen und die wissens- und kommunikationsintensive IT-Arbeit[3] kontrollieren und damit

1 Wissen über Programmierung und Wissen über den Anwendungsbereich.

2 a) organisatorisch: z.B. Ausrichtung auf Software oder auf Softwareentwicklung; b) softwaretechnisch: z.B. Zuschnitt auf Standardsoftware oder individuellen Quellcode.

3 Softwaregestaltung unterscheidet sich von der Kommunikationsarbeit, wie sie Kruse (2004) allgemein als charakteristisch für IT-Arbeit ausgemacht hat: Für ihn ist sie Interaktion der IT-Arbeitenden mit Computertechnik sowie Kommunikation mit den Kund:innen und das diskursive

die Möglichkeiten der Softwaregestaltung zwischen Individual- und Standardsoftware. Dies geschieht **erstens** dadurch, dass die vorliegende Untersuchung unterschiedliche Formen der Organisation von Softwaregestaltung konzeptionell als Transformation der Arbeitskraft der Beteiligten an ihr beschreibt. Wie in der Einleitung bereits ausgeführt, bezeichnet das Transformationsproblem die Frage, wie Firmen die (in der Softwaregestaltung) eingesetzte, auf dem Arbeitsmarkt gekaufte Arbeitskraft in ihren Organisationen in reale Arbeitsleistung überführen können (vgl. Marrs 2010: 331). **Zweitens** geht es bei den Folgen der Softwaregestaltung darum, wie sich der Arbeitsprozess der Softwaregestaltung zur Softwareanwendung verhält. Denn dieses Verhältnis ist in den Fallstudien sehr unterschiedlich und hat Folgen für die soziotechnische Arbeitsgestaltung in den anwendenden Organisationen: Welche Konflikte bestehen zwischen Softwaregestaltung und -anwendung und wie beeinflussen sie sich?

Als konzeptionelle Grundlage, um den Arbeitsprozess zu analysieren, verwendet die Forschungsarbeit den Begriff des Netzwerks. Warum? In den Fallstudien findet die Arbeit entweder in einer Matrixorganisation oder in einer reinen Netzwerkorganisation statt. Für die Matrixorganisation ist IT-Projektarbeit typisch, die in einer anwendenden Organisation quer zu einer Linienorganisation aus Abteilungen wie IT, Vertrieb, Netzbetrieb oder Logistik und in einer Aufbauorganisation, bestehend aus mehreren Hierarchieebenen, stattfindet. Wenn sie firmenübergreifend erfolgt, kann eine Mischung aus Hierarchie, Markt und Netzwerk bestehen. In reinen Netzwerken kann es erstens so weit gehen, dass es keine formalen Hierarchien gibt und Führungskräfte unwichtig z.B. für die Vorgabe von Arbeitspaketen sind. Zweitens bestehen keine Marktverhältnisse und z.B. Auftraggeber und -nehmer Arbeit(sschritte) nicht preislich bewerten. Ob Matrixorganisation oder reines Netzwerk, zentralisierte Softwaregestaltung in einer Softwarefirma oder dezentral in einem EVU: Immer geht es in den Fallstudien um horizontale Kooperation und Kommunikation – ob zwischen Einzelpersonen des gleichen Teams oder unterschiedlichen Teams, Abteilungen, Organisationen. Das Kapitel geht bei der Konzeption des Netzwerkbegriffs über jenen hinaus, den Kruse für die IT-Arbeit verwendet. Es trifft zwar zu, dass Softwaregestaltung wie IT-Arbeit im Allgemeinen eine »Netzwerkarbeit zwischen den sozialen und technischen Netzen und […] Netzwerkarbeit innerhalb der sozialen Netzwerke« (ebd. 320) ist. Allerdings reicht diese Beschreibung nicht dafür aus, um konzeptionell einzubetten, was in den Fallstudien passiert, um das Transformationsproblem im Arbeitsprozess der Softwaregestaltung zu lösen.

Vielmehr führt das Kapitel einen eigenen Begriff von Netzwerk ein: der **soziotechnischen Netzwerkarbeit**. Mit diesem Konzept ist es möglich, die in der Empirie untersuchten Formen der Softwaregestaltung, die in sehr unterschiedlichen Konstellationen statt-

---

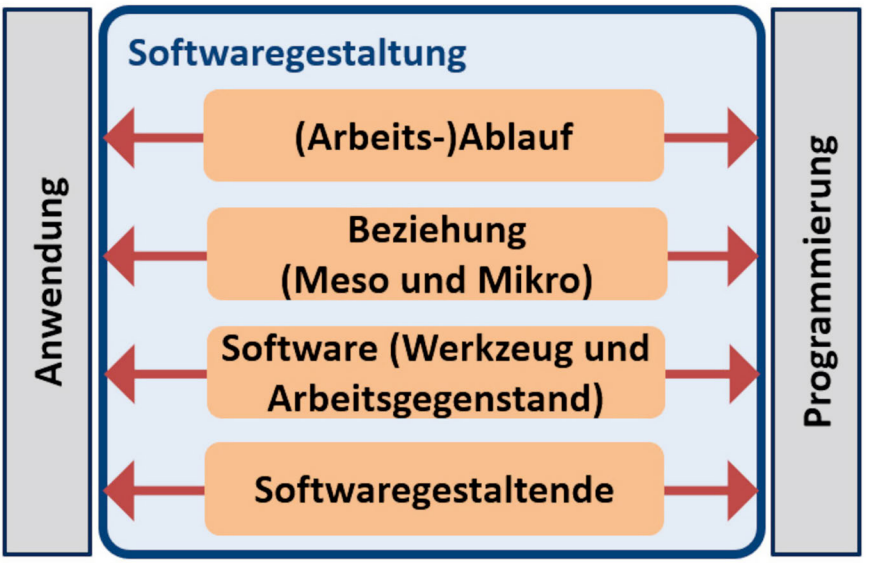
und reflexive Nachdenken, wie diese Kommunikation zu gestalten ist (vgl. ebd. 295). Seine befragten Personen verstehen sich allesamt als Dolmetschende zwischen zwei Welten (vgl. Kruse 2004: 315). Sie realisieren ihre Arbeitsvollzüge durch Kommunikation (vgl. ebd. 324). Für sie ist ein kommunikativer Erfahrungs- und Informationsaustausch notwendig, um die Überlastung mit Informationen bewältigen zu können (vgl. ebd. 297f.). Bei der Softwaregestaltung geht es dagegen im Kern vielmehr darum, via einen kollektiven Kommunikationsprozess eine quelltextbasierte Technik zu gestalten. Kruse hat folgende IT-Arbeitenden interviewt: Systemadministrator, PC-Techniker, Internetcafébetreiberin, IT-Projektmanager, Web-Designerin, Internetdienstleister (vgl. Kruse 2004: 156ff.).

finden, analytisch zu fassen. Das Konzept der soziotechnischen Netzwerkarbeit zeigt, dass Softwaregestaltung ein soziologisches und kein informationstechnisches Problem ist. Zudem verankert es den in den Fallstudien entwickelten Analyserahmen in der Forschungslandschaft. Dazu fasst das Kapitel ausgehend von Forschung zu organisationalen Netzwerken und IT-Projekten Netzwerke als Mehr-Ebenen-Gebilde auf. Es beschreibt, wie die Transformation der Arbeitskraft teilweise ohne Hierarchien oder Märkte gelingt, lässt aber auch eine Mischung aus Netzwerk, Markt und Hierarchie zu. Wie gelingt die Transformation der Arbeitskraft in der soziotechnischen Netzwerkarbeit? Indem vier Ebenen zusammenwirken:

1. Durch einen **Ablauf,** der festlegt, wie Anforderungen entstehen, wie sie zu den Programmierenden gelangen, wer die Beteiligten sind und welche verschiedenen Feedbackmechanismen es zwischen Anwendung und Programmierung gibt.
2. Durch **Beziehungen**, welche helfen, die Arbeitsteilung zwischen Anwendung und Programmierung zu überbrücken. Dazu gehört erstens eine kooperative Zusammenarbeit auf **organisationaler** Ebene (Meso) zwischen IT-Abteilung und Fachbereichen bzw. IT-Dienstleistungsunternehmen (IT-DL), Softwarefirma und anwendenden Organisationen. Die Zusammenarbeit ermöglicht eine entsprechende übergreifende Steuerungsstruktur, auch wenn Interessendifferenz und Machtungleichgewichte zwischen den beteiligten Organisationen bestehen. Dazu gehören zweitens die **interpersonalen** Beziehungen (Mikro), die auf Vertrauen, Kooperationsbereitschaft und Reziprozität basieren. Die Fallstudien im 8. Kapitel und die Forschung zeigen, dass diese notwendig sind, damit Mitarbeitende kooperativ zusammenarbeiten, und dass rein formale Abläufe wie IT-Projekte oder Scrum nicht ausreichen.
3. Für die Transformation der Arbeitskraft ist neben Ablauf und Beziehungen die **Software** entscheidend. Im Gegensatz zur überwiegenden Diskussion über die Folgen der Digitalisierung für die Arbeit kontrolliert sie Arbeit nicht nur einschränkend, überwachend und steuernd. Vielmehr ermöglicht sie selbstständiges, kommunikations- und wissensintensives, kooperatives Zusammenarbeiten – sei es durch Ticketsysteme, E-Mail-Programme oder Projektmanagementlösungen. In einigen Fallstudien des Empirie-Kapitels ist Software auch ein Kontrollinstrument für Führungskräfte oder Kundschaft. In anderen Fallstudien ist sie nur dazu da, eine horizontale, abteilungs- oder organisationsübergreifende Kooperation und den Input für Anforderungen einzelner Beteiligter zu ermöglichen. Darüber hinaus prägt die Softwarearchitektur die Softwaregestaltung. Sie entscheidet beispielsweise darüber, wie Firmen eine Standardsoftware erweitern oder anpassen können und welche Abhängigkeiten zwischen Organisationen bestehen (z.B. ob einzelne Teams oder Organisationen unabhängig von anderen einen Teil einer Software gestalten können).
4. Die vierte Ebene sind die **Softwaregestaltenden** und ihre Rollen, die sie jeweils im Arbeitsprozess einnehmen. Das Kapitel zeigt, wie das Rollenkonzept zur Beschreibung der Kontrolle der Arbeit von Softwaregestaltenden eingesetzt werden kann. Diese bekommen weniger konkrete Arbeitsschritte vorgegeben, als dass vielmehr Erwartungen an sie bestehen, die sie zu erfüllen haben: z.B. dass sie soziale Strukturen wie IT-Projekte etablieren, Treffen organisieren, sich kooperativ verhalten oder

selbstständig und eigeninitiativ arbeiten und Ergebnisse liefern. Zudem stellen die Softwaregestaltenden betriebliche Hierarchien in Frage. Gestaltet beispielsweise ein EVU abteilungsübergreifend eine Software, macht es einen Unterschied für die Umsetzung der Softwaregestaltung, ob die Softwaregestaltenden hierarchisch über oder unter den Abteilungsleitenden stehen. Welche Entscheidungen können sie im Sinne der Softwaregestaltung durchsetzen und welche nicht? Werden sie Teil des Managements?

Abbildung 6: Vier Forschungsbereiche, die konzeptionelle Bezüge für soziotechnische Netzwerkarbeit zwischen Anwendung und Programmierung liefern sollen, und zugleich die vier Ebenen, auf denen die Kontrolle zur Transformation der Arbeitskraft bei soziotechnischer Netzwerkarbeit basiert.



Zur konzeptionell adäquaten Beschreibung, wie in den Fallstudien Organisationen Softwaregestaltung kontrollieren, gehört, dass die vier Ebenen nicht immer in gleichem Maße zur Transformation der Arbeitskraft beitragen. Es ist typisch für die soziotechnische Netzwerkarbeit, dass sich die vier Ebenen flexibel ergänzen können, z.B. dass einzelne Softwaregestaltende durch ihre Arbeit schwach formalisierte Abläufe kompensieren.

Die Möglichkeiten der Softwaregestaltung zwischen Individual- und Standardsoftware nützen zu können, indem eine Organisation einen entsprechenden Arbeitsprozess der Softwaregestaltung etabliert und das Transformationsproblem löst, ist das eine interessante Untersuchungsfeld. Das andere ist, wie sich der Arbeitsprozess der Softwaregestaltung auf jenen der Softwareanwendung auswirkt und wie er die Arbeit der Anwendenden kontrollieren hilft. Dieses bestehende Forschungsdesiderat beleuchtet und untersucht die vorliegende Dissertation mit dem Konzept der **soziotechnischen Arbeitsgestaltung**. Dafür arbeitet das Kapitel anhand von Forschungsliteratur heraus, inwie-

fern es sich bei der Softwaregestaltung um eine besondere Form der Rationalisierung der Softwareanwendung handelt. Zugleich grenzt es diesen Ansatz von anderen ab, die das Verhältnis von IT und Arbeit beschreiben (z.B. jenen der Informatisierung). Die Forschung zu den Folgen von Standard-ERP-Software für die Softwareanwendung rezipiert der letzte Abschnitt. Allerdings lassen sich viele dieser Folgen entweder auf die Funktionalität der Software zurückführen oder darauf, dass es sich um eine fertige Standardsoftware handelt und weniger auf den Gestaltungsprozess.

Das Kapitel zeigt zunächst Forschungslücken auf und wie sich die vorliegende Arbeit an die Diskussion über die Kontrolle von Wissensarbeit anschließt. Es untermauert, dass sich der Netzwerkbegriff besser eignet als jene von Markt oder Hierarchie, um zu beschreiben, wie der Arbeitsprozess der Softwaregestaltung die Arbeitskraft transformiert. Dann führt ein Abschnitt Forschungsarbeiten zu IT-Projekten an, denn IT-Projekte sind Beispiele für soziotechnische Netzwerkarbeit, und auch sie ermöglichen auf den vier Ebenen die Transformation der Arbeitskraft. Danach betten mehrere Forschungsarbeiten konzeptionell ein, wie die Ebenen der organisationalen und interpersonalen Beziehungen, die Ebene der Software und der Softwaregestaltenden die Kontrolle von Softwaregestaltungsarbeit ermöglichen. Zuletzt geht es um das Verhältnis von Softwareanwendung und Softwaregestaltung.

6.1. Softwaregestaltung als Arbeitsprozess: Die Lösung des Transformationsproblems durch soziotechnische Netzwerkarbeit

Die Frage nach Kontrollformen, welche zu einer kooperativen Wissensarbeit wie der Softwaregestaltung passen, ist Gegenstand einer umfangreichen Diskussion in der Forschung. Laut Kalkowski/Mickler stoßen Hierarchien und formale Strukturen bei der Transformation von Arbeitskraft bei kooperativen Projekten an ihre Grenzen, weil sie zu wenig Handlungsspielraum zulassen (vgl. Kalkowski/Mickler 2015: 38). Um Softwaregestaltung zu organisieren, ist eine andere Form der Kontrolle notwendig:

> »[A] shift from behaviour control toward knowledge control; the goal of the latter is to elicit as much knowledge as possible from knowledgeable workers« (Rennstam 2012: 1072).

Ob Adler/Borys (1996) Rede einer befähigenden (»enabling«) Formalisierung, Friedmans (1977) verantwortliche Autonomie oder die Diskussion über die Subjektivierung von Arbeit: In allen drei Fällen geht es darum, dass bestimmte Formen von Arbeit so zu organisieren sind, dass trotz Freiräumen die Beschäftigten zum Organisationszweck beitragen. Anders als in Bürokratien erwartet die Softwaregestaltung strukturbedingt von den Wissensarbeitenden – in diesem Fall den Softwaregestaltenden –, dass sie ihre Subjektivität einbringen. Subjektivität ist kein Störfaktor mehr wie im Taylorismus (vgl. Minssen 2017: 303). Betriebe haben »erhöhten funktionalen Bedarf an Subjektivität« (Minssen 2011: 119). Sie ist relevant geworden für die Rationalisierung (vgl. ebd. 118). Softwaregestaltende warten nicht auf detaillierte Anweisungen durch die Forgesetzten, sondern werden zu »›Mittätern‹ der Kontrollregime« (Schaupp 2021: 114).

Netzwerkformen wie Projekte oder Kooperationen zwischen Firmen zeichnen sich dadurch aus, dass sie diesen Raum für Subjektivierung bei gleichzeitiger Handlungsstrukturierung nutzen. In ihnen koexistieren verschiedene Kontrollformen nebeneinander. Sie sind Beispiele dafür,

> »how these coercive and dominant forms of control coexist and interact with the more enabling and knowledge-eliciting form[s]« (Rennstam 2012: 1086).

Was eine besondere Herausforderung bei der Softwaregestaltung ist: Es geht um Wissensarbeit in team-, abteilungs- und organisationsübergreifenden Konstellationen. Die Forschung benennt die Verbindung von Arbeitsprozess und organisationalen Netzwerken in und zwischen Unternehmen als eine Forschungslücke:

> »In consequence, we do not only need more differentiated theoretical frameworks originating from the work of Gereffi[4] and others but much more empirical studies that connect the level of the work process with the organizational level of corporate or inter-corporate relations.« (Sydow/Helfen 2020: 225)

Zudem gibt es noch ein weiteres Forschungsdefizit in diesem Bereich:

> »Insgesamt handelt es sich bei diesen Zusammenhängen zwischen praktischer Kooperation, Netzwerkformen, Nutzung von IuK-Techniken, Wissenstransfer und Arbeit jedoch um ein wenig untersuchtes Gebiet, d.h. es besteht ein ausgeprägtes Forschungsdefizit.« (Schmiede 2006: 466)

Das Konzept der soziotechnischen Netzwerkarbeit kann einen Beitrag dazu leisten, diese Forschungslücken konzeptionell zu schließen, wie die nächsten Kapitel zeigen.

6.2. Weder Markt noch Hierarchie: Netzwerke als analytische Grundlage

Der Netzwerkbegriff eignet sich besonders gut, um die vielfältigen empirischen Begebenheiten der Softwaregestaltung in den Fallstudien zu beschreiben: Sie findet mal in Matrixorganisationen statt, mal in reinen Netzwerkorganisationen; mal dezentral in einem Team und mal zentralisiert in einem IT-DL, der mit mehreren EVU zusammenarbeitet. Der Netzwerkbegriff ist zudem anschlussfähig an die Forschung der letzten Jahrzehnte. Diese nutzt den Begriff, um Veränderungen in der Arbeitswelt zu beschreiben. Der Begriff der Vermarktlichung ist hier ungeeignet, da er ein anderes Kernphänomen adressiert, nämlich dass Firmen Marktmechanismen in ihrer Organisation nutzen (vgl. Sauer 2018). Der Begriff der Informatisierung beschreibt eine allgemeine Entwicklung und weniger, wie und warum Arbeit auf eine bestimmte Art und Weise organisiert ist, um Technik zu gestalten (vgl. Schmiede 2017).

4 Bekannt für seine Governance-Typen für Wertschöpfungsketten: market, modular, relational, captive, hierarchy (vgl. Gereffi/Humphrey/Sturgeon 2005).

Die Forschung zeigt, dass es unterschiedliche Gründe gibt, warum der Netzwerkbegriff nützlich für die Analyse der Softwaregestaltung ist: Es sind 1. theoretische (weder Markt noch Hierarchie), 2. organisatorische (vertikale Desintegration, Wissensgrenzen, Projektarbeit) und 3. Technische Gründe (Verbindung zwischen IT und organisatorischen Netzwerken).

6.2.1. Theoretisch: Netzwerke in Abgrenzung zu Markt und Hierarchie

Seit mehr als 30 Jahren beschreiben Forschende eine sich von Märkten und Hierarchien unterscheidende Organisationsform (z. B. Williamson 1985, Powell 1990, siehe dazu Windeler/Wirth 2010). Allgemein wird eine Krise der vertikalen, bürokratischen Organisation und abgeschwächten Firmengrenzen konstatiert (vgl. Ahrne/Brunsson 2011). Theoretisch besteht ein weitreichender Konsens, dass mit Kategorien wie Markt oder Hierarchie allein nicht mehr sämtliche Organisationsformen beschrieben werden können. Was nun genau dieses fehlende Dritte sein soll, ist umstritten. Das fängt bei begrifflichen Fragen an: Zur Diskussion stehen u.a. die Begriffe Kooperation, Gemeinschaft, Clans, Netzwerke, langfristige Beziehungen. Auch darüber, wie diese Begriffe inhaltlich gefüllt werden sollen, besteht eine rege Diskussion. Bei Autor:innen wie Lamoreaux/Raft/Temin (2003), Bradach/Eccles (1989) oder Wiesenthal (2000) können die drei Organisationsformen kombiniert werden (und werden empirisch auch kombiniert). Der Netzwerk-Begriff enthält für Ahrne/Brunsson (2011) keine organisatorischen Elemente: Es ist flexibel und spontan (vgl. ebd. 97). Die Beziehungen in Netzwerken sind nicht hierarchisch, sondern basieren auf Reziprozität, Vertrauen und sozialem Kapital (vgl. ebd. 88). Es gibt zudem keinen einheitlichen theoretischen Ansatz. Mehrere Forschungsarbeiten nutzen die Strukturationstheorie von Giddens (vgl. Windeler/Wirth 2010: 580ff., Longen 2015). Williamson (1985) verwendet die ökonomische Transaktionskostentheorie.

Begrifflich hat sich die vorliegende Arbeit bereits für das »Netzwerk« als dritte Kategorie entschieden. Konzeptionell folgt diese Arbeit Powell (1990), Kalkowski/Micker (2015) und Sydow/Windeler (2000), die von Netzwerken als eigenständigen Organisationsformen ausgehen. Dabei sind Netzwerke nichts Eindimensionales. Für Sydow/Windeler (2000), Apitzsch (2006) oder Kalkowski/Mickler (2015) bestehen Netzwerke aus mehreren Ebenen und Dimensionen. Wie in der Matrixorganisation können sich verschiedene Kontrollformen mischen. Für Apitzsch sind unterschiedliche Beziehungseigenschaften auf interpersoneller, organisationaler und interorganisationaler Ebene kombinierbar. Was an Inhalten ausgetauscht wird (bspw. Informationen), muss nicht mit einer bestimmten Beziehungsart (bspw. starke Formalisierung), Beziehungsbasis (bspw. Vertrauen) oder Intensität (bspw. stabil und längerfristig) einhergehen (vgl. Apitzsch 2006: 21f.). Bei Kalkowski/Mickler sind auf den Ebenen der organisationsübergreifenden Kooperationsstruktur, der Projektorganisation und des Kooperationsverhaltens auch verschiedene Kombinationen möglich, bspw. ein hierarchischer Kooperationstyp, eine schwach formalisierte Projektstruktur und regelmäßige Interaktionen (vgl. Kalkowski/Mickler 2015: 88).

Bei der genaueren konzeptionellen Beschreibung der unterschiedlichen Ebenen des Netzwerks setzt die vorliegende Untersuchung auf eigene Kategorien. Denn andere Ansätze sind (noch) allgemein(er) und ihre Begrifflichkeiten helfen nicht, die Besonderhei-

ten der Softwaregestaltung zu erfassen. Es gibt sicherlich Anknüpfungspunkte mit dem oben genannten strukturationstheoretischen Ansatz von Giddens oder jenem allgemeinen Kooperationsbegriff von Kalkowski/Mickler (2015). Wie Giddens begreift Sydow und auch die vorliegende Arbeit Struktur und Handeln als rekursiven Zusammenhang. Allerdings ist der begriffliche Apparat (vgl. Giddens 1988, Kalkowski/Mickler 2015: 68ff.) nicht auf die Softwaregestaltung ausgerichtet[5]. Die in der Einleitung des Kapitels beschriebenen vier Ebenen der soziotechnischen Netzwerkarbeit eignen sich besser, um die Empirie des Arbeitsprozesses der Softwaregestaltung und seine spezifischen Probleme zu erfassen.[6] Zudem spielt Technik bei den genannten Autoren keine wesentliche Rolle für die Netzwerkarbeit (nur z.B. zur Kontrolle der Projektarbeit, vgl. Kalkowski und Mickler 2015: 88). Ferner räumen sie, anders als die vorliegende Arbeit, Wissensarbeitenden und deren Rollen analytisch keinen zentralen Platz in Netzwerken ein. Bei Sydow/Windeler (2000) sind Individuen zwar eine Steuerungsebene (ebd.: 3f.), aber es bleibt unklar, was deren Beitrag zu den Netzwerkstrukturen ist. Als Rolle ist nur vom »boundary spanner[7]« die Rede (vgl. Sydow/Windeler 2000: 5, 10; auch Kalkowski/Mickler 2015: 74). Letztendlich sind die Ansätze zu wenig auf die Softwaregestaltung ausgerichtet, bei der Wissensarbeitende und IT eine zentrale Rolle spielen.

6.2.2. Organisatorisch: Netzwerke aus und in Organisationen

Die Softwaregestaltung findet in einem organisatorischen Kontext statt, zu dem der Netzwerkbegriff passt, wie die nachfolgend zitierte Forschung zeigt: Erstens lagern immer mehr Organisationen (IT-)Tätigkeiten aus (vgl. Miozzo/Grimshaw 2005: 1421f., vgl. Flecker/Haidinger/Schönauer 2013, vgl. Howcroft/Richardson 2012: 124f., vgl. Puranam/Alexy/Reitzig 2014: 172f.). Dies führt zweitens dazu, dass statt klarer Hierarchien ein Spannungsverhältnis zwischen zentraler Steuerung und dezentraler Autonomie besteht (vgl. Hirsch-Kreinsen 1995). Wie die Fallstudien des Empirie-Kapitels zeigen werden, ist das typisch für die Softwaregestaltung. In Organisationsnetzwerken kann beides vorhanden sein: zentrale Vorgaben von Budgets und eine Selbststeuerung der

5 Bspw. ist die Unterscheidung zwischen zweckrationalem und kommunikativem Handeln (Kalkowski/Mickler 2015: 69) für die Softwaregestaltung schwierig. Die Softwaregestaltung verwendet Kommunikation instrumentell und es ist fragwürdig, ob es wirklich um Wahrhaftigkeit und Verständigung geht, wie es für kommunikatives Handeln gilt: »Rationalitätsmaßstab sei die Wahrhaftigkeit intentionaler Äußerungen und die Richtigkeit von Normen« (Vormbusch 2002: 107). Ausgiebig mit den Grenzen der Theorie von Habermas für die Anwendung auf die Softwaregestaltung hat sich Andelfinger (1997) auseinandergesetzt (sie wird trotzdem verwendet – z.B. Ross und Chiasson 2011). Aber auch die anderen Begriffe der Theorien (bspw. Sozialintegration vs. Systemintegration bei Giddens) sind zu weit weg vom empirischen Kern der Softwaregestaltung.

6 Wie sich die jeweiligen theoretischen Ansätze mit den empirischen Ergebnissen vertragen, wäre eine eigene Studie wert. Dies konnte die hier vorliegenden Arbeit nicht leisten – auch weil die empirischen Ergebnisse nicht so einfach in bestehende Theorien integrierbar sind bzw. die Softwaregestaltung ihre Eigenheiten hat, die sich nur mit eigenen Schwerpunktsetzungen in der konzeptionellen Begriffsbildung erfassen lassen.

7 Nach einer Definition handelt es sich bei »boundary spanners« um »vital individuals who facilitate the sharing of expertise by linking two or more groups of people separated by location, hierarchy, or function.« (Levina und Vaast 2005)

Anwendungsarbeit, weil Netzwerke den Arbeitsvollzug unbestimmt lassen und es so ermöglichen, dass Beschäftigte einen subjektiven Beitrag leisten (vgl. Schmiede 2006: 4). Aber es kann auch eine »Zentralisierung und Hierarchisierung von Unternehmensnetzwerken […] [und] asymmetrische Beherrschungsverhältnisse« (Apitzsch 2006: 10) bestehen, z.B. wenn die Softwarefirma den Standard setzt oder Einstellungsmöglichkeiten des Standards vorgibt.

Drittens helfen Netzwerke, interdisziplinäre Wissensgrenzen zwischen Firmen oder innerhalb von Firmen zu überwinden. Kooperationen zwischen Firmen sind eine Form davon (vgl. Kalkowski/Mickler 2015). Erst in Netzwerken können abteilungs- und firmenübergreifende, interdisziplinäre Praxisgemeinschaften entstehen (»communities of practice« im Sinne von Wenger 1999), die für den Wissensaustausch notwendig sind:

»Where practice doesn't prepare the ground, knowledge is unlikely to flow« (Brown/Duguid 2001: 207).

Ein Beispiel für solche Praxisgemeinschaften sind Projekte. Sie sind als vierter Punkt, der aus organisatorischer Sicht für den Netzwerkbegriff spricht, zentral in der heutigen Arbeitswelt[8]. Für Apitzsch (2006) sind Projekte eine von mehreren Netzwerkformen. Mehrere Studien weisen darauf hin, dass immer mehr Mitarbeitende außerhalb von Linienaufgaben in Projekten arbeiten, und Scrum ist eine projektorientierte Arbeitsform, die diesen Trend verstärkt (vgl. Baudach 2018: 165). Castells (2001) sieht Projekte zentral für die Network Society an. Im Buch »Der neue Geist des Kapitalismus« haben Boltanski/Chiapello die Managementliteratur untersucht und herausgearbeitet, dass Projekte über die Jahrzehnte eine immer größere Rolle spielen. Sie beschreiben bereits, was sich bei der Einführung von SAP unter 6.3 zeigt: dass bei Projekten in Firmen »dauerhafte Verbindungen aufgebaut werden, die anschließend in den Hintergrund treten, aber weiter verfügbar bleiben« (Boltanski/Chiapello 2003: 149). Eine Studie zur Energiewirtschaft stellt fest, dass in den untersuchten Organisationen projektorientierte und vernetzte, temporäre Zusammenarbeit (Netzwerkorientierung) und Orientierung an der Kundschaft und Wettbewerbsfähigkeit (Marktorientierung) wichtiger wird und die Orientierung an Industrie (u.a. technische Effizienz) und Staatsbürgerlichkeit (u.a. Kollektivinteresse) abnimmt (vgl. Jacobsen/Blazejewski/Graf 2017).

6.2.3. Technisch: digitale Netzwerke

Neben diesen theoretischen und organisatorischen Gründen verbinden einige Autoren mit dem Netzwerkbegriff eine zentrale Stellung der IT in der Arbeitswelt. Die Einleitung des Kapitels hat bereits Kruse als Beispiel dafür genannt, der bei IT-Arbeit

8 Projektarbeit ist in der modernen Arbeitswelt weit verbreitet. Wie weit, dazu konnte der Verfasser leider keine genaueren Statistiken zu durchgeführten Projekten in Firmen oder Branchen finden. Anfragen an privatwirtschaftliche Firmen, die Zertifikate wie IPMA, PRINCE2 oder PMP anbieten, blieben erfolglos. Allerdings finden sich online Informationen: In Deutschland haben 16.982 Personen ein gültiges Zertifikat Project Management Professional (PMP) (Quelle: https://www.pmi.org/certifications/certification-resources/registry, Stand: 04.01.2024).Ein Zertifikat der IPMA (International Project Management Association) der Kategorien A bis D haben von 1995 bis 2020 jährlich im Schnitt 14.456 Personen weltweit gemacht (Quelle:https://www.vzpm.ch/fileadmin/dokumente/downloads/Deutsch/IPMA_Yearbook_2020.pdf, heruntergeladen am 04.01.2024).

von Netzwerkarbeit spricht. Castells hat bereits 1996[9] in seinem Buch eine globale Netzwerkgesellschaft beschrieben, die auf Informationstechnologien beruht. Interorganisationale IT-Systeme vereinfachen ab den 1990er Jahren die Kommunikation zwischen den Organisationen, zentralisieren Kontrolle, machen Transparenz, Wissensaustausch und Entscheidungsdelegation (Dezentralisierung) technisch umsetzbar (vgl. Bjørn-Andersen/Raymond 2014: 190). IT macht das Auslagern bestimmter Tätigkeiten möglich (vgl. Grimshaw et al. 2002: 188, Robertson/Verona 2006: 90, Sahaym/Steensma/Schilling 2007, Zammuto et al. 2007: 754ff.). Longen sieht einen Zusammenhang zwischen Firmen-Netzwerken der Call-Center-Arbeit und der verwendeten IT (Longen 2015). Auch beim Ansatz der systemischen Rationalisierung aus der Industrie- und Arbeitssoziologie spielt die IT eine zentrale Rolle in Organisationsnetzwerken. Sie sorgt für die »Optimierung des gesamtbetrieblichen Prozesses, die Organisierung von Markt- und Austauschprozessen und die Gestaltung der Beziehungen zwischen Betrieben und Unternehmen« (Sauer 2017: 286). Manche Autoren sehen gar Firmen als überflüssig an, weil potenziell jeder dank IT individuellen Zugang zu allen relevanten Informationen hat und jeder mit jedem kommunizieren kann (vgl. Symon 2000: 393).

6.3. Ein Beispiel für soziotechnische Netzwerkarbeit: IT-Projekte in Matrixorganisationen

Den Weg zu einem allgemeinen Konzept der soziotechnischen Netzwerkarbeit, das zu sämtlichen Fallstudien passt, weist IT-Projektearbeit. IT-Projekte sind auch deshalb relevant für die Untersuchung, weil sie Firmen bei der ERP-Einführungen einsetzen und es mehrere Forschungsarbeiten dazu gibt. Solche Projekte und die aus ihnen resultierenden Gestaltungsnetzwerke nach ERP-Einführung weisen die vier Ebenen soziotechnischer Netzwerkarbeit auf.

Allgemein zeigen sich in Projekten die vier Ebenen wie folgt:

1. **Ablauf – zwischen starker und schwacher Formalisierung**: Projekte schwanken zwischen zu starker und zu schwacher Formalisierung (vgl. Heidling 2018: 224). Durch Handbücher für Aufbau- und Ablauforganisation eines Projektes, entsprechendes Management und entsprechende Kennzahlen kann eine Formalisierung stattfinden (vgl. Kalkowski und Mickler 2005: 59). Trotzdem lassen sie für einzelne Beschäftigte Spielräume, um selbstständig fachliche Entscheidungen zu treffen und sich bspw. mit Kund:innen abstimmen zu können (vgl. Kalkowski/Mickler 2005: 59).
2. **Beziehungen – dank Projekt zwischen Organisationseinheiten und Einzelpersonen**

    - **Organisationale** Beziehungen – Abteilungsgrenzen und Hierarchien überbrücken: Projekte überbrücken Grenzen aus Fachabteilung, Teams und Hierarchien in Organisationen, damit unterschiedliche »Spezialdisziplinen, Domänen, Wissensbereiche, Einzelpersonen« (Heidling 2018: 211) zusammenkommen können.

9 Auf Deutsch 2001.

Sie ermöglichen einen verbesserten Informationsfluss und Spezialist:innen können bspw. durch Mitarbeit an mehreren Projekten ausgelastet werden (vgl. Ford/Randolph 1992: 273). Die Arbeitsteilung hat in der Matrixorganisation mithin zwei Ebenen: die Arbeitsteilung zwischen den Fachabteilungen und ihren Hierarchien (Ablauf- und Aufbauorganisation) und die Arbeitsteilung im Projekt selbst.

- **Interpersonelle** Beziehungen – auszuhandeln zwischen Führungskräften und Mitarbeitenden (horizontal und vertikal): Es gibt eine geteilte Autorität zwischen der Hierarchie der Organisation, in der das Projekt stattfindet, und dem Projekt selbst (vgl. Ford/Randolph 1992: 271). Es gibt Konflikte um Ressourcen, um Prioritäten, es gibt Rollenkonflikte zwischen Projekt-Rolle und Rolle in der Abteilung/im Team (vgl. Ford/Randolph 1992: 275). Die Projektleitung hat oftmals keine Weisungsbefugnis, diese liegt vielmehr bei der Team- oder Abteilungsleitung. Bei unternehmensübergreifenden Projekten findet ein Interessenausgleich und -abgleich durch Mitarbeitende und Projektleitung statt und ein großer Teil der Projektsteuerung erfolgt situativ auf operativer Ebene (vgl. Heidling 2018: 226). Kompromissfähigkeit, Verhandlungskompetenz, belastbare Vertrauensbeziehungen und personengebundener Austausch sind gefragt (vgl. ebd.: 227).

3. **Software**: Bei IT-Projekten sprechen Autoren von »technisch-bürokratischer Umklammerung der Projektarbeit« (Kalkowski/Mickler 2005) durch verschiedene Software-Werkzeuge wie Projektmanagement-Tools.
4. **Wissensarbeitende – Erwartung, Handlungsorientierungen zu kombinieren**: In Projekten ist Handeln stärker auf Kooperation und Kommunikation ausgelegt, bei gleichzeitiger ökonomischer und unternehmerischer Orientierung (vgl. Heidling 2018: 213f.). Es verschränken sich planmäßig rationales Handeln mit erfahrungsgeleitet-subjektivem (vgl. ebd. 222).

Konkreter zeigen sich die vier Ebenen der Softwaregestaltung bei Projekten der Einführung von ERP-Standardsoftware. Dort passen Organisationen im Zuge der Implementierung den Standard an, nehmen Einstellungen vor oder erweitern ihn. Es findet Softwaregestaltung statt. Es gibt einige Forschungsarbeiten, die IT-Projekte zur Einführung von ERP-Software untersuchen (bspw. Light/Wagner 2006, Svejvig/Jensen 2013, Conrad 2017). Auch sonst spielen IT-Projekte in der IT-Arbeit eine Rolle: so beim Thema Internet of Things (IoT) (vgl. Ziegler 2020: 225, 259) oder bei Firmenkooperationen (vgl. Kalkowski/Mickler 2015: 221ff.).

Hohlmann geht tiefergehend darauf ein, wie die Softwaregestaltung des ERP-Standards von SAP über das Implementierungsprojekt hinaus organisiert ist:

1. **Ablauf**: Zum einen bestimmt das IT-Projekt zur Einführung der ERP-Software selbst den Ablauf. Zum anderen sind die während und danach existierenden Gestaltungsnetzwerke die Basis des Ablaufs, um abteilungsübergreifend zusammenzuarbeiten und die Standardsoftware anzupassen, einzustellen, zu erweitern (vgl. Hohlmann 2007: 353).

2. **Beziehungen** jenseits bestehender Hierarchien und abteilungsübergreifend: Die Gestaltungsnetzwerke entstehen aus dem Projektteam der SAP-Einführung innerhalb der Firmen und untergraben alte Hierarchien. Es entstehen neue interdisziplinäre Netzwerke, wobei für deren Macht Wissen und Expertise entscheidend sind (vgl. ebd. 359f.). Die interdisziplinären Wissensbestände betreffen die anwendende Organisation, die fachlichen Prozesse und die ERP-Software selbst. Erst die drei Wissensbereiche zusammengenommen[10] ermöglichen eine Anpassung der Software (vgl. ebd.: 55).
3. **Software**: Insgesamt sieht Hohlmann eine enge Kopplung (ebd. 34f.) von Software und Organisation. Sie spricht von einem eng gekoppelten soziotechnischen Gesamtsystem (ebd. 368), weil nicht nur das ERP-System für die Arbeit in den Organisationen zentral wird, sondern weil IT- und Fachwissen nun Hand in Hand gehen und die anwendenden Organisationen neue, softwarebezogene Rollen wie Key User:innen etablieren müssen (s.u.).
4. **Wissensarbeitende**: Die neue Qualifikation, zwischen System und Fachabteilung zu vermitteln, verkörpert exemplarisch die Rolle Key User:in, die sich dadurch von anderen Anwendenden abhebt (vgl. ebd. 348). Sie wendet eine Software nicht nur an. Sie nimmt darüber hinaus Einstellungen an ihr vor, gibt neue Anforderungen für die Software auf oder arbeitet direkt mit Programmierenden zusammen. Gleichzeitig kennt sie sich fachlich in Bezug auf einen konkreten Anwendungsbereich gut aus und arbeitet eng mit den Anwendenden zusammen. Key User:innen agieren als Übersetzer:innen, und indem sie an der Softwaregestaltung mitwirken, halten sie die Software auch bei veränderten Anforderungen durch die Umwelt der Organisation oder der Anwendenden einsatzfähig und die Organisation effizient (vgl. ebd. 340, 357, 370).

Letztendlich zeigt sich, dass Organisationen mit einem ERP-System nicht nur ein neues Werkzeug in Betrieb nehmen. Langfristig ändert sich ihre interne Organisation, um diese Software zu gestalten. Damit beschreibt Hohlmann sowohl die Etablierung (via Projekt) als auch die Folgen von Softwaregestaltung: und zwar nicht nur für die Anwendenden, sondern auch jene Folge, dass in den anwendenden Organisationen neue Rollen, Gestaltungsnetzwerke und Wissensbestände entstehen.

Um diese vier Ebenen allgemeiner zu fassen, erarbeitet der nächste Abschnitt eine konzeptionelle Beschreibung des Netzwerkes unabhängig von Matrixorganisation und Projektarbeit, um so den unterschiedlichen Konstellationen zu entsprechen, in denen die Softwaregestaltung in den Fallstudien stattfindet.

6.4. Soziotechnische Netzwerkarbeit: die Ebenen Beziehungen, Software und Wissensarbeitende

Die Matrixorganisation und IT-Projekte sind lediglich eine unter mehreren Möglichkeiten, Softwaregestaltung zu organisieren. Für einen allgemeinen konzeptionellen

10 Sie nennt die Kombination Integrationswissen.

Rahmen, der für alle Fallstudien gleichermaßen gilt, schlägt dieses Kapitel das Konzept der soziotechnischen Netzwerkarbeit vor. Dafür arbeitet der Abschnitt im Folgenden anhand von Forschungsergebnissen für die Netzwerkebenen Beziehung, Software und Softwaregestaltende heraus, wie sie zur Transformation der Arbeitskraft durch die Vermittlung von Handlung und Struktur und damit zur Kontrolle von Arbeit beitragen. Abschließend folgt eine Analyse einer spezifischen Eigenschaft von Netzwerkarbeit: dass die unterschiedlichen Ebenen flexibel und je Kontext unterschiedlich zur Transformation der Arbeitskraft beitragen.

Für die konzeptionelle Ausarbeitung der Ebene Ablauf konnten keine nützlichen Bezüge in der Forschung gefunden werden.

6.4.1. Organisationale und interpersonelle Beziehungen

Industriespezifische Softwaregestaltung findet zwischen Einzelpersonen, innerhalb oder zwischen Organisationseinheiten statt. Für die soziotechnische Netzwerkarbeit sind diese interpersonellen und organisationalen Beziehungen zentral. Einerseits setzt die soziotechnischen Netzwerkarbeit voraus, dass ein bestimmter Grad an Kooperation in den interpersonalen und organisationalen Beziehungen vorhanden ist. Andererseits ist es Teil der soziotechnischen Netzwerkarbeit, kooperative Beziehungen zu erhalten. Wann organisations- und abteilungsübergreifend kooperative Zusammenarbeit gelingt, was Voraussetzungen und Hindernisse sind, zeigen Forschungsarbeiten zur kooperativen Zusammenarbeit anhand von drei Punkten[11]:

1. Wenn IT-Abteilungen (ob in- oder outgesourced) mit den Fachabteilungen kooperativ zusammenarbeiten (6.4.1.1).
2. Wenn trotz unterschiedlicher Interessen und Wissen zwischen IT-DL, IT-Abteilung und anwendenden Organisationen Strukturen zur kooperativen Zusammenarbeit bestehen (6.4.1.2).
3. Wenn über formale Netzwerkstrukturen wie Projekte hinaus interpersonale Beziehungen bestehen, die auf Vertrauen, Reziprozität und Kooperation basieren, und zwar sowohl für gemeinsame strategische Entscheidungen als auch für die operative Zusammenarbeit (6.4.1.3).

6.4.1.1. Arbeitsteilung zwischen Anwendung und Entwicklung – die IT-Abteilung

Wenn IT-Projekte in Unternehmen stattfinden, ist entweder eine IT-Abteilung oder ein IT-DL involviert. In den Fallstudien des Empirie-Kapitels ist das immer der Fall. Die Forschung weist darauf hin, dass eine IT-Abteilung nicht automatisch kooperativ mit

11 Phänomene, die eine Besonderheit der Wertschöpfungskette der Softwareentwicklung darstellen, aber auf die der in der Empirie untersuchten Softwaregestaltungsprozess keine direkte Auswirkung hatte, lässt die Untersuchung außen vor: so wie z.B. den Lock-in-Effekt durch die Implementierung einer Standardsoftware, d.h. eine starke Abhängigkeit vom softwareanbietenden Unternehmen (vgl. Hohlmann 2007: 334), aber auch den Lebenszyklus einer Software, der eine Abhängigkeit der Anwenderorganisation von der weiteren Wartung, von Updates und der Weiterentwicklung durch die Softwarefirma erzeugt.

den Fachbereichen zusammenarbeitet. Es gibt unterschiedliche Organisationsformen der IT-Abteilung – mal weniger und mal mehr hierarchisch oder marktförmig.

Das Forschungsgebiet, das die Beziehung zwischen IT- und Fachbereich untersucht, nennt sich IT-Alignment. Dabei geht es nicht nur um die Entwicklung einer einzelnen Lösung, sondern um die Organisation einer ganzen IT-Landschaft, d.h. eine Vielzahl an Softwarepaketen und Hardware. Durch das IT-Alignment soll verhindert werden, dass IT-Lösungen »not effectively support organizational activities and are more prone to miss business innovation opportunities provided by new information technologies« (Valorinta 2011: 47) – ob intern oder ausgelagert. Auch in der Forschung zum IT-Alignment sind es nicht Märkte oder Hierarchien, welche die interdisziplinäre Zusammenarbeit zwischen IT und Fachbereichen dominieren sollten. Vielmehr sind für eine kooperative Zusammenarbeit ein regelmäßiger Austausch zwischen IT und Fachbereichen, Co-Lokation, gemeinsame Planung, gemeinsame Projekte und Bildung von sozialem Kapital hilfreich (vgl. Reich/Benbasat 2000, Chan/Reich 2007, Masak 2006 Schlosser et al. 2015).[12] Dazu gehört, dass Führungskräfte wie Chief Information Officer (CIO) sowohl an Praktiken der eigenen IT-Einheit als auch an jenen der Fachbereiche teilnehmen (vgl. Valorinta 2011: 50). IT-Projektmanagende sind ein interdisziplinäres Bindeglied und es ist wichtig für den Erfolg von Projekten, dass sie über IT- und Fachwissen verfügen. Es ist vom hybriden IT-Projektmanagement die Rede:

> »[A] hybrid IT PM whose expertise includes technologies and techniques used on the project and who also possesses relevant knowledge of organizational operations, in-depth knowledge of the functioning of user departments, or expertise in the specific application area of the system« (Ko/Kirsch 2017: 318).

Untersuchungen zeigen Fälle, in denen IT-Abteilungen die Kooperation erschweren oder nicht auf Kooperation ausgerichtet sind. Guillemette & Parè (2012) machen fünf verschiedene Typen aus, die unterschiedlich eng mit den Fachabteilung zusammenarbeiten. Statt auf langfristige Beziehungen zu setzen, die förderlich für den Wissensaustausch wären, entscheiden manche Firmen, ihre IT marktförmig zu organisieren. IT-Sourcing-Abteilungen prüfen dann stetig, was Firmen extern an IT-Arbeit einkaufen und was sie selber machen. Das betreiben die Firmen so intensiv, dass gar die Rede von der »Industrialisierung des IT Sourcings« (von Jouanne-Diedrich et al. 2005) ist. Dabei gibt es Forschung, die zeigt, dass, wenn Firmen komplexe IT-Wissensarbeit auslagern, sie dann in die interne IT-Organisation investieren müssen, damit die Zusammenarbeit funktionieren kann (vgl. Tiwana/Kim 2016). Erschweren können die Kooperation auch zentralisierte IT-Abteilungen, die zentral Kooperationen steuern wollen und so dezentralen Initiativen in Organisationen entgegenstehen. Alternativen dazu sind dezentralisiert oder föderal organisierte IT-Abteilungen (vgl. Sesay/Ramirez 2016). Laut Masak ist die IT in Firmen meist föderal organisiert (vgl. Masak 2006: 209) und somit

12 Eine Studie hat herausgefunden, dass es eine Kluft zwischen »talk« und »action« bei IT-Projekten gibt, was die Kommunikationsprozesse zwischen IT und anderen Abteilungen anbelangt. Auch wenn sie von der Projektmethode vorgesehen waren und Projektleitende sich öffentlich hinter die vorgesehenen Kommunikationsprozesse stellen, wurden sie nicht eingehalten (vgl. Monteiro de Carvalho 2013).

von einer Zentrale unabhängiges Arbeiten Teil ihrer Organisation. Wie auch immer die IT-Abteilung organisiert ist: Die Softwaregestaltung muss mit den verschiedenen Typen von IT-Organisationen zurechtkommen. Hürden durch die hierarchische oder marktförmige Organisation gilt es zu überwinden, soll die Kontrolle der Softwaregestaltung gelingen.

6.4.1.2. Übergreifende Zusammenarbeit trotz unterschiedlicher Interessen

Letztendlich hängt die kooperative Zusammenarbeit davon ab, inwiefern Fach- und IT-Abteilungen, EVU und IT-DL bzw. Softwarefirmen unterschiedliche Interessen überwinden. Innerhalb von Konzernen gibt es zumindest Hierarchien, die den Abteilungen übergeordnet sind und die Zusammenarbeit zwischen diesen steuern können (z.B. ein Lenkungskreis aus Projektleitung und Abteilungsleitung bei einem IT-Projekt). Zwischen Organisationen ist es schwieriger, weil die Organisationen nicht immer auf Augenhöhe agieren, wie die im Folgenden zitierte Forschung zeigt.

Es gibt ein umfangreiches Forschungsfeld zu den unterschiedlichen Steuerungsformen in Organisationsnetzwerken und entsprechende unterschiedliche Begriffe wie Netzwerk-Governance, Meta-Organisation oder Netzwerk-Orchestrierung (vgl. Helfen/Wirth 2020: 14ff.). Wichtig für die Untersuchung hier ist, dass die Forschung feststellt, dass sich Steuerungsstrukturen im Zeitverlauf verändern und ein Ergebnis dynamischer Lernprozesse sein können. Dafür sind allerdings Kapazitäten für Wissensaustausch und Reflexion zwischen Firmen notwendig, damit die Organisationen die jeweils passenden organisationsübergreifenden Steuerungsstrukturen finden können (vgl. Mola et al. 2017: 1293, van Fenema/Keers/Zijm 2014: 205). Solche Lernprozesse zeigen einzelne Fallstudien des Empirie-Kapitels.

Dabei muss für eine längerfristige Kooperation deren Steuerung mit Konflikten zurechtkommen, die aus unterschiedlichen Interessen und ungleichen Wissensbeständen entstehen[13]. Kaniadakis arbeitet heraus, wie Machtungleichgewichte entstehen, wenn die anwendende Organisation die Anpassungen einer Software einer darauf spezialisierten Firma überlässt. Die Unternehmen müssen dann Wege finden, die externen kooperierenden Organisationen zu kontrollieren und sich an der Umsetzung zu beteiligen, auch wenn sie über weniger Wissen verfügen (vgl. Kaniadakis 2012: 270). Ungleichgewichte bei der Auslagerung von IT-Arbeit entstehen u.a. dadurch, weil es bei der auslagernden Organisation intern zu Kompetenzverlust, Kontrollverlust, einem Abbau des organisationalen Lernens oder der Innovationskapazitäten kommt (vgl. Miozzo/Grimshaw 2005: 1424). In der Untersuchung von Peled (2001) gelingt die Kontrolle der Expert:innen gar nicht mehr und es entsteht ein »consultant-centered-regime«, weil die öffentliche Verwaltung im IT-Bereich Kompetenzen ausgelagert hat und damit Wissen und Kontrolle verliert. Sie ist von anderen abhängig, um ihre Leistung zu erbringen: »Government authority is increasingly being shared with its proxies« (Peled 2001: 509). Auch Flecker/Holtgrewe ziehen den Schluss, dass es bei Auslagerung von IT-Wissen »langfristig zu einer Verschiebung der Machtbeziehungen zugunsten der privaten Dienstleister« (Flecker/Holtgrewe 2008: 314) kommt. Zwischen IT-Dienstleistenden und

13 In der Fallstudie KOOP2 des Empirie-Kapitels zeigt sich, was passiert, wenn das nicht gelingt.

den Auftraggebenden bzw. der Konzernzentrale kann es zu Machtkämpfen kommen (vgl. Mezihorak 2018: 825ff.).

Machtkämpfe aufgrund von unterschiedlichem Wissen und Spezialisierungen existieren auch innerhalb von Firmen. Es ist nicht ausgeschlossen, dass Beschäftigte die IT nutzen, um gegen die Organisation zu arbeiten (vgl. Symon 2000: 400ff.). Silva hat allgemein für IT-Systeme beschrieben, dass die interne IT-Abteilung Machtquellen wie spezifische IT-Ressourcen sowie Wissen besitzt und Unsicherheit erzeugen kann, die nur sie selbst aus der Welt zu schaffen fähig sind. Zum Beispiel kreieren schlecht dokumentierte Systeme Abhängigkeit (vgl. Silva 2005: 56). Ganz allgemein ist die Einführung und Entwicklung von Software ein Schauplatz für Mikropolitik (vgl. Ortmann et al. 1990).

Wie der nächste Punkt zeigt, spielen für eine kooperative Zusammenarbeit nicht nur Beziehungen auf der Meso-, sondern auch auf der Mikro-Ebene eine Rolle.

6.4.1.3. Netzwerkstrukturen reichen nicht: Vertrauen, Reziprozität und Kooperationsbereitschaft

Was bei Netzwerken wie Projekten allgemein und der Softwaregestaltung im Besonderen eine große Rolle spielt, sind die direkten Kontakte zwischen den Beschäftigten über ihre jeweiligen Organisationen, Abteilungen oder Teams hinweg. Die hier in der Folge zitierten Forschungsarbeiten zeigen, dass rein formale Abläufe oder formal definierte Stellen für kooperative Zusammenarbeit nicht ausreichen. Gleichzeitig zeigen Untersuchungen, dass kooperative Beziehungen schnell wieder verloren gehen können, wenn die Praxis der Kooperation, des Vertrauens und der Reziprozität nicht mehr besteht. Ihre Erwartungen müssen die Beschäftigten zwischen Organisationen oder Organisationseinheiten wie Teams oder Abteilungen immer wieder abgleichen – bei strategischen Fragen und für die operative Zusammenarbeit.

Kooperative Beziehungen zwischen Beschäftigten, die Teil unterschiedlicher Organisationseinheiten sind, stellen sich nicht automatisch ein, nur weil beispielsweise Netzwerkstrukturen wie Projekte vorhanden sind. Eine Studie kommt zu dem Schluss, dass Projektnetzwerke, die mit hierarchischen Organisationsformen koexistieren müssen, nur eine beschränkte Dynamik für kooperatives Handeln entwickeln (vgl. Rüegg-Stürm/Young 2001). Diese entsteht erst durch individuelle Kooperationsbereitschaft, Sozialkompetenz, breit verankerte Verantwortungsbereitschaft und verändertes Führungsselbstverständnis/-verhalten. Dezentrale, teamorientierte Arbeitsformen brauchen Zeit, vertikale und horizontale Kommunikationsprozesse entstehen erst zögerlich. Am schwierigsten ist das partnerschaftliche und verbindliche Zusammenarbeiten (vgl. ebd. 198).

Dass Zeit in Beziehungen eine Rolle spielt, hat bereits Powell festgestellt: Erst mit der Zeit entsteht aus reziproken Beziehungen und geteilten Interessen Vertrauen (vgl. Powell 1990: 305). Für Adler und Heckscher basiert Vertrauen auf »the degree to which members of the community believe that others have contributions to make towards this shared creation« (Adler/Heckscher 2006: 21) und damit auf Annahmen über die Zukunft. Mit der Zeit wissen beispielsweise Zuliefererorganisationen, was die belieferten Unternehmen wollen, und es sind keine Absprachen mehr notwendig (vgl. Uzzi 1997. 46). Es gibt Verhandlungsroutinen, man passt sich gegenseitig an und macht mehr, als im Vertrag steht (vgl. ebd. 47).

Ein anderer Weg zu kooperativen Beziehungen im Arbeitsalltag ist jener, den Bolte und Porschen mit der »Organisation des Informellen« beschreiben. Job Rotation, Hospitation, Promotor:innen oder Trainee-Programme für Einsteiger:innen stiften informelle Beziehungen in Organisationen (vgl. Bolte/Porschen 2007). Dadurch entstehen Netzwerke innerhalb von Firmen, ohne dass eine größere Reorganisation notwendig ist.

Gerade bei der Einführung oder der Gestaltung von Software ist besonders viel Vertrauen notwendig. Beschäftigte werden IT-Berater:innen mit Misstrauen begegnen, wenn sie um ihre gewohnte Arbeitsweise fürchten müssen. Konflikte bei Änderungen an IT-Systemen sind etwas Besonderes. Verändert sich die Arbeit der Betroffenen, kann das einen Angriff auf deren Fachwissen darstellen (vgl. Boonstra 2006, Boonstra/de Vries 2015).

Doch auch wenn vertrauensvolle und kooperative Beziehungen wichtig sind, stellen sie sich nicht immer ein. Die Forschung stellt fest, dass in Kooperationsnetzwerken generell ein geringes Vertrauen und eine geringe Loyalität besteht (vgl. Grimshaw et al. 2002: 200, vgl. Brinkmann/Dörre 2006: 139, Howcroft/Richardson 2012: 122, vgl. Holtgrewe 2014: 17ff.). Bei einem Digitalfunkkonsortium mehrerer Firmen war das Problem, dass diese selbst explizites Wissen nicht geteilt haben, weil man Angst hatte, Schlüsseltechnologie zu verlieren, und Ingenieur:innen ihrer jeweiligen Firma loyaler als dem Verbund waren (vgl. Hirschfeld 2000: 268 und 277). Insbesondere marktförmige Beziehungen zwischen Anwendung und Entwicklung behindern die Zusammenarbeit, unterminieren Vertrauen und machen so den notwendigen Wissensaustausch schwer (vgl. Felin/Zenger/Tomsik 2009: 557).

6.4.2. Software kontrolliert und strukturiert das Netzwerk

Der Arbeitsprozess der Softwaregestaltung basiert in zweifacher Hinsicht auf Software: Sie ist sowohl Arbeitsmittel als auch Arbeitszweck. Die Softwaregestaltung ist Teil des Produktionsprozesses der Softwareentwicklung, der digitale Bestandteile herstellt und montiert.[14] Bei der soziotechnischen Netzwerkarbeit spielt Software eine besondere Rolle, wie es dieser Abschnitt ausgehend von unterschiedlichen Forschungsbezügen zeigt:

1. Die Kontrolle durch Software beschränkt sich nicht auf Standardisierung, Formalisierung und Überwachung. Vielmehr ist bei der Softwaregestaltung ihre koordinierende, kooperationsermöglichende und wissensaktivierende Funktion wichtig[15] (6.4.2.1).
2. Die Softwarearchitektur beeinflusst die Arbeitsteilung und den Arbeitsprozess der Softwaregestaltung (6.4.2.2).

14 Was die Untersuchung nicht näher betrachtet, ist die Rolle der IT als Infrastruktur: Breitband- und firmeninterne IT-Netzwerke sind zwar Basis der soziotechnischen Netzwerkarbeit. Sie sind aber in allen Fällen gleich, machen keinen Unterschied bei der Transformation der Arbeitskraft und sind keine Besonderheit der Softwaregestaltung.

15 Was diese Arbeit unter Kontrolle versteht, wurde bereits in der Einleitung und unter 6.1 als in der Arbeitssoziologie gängiges Transformationsproblem beschrieben.

Software ermöglicht es, dass Spielräume für Subjektivität und Autonomie wie auch eingeschränkte Handlungsspielräume gleichzeitig existieren. Sie richtet den Beitrag der Einzelpersonen auf den Organisationszweck aus und hilft, ihre Arbeitskraft zu transformieren. Das bewirkt sie selbst dann, wenn die Arbeit der Beschäftigten nicht durchgeplant ist. Die softwaretechnische Arbeitsumgebung ermöglicht eigenständiges Arbeiten mit nur wenig Kontrollaufwand für das Management.

Natürlich setzen Organisationen Software auch als Werkzeug zur Kommunikation ein (E-Mails, Chatprogramme etc.). Das ist jedoch kein gesondertes Thema der Untersuchung. Der Abschnitt über die soziotechnische Flexibilität, in der die vier Netzwerkebenen zur Kontrolle beitragen, greift das Thema aber auf (6.4.4).

6.4.2.1. Software kontrolliert die Softwaregestaltung: einschränkend und ermöglichend

Software ist heutzutage zwar in vielen Organisationen so zentral für die Prozessgestaltung wie Fließbänder in manchen Fabriken. Allerdings ist ihre Form der Kontrolle bei der soziotechnischen Netzwerkarbeit nicht identisch mit diesen, weil sie sowohl Autonomie als auch Fremdbestimmung ermöglicht. Die soziotechnische Netzwerkarbeit zeichnet aus, dass bei ihr ein und dieselbe Software zugleich a) Wissens- und Lernprozesse in Gang setzen, b) den Arbeitsablauf vorgeben und c) die Arbeit zwischen den Beteiligten koordinieren helfen kann. Dabei verteilen sich die aufgezählten Eigenschaften von Software (a, b, c) im Arbeitsprozess der Softwaregestaltung auf verschiedene Softwarelösungen (wie Ticketsystem, ERP-System, Projektmanagementlösungen, Chatprogramme, E-Mail-Programme etc.) oder Teile einer Softwarelösung. Viele Werkzeuge der Softwaregestaltung sind softwaretechnisch abgebildet – ob Projektpläne in Excel, Anforderungen im Ticketsystem oder Dokumentationen im MS Sharepoint.

Ermöglichen und Koordinieren: Software als zentrales Organisationsobjekt

Bei der Softwaregestaltung trägt Software vor allem zur Kontrolle bei, indem sie die kooperative Wissens- und Kommunikationsarbeit ermöglicht und weniger, indem sie diese einschränkt. Im Folgenden verdeutlichen einige Forschungsarbeiten zu IT-Arbeit, was damit gemeint ist, dass Software nicht nur kontrolliert durch Überwachen, Standardisieren und Formalisieren.

Darr bezieht sich auf Rennstam (2012), wenn er in seiner Untersuchung herausfindet, dass Software auch »interactive relationships between itself and knowledge workers such as development engineers« (Darr 2019: 892) fördert. Software entlockt direkt in der Interaktion zwischen Software-Objekt und Wissensarbeitenden das Wissen und regt Letztere an, sich zu engagieren. Diese Beziehung zwischen Software und den Wissensarbeitenden ist zentral für die Kontrolle – und weniger die Beziehung zwischen Management und Arbeitenden (vgl. Rennstam 2012: 1084f.). Rennstam bezeichnet das als Objekt-Kontrolle, die erklären kann, wie Software Arbeit kontrolliert, die nicht so einfach bürokratisch oder hierarchisch kontrolliert werden kann oder muss:

> »One main feature of object-control is precisely that it interrupts the formal hierarchy, allows for rearrangement on the basis of knowledge relationships, and invites organizational members to make use of their knowledge. Instead of employing bureaucracy

> as a guide for action, or electronic systems for monitoring and correcting deviances, object control involves the creation of a temporary community whose trajectory is guided by various knowledge relationships with the object.« (Rennstam 2012: 1084)

Ein Beispiel aus der Wissenschaft wäre die Software zur qualitativen Analyse MAXQDA: Mehrere Wissenschaftler:innen können an einem Projekt arbeiten. In der Interaktion mit der Software pflegen sie Daten und werten sie aus. Das machen sie ohne die Intervention einer Führungskraft.

Für Darr (2019) müssen Arbeitende die Normen des Softwareobjekts nicht internalisieren. Es reicht, wenn sie durch die Interaktion mit dem Wissensobjekt dessen Normen in der Praxis verwirklichen. Indem man sich einlässt auf diese Objekte, kann man nicht anders, als deren Normen zu akzeptieren und zu verwirklichen. Hier kommt wieder, wie schon bei Rennstam, einer Eigenschaft dieser Form der Kontrolle eine besondere Bedeutung für die Wissensarbeit zu: Autonomie, zu der auch die Möglichkeit der Kritik gehört, ist Teil der Objektbeziehung und fördert das Lernen.

> »[W]orkers must retain a certain amount of agency, in the form of public critique and even occasional resistance.« (Darr 2019: 893)

Das Konzept der Objektkontrolle schließt sich damit dem Ansatz der Affordances an, der Software ebenfalls weder als etwas rein sozial Konstruiertes noch als etwas das Handeln Determinierendes betrachtet. Die »affordance perspective recognizes how the materiality of an object favors, shapes, or invites, and at the same time constrains, a set of specific uses« (Zammuto et al. 2007: 752). Für andere Autoren gilt dies selbst für ERP-Systeme, wo »technology's consequences for organizations are enacted in use rather than embedded in technical features« (Boudreau/Robey: 2005: 14).

Um zum obigen Beispiel zurückzukommen: Kritik und gemeinsame Diskussion der Beteiligten über die Funktionsweise von MAXQDA oder über die von ihnen gemachten Eingaben oder Analysen kann den Arbeitsfortschritt befördern. Dafür reicht es aus, wenn sie ohne Führungskraft mit der Software und untereinander interagieren. Dabei entscheidet sich, welche Funktionalitäten des Programms sie wie nutzen.

Auch andere Forschungsarbeiten zeigen die koordinierende Funktion von Software[16]. Sie ist das gemeinsame Bezugsobjekt der Beschäftigten. Das können verwendete Hilfsmittel oder Werkzeuge (wie Ticketsysteme) oder die entwickelte Software selbst sein. In Studien zur Softwareentwicklung vermitteln solche Software-Objekte, unterstützen den Wissensaustausch und zeichnen sich durch ihre Interpretationsdürftigkeit aus (vgl. Barrett/Oborn 2010: 1200ff.). Allgemein können Software-Objekte zwischen IT-Abteilungen und den Fachbereichen vermitteln, z.B. indem Mitarbeitende in Gesprächen auf sie Bezug nehmen. Für die Softwaregestaltung ist dabei besonders relevant, dass das speziell bei interdisziplinärer Arbeit passiert. Dort fungieren die Objekte als »translation and transformation devices across various thought worlds« […] »they make

16 Die zitierten Quellen schreiben vom Boundary-Object oder IT-Artefakt. Für die Argumentation sind die Unterschiede zwischen den Konzepten nicht relevant und so wird allgemein von Objekt-Kontrolle gesprochen.

cross disciplinary work possible« (Nicolini/Mengis/Swan 2012: 624). Eine andere Untersuchung zeigt, wie die entwickelte Software als Diskussionsgegenstand dient, auf den sich die Beteiligten beziehen (vgl. Carugati et al. 2018: 31). Auf ihr basieren wichtige Prozesse der Softwareentwicklung wie Einspruch, Widerspruch und Konsensbildung (vgl. ebd. 29). Eine weitere Untersuchung zeigt, wie sich in der fertigen Software dann die »negotiated, embedded and sedimented sets of rules« (Ponte/Rossi/Zamarian 2009: 319) der gemeinsamen Zusammenarbeit wiederfinden.

Wie Software die Zusammenarbeit anregt und koordiniert, zeigt sich besonders an der gemeinsamen Arbeit an (Quell-)Texten. Der Quellcode, die Kommentare und Dokumentationen in der Software machen die Koordination mit anderen möglich, ohne direkt mit diesen kommunizieren zu müssen[17]. Die Interaktion mit anderen basiert auf Spuren, die diese hinterlassen haben (vgl. Bolici/Howison/Crowston 2009). Dies funktioniert bspw. dann, wenn durch Lesen des Quellcodes oder einer Anforderung klar wird, was noch zu tun ist oder ob Voraussetzungen für die Umsetzung gegeben sind. Dies geschieht, ohne dass Programmierende explizit erzählen, was sie gemacht haben (vgl. Bolici/Howison/Crowston 2016: 19).

Einschränken und Überwachen: Software gibt Rollen, Abläufe und Eingabemöglichkeiten vor und liefert Kennzahlen

Trotz selbstständigen Arbeitens sollte nicht unterschlagen werden, dass Software die Handlungsspielräume einschränkt. Das passiert aber 1. nicht wie am Fließband, sondern 2. indem Software Vorgaben macht, soziale Strukturen in soziotechnische umwandelt und Beschäftigte und Abläufe überwacht.

Inwiefern unterscheidet sich 1. die technische Kontrolle bei der Softwareentwicklung von jener durch ein Fließband? Als klassisches Beispiel für technische Kontrolle gilt für Edwards (1981) das Ford-Montageband: Es legt auszuführende Arbeitsschritte, deren Abfolge und das Arbeitstempo fest und löst damit das erste Kontrollproblem von Arbeitsleitung und -anweisung (vgl. ebd.: 131). Durch zusätzliche Verwendung von IT kommt es zur Ausweitung der technischen Kontrolle durch »Feedback-Systeme« (z.B. wenn Arbeitende Rückmeldung bekommen, dass ein Arbeitsschritt abgeschlossen ist) (vgl. ebd. 136). Damit ist das zweite Kontrollproblem der Überwachung gelöst (vgl. ebd. 138). Aus Edwards Sicht wird das dritte Kontrollproblem (Disziplinierung und Belohnung) nicht durch die technische Kontrolle gelöst (vgl. ebd. 139).

Wie schon unter 5.2.3 diskutiert, gibt es keine Best Practice, was die Kontrolle von Softwareentwicklung anbelangt. Es gibt sehr unterschiedliche Arbeits- und Organisationsformen. Grundsätzlich geben in der Softwareentwicklung die Softwarewerkzeuge nur den Rahmen vor. Klar grenzen z.B. Andrews et al. (2005) die Arbeit in der Softwareentwicklung von Fließbandarbeit ab, weil weder eingesetzte Methoden zur durchgehenden Standardisierung beitragen noch die Abfolge der einzelnen Arbeitsschritte fest getaktet ist (63f.).

> »At every stage human rather than machine intervention predominates.« (ebd.: 66)

17 Die Autoren nennen das »stigmergic coordination«.

Anders als am Fließband gibt die Technik keinen festen Takt vor. Wenn sie es tut, dann durch soziale Organisationsformen wie Scrum, in denen es Sprints gibt, deren Zeiträumen durch Menschen festgelegt werden. Abfolge und Tempo sind dann aber nicht durch Technik vorgegeben, vor allem nicht, wie lange man für einen Arbeitsschritt (z. B. die Programmierung einer Methode) braucht. Die Menschen müssen priorisieren, Änderungen am Quellcode oder Tests freigeben. In zwei Fallstudien, die Barrett in ihrer Untersuchung erforscht hat, haben die Programmierenden ihre Zeit selbst in der Hand und nur die Deadline war direkt vom Management kontrolliert (vgl. Barrett 2005: 89).

Statt wie ein Fließband schränkt 2. Software die Arbeit der Softwaregestaltenden anders ein: durch a) Vorgaben machen, b) Abläufe integrieren, c) soziale Strukturen softwaretechnisch zu stützen und d) Arbeit zu überwachen.

Zu a): Software kontrolliert, indem sie vorgibt, wie ein Beitrag und welcher Beitrag zu leisten ist. Allgemein müssen Anwendende (auch von Werkzeugen für die Softwaregestaltung) die Programmlogik nachvollziehen und sich mit den von der Software erwarteten Verhaltensweisen auseinandersetzen (vgl. Degele 2000: 67f.). Sie müssen sich an die in Software fixierten Regeln halten (vgl. Heidenreich/Kirch/Mattes 2008: 7). Dabei kann Software zu einer erhöhten Rigidität der Formalisierung führen, indem sie Arbeitsschritte als alternativlos vorgibt, sie detaillierter und umfassender verregelt, vorstrukturiert und systematisiert (vgl. Schaeffer/Funken 2008: 13f.). So gibt sie z. B. über Eingabemasken vor, was Anwendende in ein Ticketsystem eingeben können, welche Mitarbeitende es bearbeiten und an welche Teams in der IT-Abteilung die Tickets weitergeleitet werden können.

Zu b): Ganz allgemein ermöglicht IT eine Prozessorientierung sowohl im Unternehmensnetzwerk als auch in einer vermarktlichten Organisation (vgl. Sauer 2018: 196). Der oder die Einzelne und sein Input in die Software sind in einen digitalen Prozess eingebunden. Das muss aber nicht mit einer durchgetakteten, formalisierten Anwendung einhergehen. Bei komplexen Aufgaben formalisieren Unternehmen Zielparameter und erwarten von den Anwendenden Problemlösungen und strukturierende Arbeit, mit der sie die Lücke im digitalen Prozess füllen (vgl. Kleemann/Matuschek 2008).

Neben der Vorgabe durch Software, wie Beschäftigte einen Beitrag und welchen Beitrag sie leisten können, macht IT (Hardware und Software) c) im Allgemeinen soziale Strukturen starrer (vgl. Mutch 2010). So fixieren die Benutzendenrollen in der Software die sozialen Rollen oder der Programmablauf von ERP-Software die Arbeitsroutinen softwaretechnisch. Das heißt, soziale Organisationselemente wie Rollen oder Routinen bekommen (software-)materielle Bestandteile (vgl. Volkoff et al. 2007: 840f.). Routinen und Rollen werden soziotechnisch: Sie existieren in der Software und der sozialen Organisation. Bei einer angepassten ERP-Standardsoftware für eine anwendende Organisation spricht Hohlmann von einer engen Kopplung (ebd. 34f.) von Software und Organisation und Brödner davon, dass diese wie »flüssiger Beton« (Brödner 2002 nach Remer 2008: 42) in der Organisation aushärtet.

Software gibt nicht nur vor, wie Beschäftigte einen Beitrag leisten können, oder verwandelt soziale Strukturen wie Rollen in soziotechnische. Sie d) überwacht auch Arbeit. Bereits Zuboff (1988) hat vom »electric panopticum« gesprochen: Kontrolle kraft Transparenz. Die IT gilt seit längerem als Grundlage für Gruppen, Cost- und Profitcenter etc. Sie ermöglicht die Dezentralisierung durch Planungs-, Budget- und Kennziffernsyste-

me und die Kontrolle durch Rückmeldung (vgl. Kocyba 1999: 102f.). IT-Systeme können für das Management in Bezug auf Kontrolle das Gefühl einer kleinen Firma bewahren, indem sie eine zentrale Datenbank für die firmenweite Auswertung zur Verfügung stellen (vgl. Sahaym/Steensma/Schilling 2007: 867). Das gilt u.a. für die Produktion, weil die IT deren zentrale Steuerung möglich macht (vgl. Silva 2005: 50), wie auch für einen spanischen Energieversorger (vgl. Tsamenyi/Cullen/González 2006). Über Kennzahlen können Organisationen Teams kontrollieren und selbstständiges Arbeiten ermöglichen. Sie können

> »zu einer orientierenden und steuernden Instanz auch im Prozess der Selbstorganisation der Teams werden. Gleichzeitig vermitteln sie – im Sinne der systemischen Integration des gesamten Unternehmens – zwischen der Selbstorganisation in den Teams und der Ebene des (hierarchischen) Managements« (Boes et al. 2018: 186).

Der Studie von Boes et al. (2018) kommt hier eine besondere Bedeutung zu, weil sie auch den Einsatz von Scrum berücksichtigt und zwei ihrer Fallstudien im Bereich Softwareentwicklung liegen. Die Überwachung durch Kennzahlen ermöglicht dabei die Selbstkontrolle des Teams und des Einzelnen.

6.4.2.2. Prägt die Organisation: die Softwarearchitektur

Software ist nicht nur wichtig für die Kontrolle der Softwaregestaltung. Sie prägt durch ihre Architektur die Organisationsstruktur und damit die Transformation der Arbeitskraft. Es gibt eine bereits jahrzehntealte Diskussion über das Verhältnis von Organisation und Softwarearchitektur. Am bekanntesten ist Conways Law. In dem Aufsatz aus dem Jahr 1968 vertritt Conway, der sich allgemein mit »system design« von Technik beschäftigt, die folgende These:

> »The basic thesis of this article is that organizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations.« (Conway 1968: 31)

Bezogen auf die Softwareentwicklung bedeutet das, dass man »in den Strukturen des Softwaresystems die Strukturen der jeweiligen Entwicklungsorganisation« (Masak 2006: 232) wiederfindet.

Neuere Forschung zeigt, dass Conways Law bei der Softwareentwicklung nicht immer zutrifft. Zudem ist dafür, wer mit wem intensiv kommuniziert, nicht nur die Aufteilung der Software in Module oder Bausteine relevant wie bei Conways Law. Es geht auch um Abhängigkeiten, die entstehen, wenn Firmen oder Abteilungen gemeinsam Standardbausteine bzw. -module entwickeln.

Inwiefern wirkt Conways Law bei Softwareentwicklung nicht immer? Unabhängig von Conway wird das Thema begrifflich unterschiedlich gefasst: das eine Mal als Modularisierung, ein anderes Mal als Spiegelung. Unabhängig davon, ob es sich um Software, Autos oder andere Maschinen handelt, stellt eine Metanalyse aus dem Jahr 2016 fest:

> »Over two-thirds (70 %) of the descriptive studies provide strong evidence of mirroring, 22 % provide partial support, while only 8 % do not support the hypothesis.« (Colfer/Baldwin 2016: 710)

Das heißt also, dass sich bei 70 % die technische Architektur in der Arbeitsteilung spiegelt. Das liegt daran, weil jene, die sich mit einem technischen Teil auskennen und Verantwortung dafür haben, sich irgendwie mit anderen absprechen müssen, um ihre Arbeit zu koordinieren (ebd. 712).

> »[O]rganizational ties will be dense within modules of the system where technical dependencies are dense and sparse across modules where technical dependencies by definition are sparse.« (ebd. 713)

Die Autoren fanden aber auch heraus, dass offene, kollaborative Projekte im IT-Bereich die geringste Unterstützung für die Spiegelungshypothese aufweisen (ebd. 726). Sie führen drei Gründe auf, die alle darauf zurückzuführen sind, dass Software digital und damit leichter via Vernetzung und entsprechender Entwicklungsumgebung zu organisieren ist.

Auch andere Untersuchungen stellen fest, dass es bei der Softwareentwicklung zur Spiegelung kommen kann, aber nicht muss. In einer Fallstudie zeigte sich am Anfang noch Conways Law: Eine Firma hatte Teile ihrer Softwareentwicklung ausgelagert. Weil die Kommunikation innerhalb der Teams besser war als zwischen den Teams, gab es je Team einen Quellcode, um den sich das Team gekümmert hat. So entwickelten sich entsprechend drei voneinander geschiedene Systeme (Insellösungen) heraus. Doch die Firma wollte dies unterbinden und es gelang ihr auch: Sie führte einheitliche agile Entwicklungsmethoden[18] ein und es gab tägliche Videokonferenzen. So überwand die Firma die Barrieren in der Kommunikation zwischen den Teams, was sich dann auch in der Softwarearchitektur niederschlug. Auch wenn die Kommunikation innerhalb der Teams weiterhin intensiver war, konnte die Firma die Folgen davon auf die Architektur (Insellösungen) durch aktive Intervention und Etablierung informeller Kommunikation unterbinden (vgl. Hvatum/Kelly 2005: 3f.). In einem anderen Fall (der auch nicht Teil der Colfer/Baldwin-Studie ist) gelang dies nicht und es entstanden Insellösungen (vgl. Swan/Scarbrough 2005: 932). Letztendlich zeigt dies, dass Software sich anders verhält als andere Technologien und bei ihr der Spielraum größer ist, zu intervenieren und die Spiegelung von Architektur und Kommunikationsstrukturen der Softwareentwicklung zu unterbinden. Zwei Autor:innen sprechen deshalb von Conways *force* (vgl. Hvatum/Kelly 2005), weil es weniger ein Gesetz als eine Kraft ist, welche das Verhältnis von Softwareentwicklung und -architektur in eine bestimmte Richtung tendieren lässt, es aber nicht determiniert.

Bei der Architektur spielt für die vorliegende Arbeit aber nicht nur die Aufteilung einer Software in unterschiedliche Bausteine bzw. Module und entsprechende Schnittstellen eine Rolle. Es geht auch darum, ob Organisationen etwas individuell oder als

18 In diesem Fall: Extreme Programming. Eine iterative Methode der Softwareentwicklung, bei der es wichtiger ist, eine Anforderung umzusetzen, als einem formalisierten Vorgehen zu folgen.

Standard gestalten. Gestalten mehrere Organisationen oder Organisationseinheiten wie Abteilungen gemeinsam einen Standardsoftwarebaustein, entstehen Abhängigkeiten. Wenn alle mitgestalten wollen, muss ein Austausch darüber stattfinden, was der Standard können soll.

In seiner Dissertation untersucht Remer die Umstellung einer Softwarearchitektur: Statt eines großen, monolithischen Systems will die Organisation viele kleine, über definierte Schnittstellen flexibel aufrufbare Bausteine programmieren[19]. Remer stellt fest, dass solche Architekturinnovationen Änderungen an eingespielten sozialen Beziehungen verlangen. So kann er zeigen, dass die Firma ein neues Gremium einrichten muss. Dies ist notwendig, weil Anforderungen an einen Baustein mit den unterschiedlichen Stakeholdern, die diesen Baustein nutzen, abzustimmen sind. Das ist eine Folge der Architektur: Die Fachbereiche der Firma können nicht mehr allein über ihre Software(teile) entscheiden (vgl. ebd. 145). Die Einführung einer auf flexible Bausteine setzenden Softwarearchitektur hat somit ihre Risiken, weil es zu komplexen organisationalen und technischen Abhängigkeiten kommen kann und es schwieriger wird, über diese den Überblick zu behalten (vgl. ebd. 168). Das heißt, hier spiegeln sich zwar einerseits die Kommunikationsstrukturen, weil bestimmte Teams für einen Baustein zuständig sind und sie programmieren. Andererseits gibt es auch keine Spiegelung, weil sich die Kommunikation für die Abstimmungen und um sich auf die Gestaltung eines (Standard-)Bausteins zu einigen, nicht in der Softwarearchitektur niederschlägt und z.B. deren modulare Aufteilung ändert

So stehen die Organisationen bei der Softwaregestaltung eines Standards vor der Wahl: Statt für Abstimmungen aufwendige Kommunikationsstrukturen zu etablieren, lassen sich Abhängigkeiten reduzieren und eine flexible Architektur in Form einer »Lego-Logik« (vgl. Schmierl/Pfeiffer 2005) ermöglichen, wenn sich alle Beteiligten einem Standard-»Lego«-Baustein und seinen Schnittstellen unterordnen und sämtliche Kommunikation und sämtliches Wissen für diesen innerhalb eines Teams existiert. Eine gewisse Flexibilität im Zusammenspiel von organisationsspezifischer Praxis und softwaretechnischen Funktionen kann erhalten blieben, indem bspw. individuelle Einstellungen an einem Standardsoftware-Baustein möglich sind (vgl. Wolff et al. 1999: 303), wie es z.B. für ERP-Systeme typisch ist. Dann besteht keine Abhängigkeit, weil sich für die individuellen Einstellungen die anwendende Organisation nicht mit anderen abstimmen muss.

Ob Conways Law bewusst zu unterbinden oder Abhängigkeiten bei Standardbausteinen zu managen: Für die soziotechnische Netzwerkarbeit ist die Softwarearchitektur entscheidend, weil sie die Organisation der Softwaregestaltung prägt (wie die Fallstudien zeigen werden).

6.4.3. Softwaregestaltende: Arbeiten zwischen Anwendung und Programmierung

Ein weiteres zentrales Element der soziotechnischen Netzwerkarbeit sind die Wissensarbeitenden selbst. Sie sind entscheidend für das Verständnis der Fallstudien. Für die Arbeit und wie der Analyserahmen zeigt (siehe 8.1.3), fungiert das Konzept der Rolle allgemein als theoretisches Bindeglied zwischen Individuum und Organisation. Bei der so-

19 Die Firma hat auf Micro Services als Teil einer Service Oriented Architecture (SOA) umgestellt.

ziotechnischen Netzwerkarbeit sind Rollen dreifach strukturgebend: als Rollen im herkömmlich soziologischen Sinne, als Rollen speziell, um Netzwerke zu etablieren, und als spezifische Rollenerwartungen der Softwaregestaltung. Bevor der Abschnitt dies anhand von Forschungsliteratur zeigt, verortet der nächste Punkt die Softwaregestaltenden zwischen Anwendung und Programmierung in die allgemeine Diskussion zur Wissensarbeit und liefert einige Zahlen zu dieser Berufsgruppe. Die Forschung zur Rolle der Produktmanagenden in einer Softwarefirma verdeutlicht, wie Softwaregestaltende arbeiten und welche Erwartungen an sie bestehen.

6.4.3.1. Wissensarbeit Softwaregestaltung: Vermitteln zwischen Anwendung und Programmierung

Softwaregestaltung ist als Wissensarbeit eine Form von Dienstleistungsarbeit. 2021 arbeiteten in Deutschland 33,66 Mio. Beschäftigte im Dienstleistungssektor, davon 1,07 Mio. in der IT-Branche. Zudem beschäftigten 2020 19 % aller Unternehmen in Deutschland IT-Fachkräfte.[20] Die Zahlen zu einzelnen IT-Beschäftigtengruppen sind leider nicht so detailliert. Für die Softwaregestaltenden sind nur wenige vorhanden[21].

Tabelle 3: Veränderung der IT-Beschäftigten in drei Kategorien zwischen 2013 und 2022.

Berufsgruppen IT	31.12.2012	30.06.2022	Differenz	in %
Informatik	201.678	291.905	90.227	+45 %
IT-Systemanalyse, Anwendungsberatung, IT-Vertrieb	135.194	218.601	83.407	+62 %
– ERP-Beratung	96.486	k. A.		
IT-Netzwerktechnik, IT-Koordination, IT-Administration, IT-Organisation	138.139	198.914	60.775	+44 %
– IT-Projektleitung	22.734	k. A.		
Softwareentwicklung und Programmierung	152.274	304.005	151.731	+100 %
Summe	627.285	1.013.425	386.140	+62 %

(Quelle: https://statistik.arbeitsagentur.de, abgerufen Februar 2023 und Zahlen von 2012 für ERP-Beratung und IT-Projektleitung von https://job-futuromat.iab.de, abgerufen Februar 2023)

Für IT-Projektleitung und ERP-Beratung gibt es nur Daten für 2012. Es zeigt sich auf jeden Fall, dass sich in den letzten zehn Jahren die Anzahl der Programmierenden verdoppelt hat. Dass es über 100.000 ERP-Beratende, über 22.000 IT-Projektleitende und mehr als 300.000 Programmierende gibt, belegt, dass Organisationen permanent ihre

20 Vgl. https://www.destatis.de, abgerufen am 16.11.2022.

21 Es gibt zwar auch Kategorien wie »43413-102 – ERP-Anwendungsentwickler/in« in der Statistik der Arbeitsagentur. Allerdings fehlen dazu die Zahlen. Die Daten für die ERP-Berater:innen und IT-Projektleiter:innen für das Jahr 2012 wurden von der Webseite https://job-futuromat.iab.de des IAB kopiert.

Softwarelandschaften verändern. Diese Zahlen sind allerdings lediglich grobe Richtwerte, weil sich bspw. IT-Projektleitende nicht alle mit der Softwareentwicklung/-gestaltung beschäftigen. Zudem fehlen Rollen wie Product Owner:in, Scrum Master:in, Anforderungsmanagende, Anwendungs- oder Applikationsbetreuende oder Key User:in. Es ist auch wenig über deren Qualifikationsprofile, Karriereverläufe oder Arbeitsbedingungen bekannt (ob qualitativ oder quantitativ). Fest steht jedoch, dass sie, wenn sie in Nicht-IT-Unternehmen tätig sind, zu den indirekten Bereichen der Unternehmen gehören und somit die Produktion oder die Erbringung von Dienstleistungen unterstützen. Dort gilt es den technischen Fortschritt zu steuern und zu bewerten und Innovationen anzustoßen. Sie gehören zur steigenden Anzahl an Akademiker:innen in den Organisationen (vgl. Strulik 2017: 322). Als Höherqualifizierte ist deren zunehmende Zahl Teil der Polarisierung der Qualifikationsstruktur im Dienstleistungsbereich und Teil des Trends zur Höherqualifizierung in mittleren und höheren Dienstleistungsjobs (vgl. Overbeck 2017: 103ff.).

Was zeichnet diese Wissensarbeit grundsätzlich aus? Arbeit mit Software ist erstens Arbeit mit Wissen, Informationen und Daten (für die Unterscheidung siehe Definition unter 3.1.1). Zweitens geht es darum, zwischen der eigenen Arbeit und ihrem Kontext wie Organisationsnetzwerk, Finanzzahlen bzw. IT-Budget und Software zu vermitteln. Drittens spielen die Erwartungen und der Umgang mit Erwartungen eine wichtige Rolle (vgl. Schmiede 2015: 49f., Boes und Kämpf 2017: 184, Baudach 2018: 71).

Diese drei Elemente und noch für die Softwaregestaltung spezifischere finden sich bei der Rolle Produktmanager:in. Sie ist eine der wenigen softwaregestaltenden Rollen, zu denen es qualitative Forschungsarbeiten gibt. Es zeigt sich, was Organisationen von Softwaregestaltenden erwarten: interdisziplinär, kooperativ und kommunikativ arbeiten, Beziehung aufbauen und verschiedene Handlungsorientierungen kombinieren (wie es auch typisch für Projektarbeit ist, siehe 4.2). Wie für Arbeit in Netzwerken typisch, agieren Produktmanagende »als Gleiche unter Gleichen; sie können in ihrer Tätigkeit nicht auf Anweisungen zurückgreifen, sondern müssen Probleme im Diskurs lösen« (Bolte 2017a: 483). Die Rolle ist auf die Kooperation andere Beteiligter angewiesen und die meiste Arbeitszeit verbringt sie mit Kommunikation (vgl. Bolte 2017b: 488). Sie hat vielfältige externe und interne Beziehungen: mit der Kundschaft, kooperierenden Unternehmen, Abteilungen des eigenen Unternehmens wie Service, Vertrieb etc. (vgl. Bolte 2017a: 484). Zentral ist dabei der Austausch mit der Kundschaft und den Programmierenden (vgl. ebd. 489).

> Produktmanagende sind Mediator:innen, »die Brücke[n] zwischen den beiden Welten schlagen und Übersetzungsleistungen von der einen Welt in die anderen erbringen« (Weishaupt/Hösl 2017: 505).

Es handelt sich um Interaktionsarbeit mit einer dialogisch-explorativen Vorgehensweise und einer gegenstands- und handlungsvermittelten Kommunikation (vgl. Bolte 2017b: 490). Damit ist das Arbeitshandeln erfahrungsgeleitet-subjektivierend und nicht nur planmäßig-rational (vgl. Weishaupt/Hösl 2017: 494).

> Es geht um »wirkliches Verstehen des Problems durch Nachvollziehen, Hineinversetzen in die Position des Anderen und eine anerkennende, partnerschaftliche und empathische Beziehung zum Gegenüber« (Weishaupt/Hösl 2017: 505).

Indem Produktmanagende die Aktivitäten mehrerer Abteilungen koordinieren, nehmen sie Planungs- und Managementaufgaben wahr (vgl. Bolte 2017b: 487). Zusätzlich dazu brauchen sie fundiertes fachliches Wissen über Inhalte und Möglichkeiten der Softwareentwicklung, Kenntnisse des eigenen Produkts und über die Anwendungssituation (vgl. Weishaupt/Hösl 2017: 501).

Die Beschreibung der Rolle Produktmanager:in zeigt, wie zentral eine einzige Person für die Softwaregestaltung ist, indem sie Beziehungen knüpft und aufrechterhält; Wissen aneignet und einbringt; Anforderungen an die Software erstellt. Sie gewinnt und verarbeitet nicht nur Wissen, sondern ist Erzeuger:in und Träger:in von soziotechnischen Netzwerkstrukturen.

Eine andere Studie geht auf die Grundlagen der Handlungsregulation von IT-Projektleitenden, Beratenden und Customizer:innen in einem ERP-Einführungsprojekt ein. Sie bestätigt und ergänzt, was bereits oben zu softwaregestaltenden Wissensarbeitenden und Produktmanagenden geschrieben wurde: Sie müssen komplexe Aufgaben bearbeiten und große Informationsmengen sammeln, deuten und integrieren. Es wird Selbstmanagement und die Bereitschaft und Fähigkeit zu effektiver kooperativer Arbeit erwartet (vgl. Bläsche/Lappe 2006: 307).

6.4.3.2. Rollen der Softwaregestaltenden: erwartungsgeleitetes Handeln

Analytisch gesehen verbinden sich in der Rolle Produktmanager:in drei strukturgebende Elemente von soziotechnischer Netzwerkarbeit: Es ist eine Rolle im herkömmlichen Sinne, eine Rolle, speziell um Netzwerke zu etablieren, und in ihr zeigen sich Rollenerwartungen, die für soziotechnische Netzwerkarbeit typisch sind. Damit ist das theoretische Konzept der Rolle Teil der Kontrollform soziotechnische Netzwerkarbeit. Weil Rollen in der Arbeitssoziologie bei der Kontrolle von Arbeit nicht im Fokus stehen, soll dieser Absatz kurz verdeutlichen, wie Kontrolle durch Rolle funktioniert.

Softwaregestaltung als erwartete Subjektivität: Rollen, Subjektivierung und subjektivierendes Arbeitshandeln

Was ist eine Rolle? Sie wird als Bündel an Erwartungen an die Träger einer Position definiert, die von Einzelnen unabhängig, sozialstrukturell verankert, von Normen geregelt und von Sanktionen beeinflusst sind (vgl. Griese 2002: 458). Dahrendorf hat Handelnde, die nicht dem Nutzenkalkül folgen, sondern gemäß einer Rolle handeln, als Homo sociologicus bezeichnet. Als solcher agiert jemand aber nicht automatisch. Die Rollenerwartungen sind nicht immer klar, ihre Einhaltung wird nicht immer überwacht. Es muss erst gelernt werden, wie die Rolle auszufüllen ist. Manchmal steht, was die soziale Norm verlangt, im Widerspruch zu dem, was die oder der Einzelne von sich aus tun will (vgl. Schimank 2010: 171). So muss zwischen dem (komplikationslosen) role-taking und der aktiven, interpretierenden Auseinandersetzung mit einer Rolle im role-making unterschieden werden (vgl. Schimank 2010: 67). Wie bereits unter 6.3 beim Thema IT-Projektarbeit erwähnt, ist das Konzept der Rollenkonflikte für die Netzwerkarbeit relevant.

Denn in der Matrixorganisation kann eine Person mehrere Rollen haben: eine im Projekt und eine in seinem Team. Beide können miteinander in Konflikt geraten und der Einzelne muss diese Konflikte lösen (bspw. Zeit für das Team vs. Zeit für das Projekt).

Warum sind Rollen in der Softwaregestaltung wichtig? Das kann ausgehend von der Forschung zur Arbeit in der Filmbranche gezeigt werden. Dort verlassen sich am Filmset ab dem ersten Moment die Projektmitglieder »on role expectations to guide relationships and tasks« (Bechky 2006: 14). Die Rolle (ob Regisseur, Kameramann etc.) ist auch deshalb so zentral, weil, anders als in einem technischen Kontext (z.B. am Fließband), der Beitrag der Technik zur organisatorischen Stabilität gering ist. Es ist mehr die erhöhte Sichtbarkeit des Einzelnen am Set, welche den Einzelnen unter Druck setzt, die Rollenerwartungen zu erfüllen (vgl. ebd.: 15). Was die Fallstudien im 8. Kapitel zeigen: Rollen sind immer angepasst an den jeweiligen Kontext und in Bezug zu den jeweiligen Kolleg:innen. Das zeigen auch die Rollen am Filmset, obwohl sie sehr klar und langfristig institutionalisiert sind. Es findet ein role-making statt: Sie werden gelernt, ausgearbeitet, ihre Umsetzung mit anderen verhandelt (vgl. ebd. 16f.).

Wenn die Erwartungen an die Rollen bekannt sind (z.B. an Scrum Master:innen), erwartet jede beteiligte Person deren Einhaltung. Ist eine Rolle wie das oben beschriebene Produktmanagement erst einmal etabliert, geht sie ihrer Arbeit selbstständig nach. Eine Forschung zur Softwareentwicklung stützt diese Sichtweite: Es ist wichtig, Verantwortung klar zuzuordnen, passende Fachleute einzubeziehen oder die einzelne Person für unterstützende und helfende Rollen fit zu machen, statt zusätzlich technisches Wissen zu lernen (vgl. Waterson et al. 1997: 96f.).

Neben diesen klaren, einzelnen Rollen sind für die Arbeit in soziotechnischen Netzwerken rollenunabhängige Erwartungen für die Handlungsorientierung wichtig. Wie bereits bei der Definition von Wissensarbeit und der Beschreibung der Produktmanager:in gezeigt, bestimmen die Arbeit 1. allgemeine Erwartungen an Wissensarbeitende (Subjektivität), 2. Erwartungen speziell für die Softwaregestaltenden und 3. Erwartungen, die sich situativ ergeben.

Es bestehen 1. unabhängig von der konkreten Rolle im Netzwerk allgemeine Erwartungen an die Softwaregestaltenden, von deren Erfüllung die erfolgreiche Transformation der Arbeitskraft abhängt. Allgemein ist Wissensarbeit Teil der Subjektivierung der Arbeitswelt. Statt sich von den Eigenschaften einzelner Beschäftigter möglichst unabhängig zu machen wie im Taylorismus, werden bei der Subjektivierung »spezifische subjektive Eigenschaften und Fähigkeiten« (Minssen 2011: 120) im Arbeitsprozess genutzt. Es besteht die »Erwartung von Unternehmen, dass diese Fähigkeiten tatsächlich eingebracht werden« (ebd.: 119). In der Wissensarbeit wird nicht nur planmäßig-rationales Handeln erwartet. Gehandelt werden soll situativ, erfahrungsgeleitet-subjektivierend, dialogisch-interaktiv, entdeckend-explorativ und kooperativ. Technik soll als etwas »Lebendiges« betrachtet werden, wie Subjekte, auf die man sich einstellen kann (vgl. Böhle 2010: 160ff.).

Es bestehen 2. neben diesen allgemeinen Erwartungen an Wissensarbeitende auch spezifische an jene, die Software gestalten. Kapitel 4 und 5 haben gezeigt, dass sie mit einer zeichenbasierten Technologie aus mehreren technischen Ebenen konfrontiert sind, auf die Anwendende und Entwickelnde unterschiedliche Perspektiven haben. Interdisziplinäres Arbeiten wird erwartet, bei dem Kommunikation und Wissen die wesentlichen

Arbeitsmittel sind. Kapitel 6 hat nun auch noch gezeigt, dass durch die große Bedeutung von Beziehungen, die über Hierarchien und Märkte hinausgehen, und der Technik selbst Erwartungen bestehen: sei es, Beziehungen herzustellen und zu pflegen oder sich auf die zu gestaltende Software einzulassen.

Letzteres ist wichtig für die Analyse soziotechnischer Netzwerkarbeit, weil Erwartungen 3. erst situativ erkannt und abgeklärt werden müssen und die einzelne Person entscheiden muss, ob die Erwartungen Teil des eigenen Rollen-Sets werden. Das betrifft vor allem die technischen Erwartungen, wie sich mit der Software auseinanderzusetzen und sich auf sie einzulassen. Im technischen Umfeld wird bspw. IT-Affinität erwünscht. Aber wie weit diese geht, ob jemand Fehler analysiert oder sich selbstständig mithilfe der Software und deren Dokumentation einarbeitet, hängt vom Arbeitskontext und den dort vorhandenen Sanktionen, der Sozialisation und dem intrinsischen Wollen der einzelnen Person ab.

Softwaregestaltung als Erfüllen multipler Erwartungen

Es gibt fünf Erwartungen, die die Forschung vielfach untersucht und thematisiert hat und in denen sich die normative Handlungsorientierung in Bezug auf die soziotechnische Netzwerkarbeit zeigt: kooperativ sein, selbstständig arbeiten, sich (tiefergehender) auf Software einlassen, mit Nicht-Wissen umgehen lernen und sich in den organisationalen Netzwerken bewegen. Diese Erwartungen sind vergleichbar mit unkonkreten Aufgaben, die zu erledigen sind. Deren konkreter Inhalt und die Umsetzung müssen die Beschäftigten als Rollenträger selbst situationsabhängig bewerkstelligen.

Netzwerkspezifische Erwartungen: Grenzen zu überbrücken und kooperativ zu sein

Eine vielfach in der Literatur zu findende Rolle ist jene des Boundary Spanners, der Beziehungen stiftet (siehe auch Bolte 2017: 473ff., Williams 2002, Levina/Vaast 2005). Sie werden auch »brokers« genannt und sollen verschiedene Gruppen miteinander verbinden (vgl. Swan/Scarbrough 2005: 932ff.). Sie agieren in einem doppelten Handlungsrahmen aus Netzwerk einerseits und Unternehmen andererseits (vgl. Sydow/Windeler 2000: 5, auch Kalkowski/Mickler 2015: 74). Die Rolle des Boundary Spanners kann durch Outsourcing entstehen, um als Teil »neue[r] Verbindungs- und Koordinationsfunktionen und entsprechende[r] Arbeitsrollen« (Flecker/Holtgrewe 2008: 322) für die organisationsübergreifende Zusammenarbeit zu sorgen.

Der Boundary Spanner hat die Aufgaben, das Unternehmensnetzwerk zu stabilisieren, Vertrauen zwischen kooperierenden Unternehmen trotz Abhängigkeit und Konkurrenz zu erhalten (vgl. Hirsch-Kreinsen 2002: 116) oder die Arbeit Einzelner zu einem Ganzen zu koordinieren (vgl. Heckscher 2015: 246ff.). Weil der direkte Einfluss der Hierarchie fehlt, agieren sie mittels Konsens, Beeinflussung, Verhandeln, Mediation etc., um Deals mit unterschiedlichen Parteien zu machen (vgl. Williams 2002: 117). Sie gewinnen ihre Macht dadurch, dass sie bestimmte Ressourcen über ihre Netzwerke aktivieren können. Im Zuge dessen werden auch Bedeutungen generiert und geteilt (vgl. Swan/Scarbrough 2005: 935). Hilfreich sind dabei interorganisationale Erfahrungen, transdisziplinäres Wissen und kognitive Fähigkeiten (vgl. ebd. 119). Es können im Sinne der Interdisziplinarität Leute sein, die zwischen IT und Geschäftsmodell vermitteln

(vgl. Miozzo und Grimshaw 2005: 1434). An oben genannte Produktmanagende oder auch an IT-Projektleitende werden solche Erwartungen gestellt.

Wissensarbeitsspezifische Erwartung: selbstorganisiert zu arbeiten

Teil der Arbeit in Netzwerken ist zudem, dass Führungskräfte eine andere Rolle spielen. Führungskräfte nehmen die Rolle von Unterstützenden der Wissensarbeitenden ein und werden »nahezu vollständig von ihrer klassischen hierarchischen Koordinations-, Steuerungs- und Kontrollfunktion entbunden« (Korge/Buck/Stolze (2016) nach Baudach 2018: 170). Durch Organisationsformen wie fremdorganisierte Selbstorganisation (vgl. Pongratz/Voß 1997: 212), Dezentralisierung (vgl. Hirsch-Kreinsen 1995) oder »marktgesteuerte Dezentralisierung« (Sauer 2018) werden nicht nur Aufgaben delegiert, sondern die Erwartung geschaffen, dass Mitarbeitende selbstorganisiert arbeiten. Damit gibt es Gemeinsamkeiten zwischen der Vermarktlichung innerhalb von Organisationen und den (Markt-)Beziehungen zwischen Firmen im Netzwerk (vgl. Sauer 2018: 193). Dabei ist je nach Position unterschiedlich, was dezentral das Team oder einzelne Mitarbeitende entscheiden: die Ausführung der Arbeit, ihre Überwachung, Design der Arbeit, Zuordnung von Ressourcen, Leistungsmanagement, Firmenstrategie (vgl. Lee/Edmondson 2017: 13) oder ökonomische Ziele (vgl. Hirsch-Kreinsen 1995: 425). Das betrifft speziell die Anwendung von Software, weil dort »mehr Entscheidungsspielräume und Verantwortlichkeiten auf der ausführenden Ebene« (Schulz-Schaeffer/Funken 2008: 20) existieren.

Zur Erwartung, selbstorganisiert zu arbeiten, gehört, sich mit anderen abzustimmen, sich selbst managen zu können und die Fertigkeiten seiner Kolleg:innen zu kennen. Das zeigt das Beispiel der Gruppenarbeit. Sie macht Hierarchien unwichtiger, dezentralisiert Kompetenzen und bindet innerbetriebliche Positionen an individuelle Fähigkeiten und individualisiert sie dadurch (vgl. Minssen 1999: 207). Es bestehen Erwartungen, die auch sonst bei der Zusammenarbeit von Fachleuten bestehen: Kommunikation als Teil der Arbeit zu sehen und seine Kollegenschaft gut genug zu kennen, damit die Koordinierung in der Gruppe mittels Diskurs und Kommunikation erfolgen kann und nicht durch Bürokratie und Hierarchie (vgl. ebd.: 211). Wer was weiß und mit wem wie kommuniziert, ist wichtig zu wissen, weil keine festen, bürokratischen Verfahrensweisen mehr vorhanden sind (vgl. Kotlarsky/Van Fenema/Willcocks 2008: 4). Erst mit diesem Wissen kann eine Selbstorganisation/-koordination stattfinden.

> »Team members must be familiar enough with each other's experiences, skills, and specialized knowledge to facilitate the emergence of expertise coordination processes« (Faraj/Sproull 2000: 1564).

Für einige Arbeitende war die Umstellung auf Gruppenarbeit schwierig, weil es nicht zum Selbstverständnis passt, Kommunikation als Teil der Arbeit zu sehen, und Gruppenarbeit eher etwas ist, was man Angestellten zuschreibt (vgl. Minssen 1999: 215). Für die einzelnen Teammitglieder bedeutet das effektives Selbstmanagement und damit einen höheren Level an »psychological development and interpersonal skills« (Lee/Edmondson 2017: 18).

Softwarespezifische Erwartung: mit Software interagieren und Beziehung zum Austausch über sie aufbauen

Eine weitere Erwartung ist, sich auf die Software einzulassen, sie kennenzulernen, zu verstehen und dies immer wieder zu tun. Wie bereits bei dem Thema Kontrolle durch Software geschrieben (siehe 6.4.2.1), sieht Rennstam als wichtigen Teil der Produktivkraft von Software an, dass ein:e Wissensarbeiter:in zu ihr eine Beziehung aufbaut:

> »[T]heir relationships with the objects of knowledge […] give rise to most of the valuable ›knowledge work‹« (Rennstam 2012: 1087).

Das betrifft die verwendeten Werkzeuge, aber auch die zu entwickelnde Software. Wie auch schon bei der Selbstorganisation spielt Kommunikation durch den vermehrten IT-Einsatz eine immer größere Rolle. Fallstudien zur Einführung von IT im Bereich Workplace Studies zeigen, dass kommunikative Handlungen notwendig sind, um Arbeit an technologisch vermittelnden Systemen durchzuführen, und dass der Arbeitstag vor allem mit Kommunikation gefüllt ist. Arbeit muss abgestimmt, integriert und koordiniert werden (vgl. Knoblauch 1996: 359). Es ist Teil der Arbeit, den elektronischen Text zu interpretieren, und er ist Ausgangspunkt für Koordination und Kommunikation (vgl. Zuboff 1988: 393f.). Organisationen, die Informationstechnologie einsetzen, sollten organisiert sein »to promote the possibility of useful learning among all members and thus presupposes relations of equality« (Zuboff 1988: 394). Öffentliche Debatten und gegenseitige Beeinflussung müssen legitim sein (vgl. Zuboff 1988: 406). Darüber hinaus ist bei der Arbeit mit Software die Subjektivität jedes Einzelnen umfassend gefragt. Sie müssen persönliche Erfahrungen, implizites Wissen, Netzwerkfähigkeit, offene Kommunikation, Spontanität, Motivation und Kreativität einbringen (vgl. Schilcher/Diekmann 2012: 37). ERP-Systeme verlangen »zahlreiche Praktiken der Überprüfung und Reparatur und diese nicht nur vorübergehend, sondern kontinuierlich« (Conrad 2017: 182). Statt des immer gleichen Handgriffs verlangen die in der Softwaregestaltung entwickelte Software und die verwendeten softwarebasierten Werkzeuge mentale Interaktion und situatives Handeln von den Anwendenden.

Softwaregestaltungsspezifische Erwartung: mit Nicht-Wissen in Kontext von softwaretechnischer Interdisziplinarität umzugehen

Die Arbeit der Softwaregestaltung basiert nicht nur auf Wissen, sondern auch darauf, mit Nicht-Wissen umzugehen. Das liegt an den sich ständig ändernden und komplexen Wissensbeständen, dem Transfer des Wissens in unterschiedliche Kontexte und dem dynamischen Wechsel von der Expert:innen- in die Lai:innenrolle.

Ob es um gesetzliche Regulierungen im Energiebereich oder neue Funktionalitäten von ERP-Systemen geht: Ständig müssen die Softwaregestaltenden damit zurechtkommen, dass sie etwas nicht wissen. Einige Autoren sehen den Umgang mit Nicht-Wissen allgemein als einen entscheidenden Faktor der Wissensarbeit an.

> »Wissen ist mithin – zusammengefasst – nicht positiv fest stellbarer Tat-Bestand, sondern es ist beständiger Prozess, unendliche Bemühung, Kampf gegen das Nichtwis-

> sen, fundamental subjektive, aber immer auch objektiv vermittelte Bewährung in einer grundlegend unbestimmten Welt« (Schmiede 2006: 16).

Dabei spielen die Wissensarbeitenden die tragende Rolle. Von ihnen hängt es ab, das Wissen auf den jeweiligen Kontext zu beziehen, es zu interpretieren und sich mit anderen darüber zu verständigen (vgl. Schmiede 2006: 15). Umgang mit Nicht-Wissen ist bspw. auch Teil der Arbeit von Beratenden. Diese müssen »sich von einer Praxis belehren lassen, welche sie belehren sollen« (Willke 1998: 173). Sie müssen nicht nur das fachliche Wissen haben, sondern auch die Besonderheiten der Organisationen und der angewendeten Software kennenlernen, die sie beraten wollen.

Zu einer besonderen Dynamik beim Nicht-Wissen kommt es in der Softwaregestaltung, weil sich in der Softwaregestaltung je nach Situation ändert, wer Lai:in und wer Expert:in ist. Die Software kann für Lai:innen eine Blackbox sein. Für Expert:innen kann sie allerdings ein mächtiges Werkzeug sein, dessen Potenziale sie zu nutzen wissen. Schulz-Schaeffer sieht es als Grundlage sämtlicher Technik an, dass sie eine Leistungssteigerung mithilfe von Expert:innen und damit eine Sinnentlastung für Lai:innen möglich macht (vgl. Schulz-Schaeffer 1999: 424). Bei der Softwaregestaltung kommt es zum Rollentausch: Anwendende sind Expert:innen in ihrem Anwendungsbereich und IT-Beratende, Programmierende, IT-Projektleitende o. ä. auf ihr Wissen angewiesen. Ingenieur:innen im Kraftwerks- oder Netzbereich oder Stromhändler:innen haben Domänenwissen, was den IT-Fachleuten fehlt. Vermittelnde wie bspw. Product Owner:innen oder IT-Projektleitende changieren zwischen IT- und Fachwissen. Sie haben ggf. nur so viel Fachwissen wie nötig oder konzentrieren sich ganz darauf, den Austausch zwischen Anwendung und Programmierung zu organisieren.

Netzwerkspezifische Erwartung: sich im Netzwerk zu bewegen und zu lernen (neue Karrieremöglichkeiten)

Anders als in einer Bürokratie ergeben sich im Netzwerk für die Softwaregestaltenden neue Karriere- und Lernmöglichkeiten. Die Arbeitenden sollen sich im Netzwerk bewegen. Die sich dadurch ergebenden unterschiedlichen Beziehungen, in denen sich Beschäftigte (temporär) befinden, machen eine Selbstdarstellung wichtiger.

Projekte sind temporär, die Softwaregestaltung verläuft in Phasen (mal ist mehr, mal ist weniger zu tun; mal an diesem, mal an jenem Softwarebestandteil zu arbeiten). Das führt zur Erwartung an die Beteiligten, sich in temporäre Beziehungen zu begeben und immer wieder offen für neue zu sein. Statt eines festen Beziehungszusammenhanges und klarer Karrierewege in einer Organisation entsteht die Möglichkeit, sich horizontal innerhalb einer Firma oder zwischen Firmen zu bewegen und Karriere zu machen. So ergeben sich für IT-Fachkräfte im ehemals öffentlichen Sektor neue Karrieremöglichkeiten, aber auch Zwänge, sich im Netzwerk weiter zu bewegen (vgl. Flecker/Holtgrewe 2008: 330). Dabei geht es auch darum, dass man nur Neues lernt, wenn man Teil von Netzwerken ist und sich in ihnen bewegt. Wie Hohlmann gezeigt hat, entstehen bei der Implementierung der ERP-Software von SAP Gestaltungsnetzwerke, weil diejenigen, die beim Implementierungsprojekt mitgemacht haben, etwas dazugelernt haben und langfristig die Gestaltung der Software übernehmen (vgl. Hohlmann 2007). Es kann Teil der Implementierungsstrategie sein, dass Anwendende bei der Auswahl von Hard- und Soft-

ware mitmachen, sie einführen und sich gegenseitig trainieren, damit sie sich Wissen aneignen (vgl. Robey/Sahay 1996: 106f.). In eine Position zu kommen, die eine Kombination von IT- und Branchenwissen verlangt, fördert den Aufbau von Kompetenzen (vgl. Ramioul/De Vroom 2009: 63). Dies ist es ja auch, was in der interdisziplinären Arbeit der Softwaregestaltung gefragt ist und für manche Befragte aus den Fallstudien im 8. Kapitel den Reiz ihrer Arbeit ausmacht.

Zu dieser Bewegung in Netzwerken gehört, sich immer wieder auf neue Beziehungen einzulassen. Weil das jeweilige Wissen nicht direkt dem neuen sozialen Kontakt transparent ist und man mit manchen Beteiligten (vor allem Führungskräften) nur kurz zu tun hat, ist Selbstdarstellung wichtig. Wer als einzelne Person in einem Projekt bspw. nicht untergehen will, muss am Impression Management arbeiten. Eine Untersuchung kommt zum Schluss, dass Projektmitarbeitende zu Projektdarstellenden werden (auch vor den Führungskräften) und die Karriere als Inszenierung begreifen (vgl. Funken/Stoll/Hörlin 2011). Wie stark dies ausgeprägt ist, mag variieren. Aber die Erwartung existiert und einzelne Personen erfahren Sanktionen, wenn Entscheidungsträger:innen nicht mitbekommen, was sie leisten, weil sie ihren Führungskräften nicht darstellen, was sie geleistet haben.

6.4.3.3. Softwaregestaltende als Konkurrenz zum Management? Neue Kompetenzen und Aufgaben

Welche Rolle spielt das Management bei der Kontrolle der Arbeitskraft im Arbeitsprozess der Softwaregestaltung? Die Wissensarbeitenden der Softwaregestaltung treten in Konkurrenz zu den bestehenden Managementstrukturen. Wie bereits unter 4.2. ausgeführt, kommt Hohlmann zu dem Schluss, dass durch die entstehenden Gestaltungsnetzwerke bei der ERP-Einführung alte Hierarchien untergraben und neue Kanäle geschaffen werden, wobei für deren Macht Wissen und Expertise entscheidend ist (vgl. Hohlmann 2007: 359f.). Die Forschung hat bereits festgestellt, dass bestimmte Berufsgruppen ihre Position verbessern wollen, indem sie alte Vorstellungen davon, wie man Arbeit am besten kontrolliert, angreifen. Dies ist Teil einer allgemeinen Kritik an einer funktionalistischen Sichtweise auf das Management: Es ist nicht immer gesagt, dass das Management die Arbeit am effizientesten organisiert und eine Technologie optimal einsetzt. Hier sind es aber weniger neue Managementmethoden als vielmehr technikinduzierte Möglichkeiten und Notwendigkeiten der Arbeitsgestaltung, die den Softwaregestaltenden Aufstiegsmöglichkeiten und Einfluss auf die Kontrolle der Arbeit in den anwendenden Organisationen eröffnen.

Armstrong (1985) sieht das Scientific Management à la Taylor als Methode an, die legitimieren soll, dass Ingenieur:innen am besten geeignet sind, den Arbeitsprozess zu kontrollieren, und berechtigt sind, eine gehobene Position einzunehmen (vgl. ebd.: 131). Er geht davon aus, dass eine Berufsgruppe aufsteigen kann, wenn es ihr vorzugeben gelingt, eine Lösung für eines der Kernprobleme des Kapitals zu haben (in diesem Falle: das Kontrollproblem) (vgl. ebd.: 133). Als sich dann im weiteren Verlauf der Geschichte die Buchhalter:innen und Controller:innen durchsetzen, wird das Senior Management zu »financial rather than operational or technical decision-makers« (ebd.: 136). Als nächste Gruppe würden nun auch die IT-System-Analyst:innen aufsteigen:

> »Systems analysts appear to be advancing in corporate hierarchies partly by displacing, de-skilling and devising new systems for the surveillance of productive labour, and partly by cannibalizing the tasks of the ›traditional‹ organizational professions« (ebd.: 145).

Dabei sind das keine objektiven Bedarfe der Organisation. Vielmehr sind es Anstrengungen einer Berufsgruppe

> »to develop their original techniques into a system of managerial control – competitive with other approaches – precisely as a means of achieving managerial ascendancy« (ebd.: 145).

Damit kritisiert er eine Sicht auf das Management, die auch Morozov (2019) an dem Ansatz von Zuboff in ihrem Buch über den Überwachungskapitalismus (2018) kritisiert. Diese hätte eine funktionalistische Sicht auf das Management. Sie würde der Ansicht von Chandler (1990) folgen, der das Management als etwas ansah, das für den effizienten Einsatz von Technologien wie Telefon und Eisenbahn sorgte und dadurch so wichtig in manchen Unternehmen wurde.

Bestimmte Gruppen erlangen Einfluss allein dadurch, dass eine Technologie auf bestimmte Akteur:innen oder Organisationsformen angewiesen ist, um ihre Möglichkeiten auszuschöpfen. Wie eine Fallstudie zeigt, kann die Einführung eines ERP-Systems zu einer technikinduzierten Statusaufwertung in einer Organisation führen. Es entstanden neue Anerkennungsmuster und es kam zur

> »Verschiebung von der Aufwands- hin zur Ergebnisorientierung […], die mit dem Aufstieg der Controlling-Abteilung einhergeht« (Walker 2016: 97).

Indem die IT die Zentralisierung als organisationales Leitbild ermöglicht, wird die Tätigkeit des Controllings aufgewertet (vgl. ebd. 97). Auch Hohlmann weist auf die technikinduzierte Statusaufwertung der Systemgestaltenden wie Key User:innen hin. Sie fungieren u.a. als Puffer für Marktanforderungen (vgl. Hohlmann 2007: 340). Denn Überraschungen, Ausnahmesituationen oder spezielle Wünsche der Kundschaft sind durch die ERP-Software nicht verarbeitbar. Sie kann sich nicht selbst ändern.

In puncto Softwaregestaltung stellt sich vielmehr die Frage, welche Rolle das Management überhaupt noch spielt. Die vorliegende Untersuchung hat bereits an mehreren Stellen darauf hingewiesen, dass bei Wissensarbeitenden die Führungskräfte indirekt steuern. Bei seiner Studie zur Softwareentwicklung spricht Friedman vom Management by Neglect, die er als Form der »responsible autonomy […] of a particularly informal or lax type« (Friedman/Cornford 1993: 322) beschreibt.

Damit sind die Softwaregestaltenden keine konkurrierende Beschäftigtengruppe zum Management, weil sie neue Kontrolltechniken für die Anwendenden einführen, sondern weil sie es sind, von denen abhängt, die Möglichkeiten der Softwaregestaltung auszuschöpfen. Die Fallstudien werden zeigen, dass es nicht das Management per se ist, sondern es bestimmte Führungskräfte sind, deren Position und Funktion der Arbeitsprozess der Softwaregestaltung in Frage stellt. Es gibt andere Personen des Ma-

nagements, die selbst den Arbeitsprozess der Softwaregestaltung in ihrer Organisation etablieren und kontrollieren.

6.4.4. Flexibilität bei der Kommunikation und beim Wissensaustausch

In der Forschung und den Fallstudien zeigt sich als typische Eigenschaft in der Softwaregestaltung ein Phänomen von Arbeit in soziotechnischen Netzwerken: Die verschiedenen Ebenen aus Ablauf, Beziehungen, Software und Softwaregestaltenden können flexibel stark zur Kontrolle der Softwaregestaltung beitragen. Damit ist gemeint, dass es empirisch unterschiedlich ist, wie stark die Transformation der Arbeitskraft von Ablauf, individuellen und organisationalen Beziehungen, Software und den Softwaregestaltenden abhängt. Bei den Forschungsarbeiten, die das zeigen, geht es um Softwareentwicklung. Sie zeigen einerseits, dass der Arbeitsprozess komplett digital und selbstorganisiert sein, und andererseits, dass Kommunikation unterschiedlich stattfinden kann – ob innerhalb oder zwischen Firmen. In mehreren Forschungsarbeiten geht es darum, inwiefern direkte Kommunikation durch andere Formen der Kommunikation ersetzt werden kann[22].

Erstens hängt die Kommunikation laut einer Studie zu projektförmiger Softwareentwicklung von unterschiedlichen Faktoren ab. Die Studie untersucht die Unterschiede zwischen interner und organisationsgreifender Zusammenarbeit: Wenn die Beschäftigten am gleichen Ort arbeiten, von Angesicht zu Angesicht, dann ist es egal, ob eine oder mehrere Firmen beteiligt sind. Ist die Arbeit räumlich verteilt, macht es einen Unterschied: Innerhalb von Firmen findet die Koordination via Common Ground aus Programmierstandards und gemeinsamen Werkzeugen statt. Zwischen Firmen ist ein solcher Common Ground nicht vorhanden und es muss auf Arbeit von Angesicht zu Angesicht zurückgegriffen werden. Ein weiteres Fazit: Wenn stetige Kommunikation und eine modulare Softwarearchitektur (die es Teams erlaubt, unabhängig voneinander zu arbeiten) schwierig sind, dann ist es besser, ein Entwicklungsprojekt innerhalb einer Firma durchzuführen (vgl. Srikanth und Puranam 2014).

Zweitens unterstreicht eine andere Studie zu einem Softwareentwicklungsprojekt die Bedeutung von Softwarewerkzeugen für Koordination und Kommunikation und zeigt gleichzeitig deren Grenzen auf. Softwarewerkzeuge, organisierter und informeller Austausch ergänzen sich, wenn es um die Kommunikation geht. Es existiert eine Mehr-Ebenen-Kommunikation aus E-Mails, Meetings, Telefonaten, Protokollen, Chatgruppen etc. (vgl. Heidenreich et al. 2008: 16). Einerseits gibt es Software zur Planung, Dokumentation, Steuerung des Projektes und eine Software für die Teamarbeit (E-Mail, Kalender, Aufgabenliste etc.). Andererseits werden die Anforderungen selbst analog in einem »Gremium von Managern, Projektleitern, Architekten und Marketing genehmigt« (Heidenreich et al. 2008: 10). Auch wenn jeder Einsicht in Projektfortschritt über das Planungs-Softwaresystem hat und eine effiziente Abstimmung darüber möglich ist, hat diese informationstechnologische Koordinierung ihre Grenzen: Ob Besprechungen

22 Unabhängig von der Softwareentwicklung ist umstritten, unter welchen Umständen die Co-Location oder die computervermittelte Kommunikation für den Wissenstransfer besser ist (vgl. Song et al. 2007).

über Schnittstellen, Fehler oder diverse Projekttreffen – sie finden auch über informelle Wege statt. Das liegt laut Autoren an der Komplexität der Aufgaben (vgl. Heidenreich et al. 2008: 12).

Drittens kann Softwareentwicklung komplett digital ablaufen. Dies war bei einer Open-Source-Entwicklung (dem Betriebssystem Linux) der Fall. Die Personen haben sich zum allergrößten Teil nie persönlich getroffen (vgl. Shaikh/Henfridsson 2017). Bei einem anderen Open-Source-Projekt stellen die Autoren fest, dass selbstständiges Arbeiten mit dem gemeinsamen Arbeiten an der Software verbunden ist (wobei alles digital stattfindet). Es existiert ein Common Ground dank digitaler Kommunikation (ob Chat-Foren oder E-Mail), für jedermann sichtbarer Handlungen und Kontexte (wer welchen Teil des Quellcodes programmiert hat oder gerade an ihm arbeitet) und der Visualisierung gemeinsamer Aufgaben (Ticketsystem). Zudem erlaubt es eine modulare Aufgabenarchitektur, dass Einzelne unabhängig von anderen arbeiten können. Die Beteiligten wählen selbst, welches Arbeitspaket sie bearbeiten (vgl. Puranam/Alexy/Reitzig 2014: 172).

Viertens hängt es von der Softwareentwicklungsplattform ab, wie die Nutzenden dort Wissensgrenzen überwinden (können). Eine Form kann wie jene bei der Entwicklungsplattform von SAP sein, die einige Organisationen aus den Fallstudien im Empirie-Teil einsetzen (siehe 4.38.1.4). Dort gibt es engere Beziehungen und regelmäßigen Austausch zwischen SAP und den kooperierenden Firmen, welche die Software anpassen, erweitern und einstellen. Diese Form, Wissensgrenzen zu überwinden, nennen die Autoren »bridging«. Andere Entwicklungsplattformen stellen nur online Hilfsmittel wie Dokumentationen oder Beispiel-Programmierungen zur Verfügung. Die Nutzenden müssen sich selbstständig mit der Plattform auseinandersetzen. Das nennen die Autoren »broadcasting« (vgl. Foerderer et al. 2019).

6.5. Folgen der Softwaregestaltung: Soziotechnische Arbeitsgestaltung der Softwareanwendung durch die Softwaregestaltung

Für die Frage der Folgen der Softwaregestaltung für die Softwareanwendung ist es schwierig, Forschungsliteratur zu finden. Es gibt zwar viele Arbeiten über die Folgen von Softwareeinsatz. Aber genau herausgearbeitet wurde nur in wenigen Fällen, welche Folgen davon auf den softwaretechnischen Zuschnitt wie Standard oder individuell zurückzuführen sind. Anhand der Forschung zu ERP-Software stellt 6.5.4 dar, welche Folgen für die Anwendung sich auf die Standardsoftwaregestaltung zurückführen lassen. Zu einem konzeptionellen Ansatz führt das jedoch nicht. Dafür nutzt dieser Abschnitt Forschungsliteratur über die Rationalisierung von Arbeit durch IT und über die Digitalisierung allgemein. So zeigt sich, dass das Verhältnis von Softwaregestaltung zur Softwareanwendung eines der Rationalisierung ist, grenzt das Verhältnis von anderen Ansätzen wie jenen der Informatisierung ab und stellt klar, um welchen langfristigen Wandel es dabei geht. Den Ansatz, die Softwaregestaltung als eine soziotechnische Arbeitsgestaltung der Softwareanwendung zu verstehen, arbeitet dann aber erst der Empirie-Teil konzeptionell aus, weil dafür die Literatur zu wenig hergibt.

6.5.1. Softwaregestaltung – eine Form der Rationalisierung der Softwareanwendung?

Es gibt eine Diskussion über die unterschiedlichen Formen der Rationalisierung. Was zwischen Anwendung und Programmierung passiert, ist nicht Teil dieser Diskussion. Die vorliegende Arbeit argumentiert, dass das aber relevant für die Effizienz und die Kontrolle der Arbeit der Anwendenden ist. Statt eines Technikentwicklungsdeterminismus (oder Softwaregestaltung als Blackbox) geht die vorliegende Untersuchung davon aus, dass zwischen Anwendung und Programmierung eine Rationalisierung eigenständigen Typs stattfindet. Sie vertritt eine andere Auffassung als Baethge (1996):

> »Der Weg der Rationalisierung in den Dienstleistungsbereichen war nie und ist auch heute nach dem Siegeszug der Mikroelektronik in den Büros und Verwaltungen *nicht* [Herv. i. O.] in erster Linie eine *Frage von Technikeinsatz und Technikgestaltung* [Herv. i. O.], sondern von *Organisation und Kommunikation* [Herv. i. O.]. Dies haben wir mit der Kategorie der ›systemischen Rationalisierung‹ zu kennzeichnen versucht. Es wird in ähnlicher Weise von Rock/Ulrich/Witt (1990) mit ihrem Begriff der ›kommunikativen Rationalisierung‹ angezielt.« (ebd. 20)

Im Kern der Untersuchung steht eine technikzentrierte Form der Rationalisierung der Dienstleistungsarbeit in den Firmen der Energiewirtschaft. Dabei ist die Technikgestaltung ein wesentliches Element der Rationalisierung, für die Kommunikation das zentrale Mittel ist (vor allem für den Wissensaustausch zwischen energiewirtschaftlichen Domänenexpert:innen und Programmierenden). Diese Kommunikationsprozesse selbst sind rationalisierbar – wie auch die Fallstudien im 8. Kapitel zeigen. Sie werden zu dem Zweck rationalisiert, die Softwaregestaltung zu verbessern.

Die hier vorgestellte Rationalisierung durch Technik stellt eine Ergänzung zu der von Menz/Nies/Sauer (2019, alle folgenden Zitate S. 189) aufgestellten Unterscheidung dreier Formen der Techniknutzung dar:

- Arbeitskraftbezogene Strategien, »die direkt auf die Arbeitskraft und das sogenannte Transformationsproblem zielen«.
- Innerbetriebliche und betriebsübergreifende Prozessrationalisierung (systemische Rationalisierung), bei denen »Prozesszusammenhänge, Friktionen an den Schnittstellen, die Koordination in verzweigten Wertschöpfungsketten etc.« im Fokus stehen.
- »[R]ationalisierungsunabhängige Strategien des Technikeinsatzes, insbesondere Strategien der Kundenbindung und des Marketings, die i.d.R. nur mittelbar auf die Steuerung von Arbeit und die Autonomie der Beschäftigten wirken.«

Software kann für all diese Maßnahmen entwickelt, eingesetzt und bestehende Software entsprechend angepasst werden. Die Gestaltung von Software stellt eine vierte Form dar. Es ist eine **technikentwicklungsbezogene Strategie,** die auf die Rationalisierung durch Technikgestaltung abzielt, und zwar der soziotechnischen Arbeitsgestaltung durch Softwaregestaltung.

Die sieben Fallstudien beschreiben unterschiedliche Varianten dieser Form der Rationalisierung. Dabei berücksichtigen die Fallstudien, dass es vom Anwendungsbereich abhängt, wie groß die Potenziale der softwarebasierten Rationalisierung sind. Wo die Kernarbeit auf reiner Datenverarbeitung basiert (Abrechnung, Handel etc.), gibt es mehr Möglichkeiten der Automatisierung oder Prozessintegration durch Software (siehe Kapitel 8).

Ziel dieser Rationalisierung durch Softwaregestaltung ist es, je nach Anwendungskontext die Möglichkeiten der Softwaretechnik zwischen Standard- und individueller Software optimal zu nutzen. Die Arbeitsteilung in Organisationen und Wertschöpfungsketten verläuft aus dieser Perspektive nicht mehr nach Hand- und Kopfarbeit, sondern in den Kategorien von Softwareprogrammierung, -gestaltung und -anwendung. Das Verhältnis der beiden Arbeitsprozesse reicht dabei von einem evolutionären Wandel von anwendender Organisation und Standardsoftware (vgl. Hohlmann 2007: 342ff.) bis hin zur Disruption durch Start-ups in bestimmten Branchen, die von Anfang an softwareentwickelnde Organisationen sind. Letzteres stellt ein besonderes Verhältnis beider Arbeitsprozesse zueinander dar: den Primat der Softwareentwicklung. Eine Organisation gestaltet eine Software für einen Anwendungsbereich, um diesen möglichst effizient zu bewirtschaften. So ist sie auf den Technikentwicklungsprozess der Softwareentwicklung hin rationalisiert. Übrig bleib die Arbeit für Softwareanwendende, welche Software nicht erledigen kann (oder soll).

6.5.2. Unterschied zu Informatisierung und Informationsraum

Softwaregestaltung in ihrem Verhältnis zur Softwareanwendung zu betrachten, unterscheidet sich von anderen Ansätzen wie jene von Informatisierung und Informationsraum. Es geht um die Informationsverarbeitung und nicht um allgemeine Folgen der Digitalisierung. Es geht um die Folgen des Verhältnisses der Arbeitsprozesse von Softwaregestaltung und Softwareanwendung.

Bei der Informatisierung geht es um einen »soziale[n] Prozess der systematischen Erzeugung und Nutzung von Informationen, um daraus weitere Informationen erzeugen zu können« (Boes/Pfeiffer 2006: 22). Bei der soziotechnischen Arbeitsgestaltung der Softwareanwendung durch Softwaregestaltung steht hingegen die Informationsverarbeitung im Vordergrund. Softwaregestaltung ist die Erzeugung von informationsverarbeitender Software.

Dabei hängen die Möglichkeiten der Softwareentwicklung davon ab, die abzubildenden Prozesse formalisieren zu können:

> »Die Detailsteuerung und Kontrolle der Arbeit hängen nicht nur von den technischen Möglichkeiten, sondern auch von der inhaltlichen, fachlichen und organisatorischen Gestalt der Arbeits- und Produktionsprozesse ab. Die Formalisierung der Arbeitsabläufe ist nicht Konsequenz, sondern Voraussetzung digitaler Durchsteuerung und Kontrolle.« (Nies/Menz/Sauer 2019: 187)

Der Quelltext einer Software kann nur abbilden, was formal beschreibbar ist und die Programmierenden umsetzen können. Doch wie die Fallstudien zeigen werden, macht der

Fokus auf das Verhältnis von Softwaregestaltung und Softwareanwendung klar, dass es nicht nur um die Folgen der Formalisierung von Abläufen für eine softwareanwendende Organisation geht.

Erstens macht es für anwendende Organisationen einen Unterschied, ob sie sich den formalen Abläufen einer Standardsoftware unterordnen oder ihre eigenen Abläufe formalisieren und in eine individuelle Software umsetzen. Zweitens kann im Zuge der Softwaregestaltung eine Software entstehen, die keine formalisierten Abläufe oder eine Detailsteuerung der Softwareanwendung verlangt. Es existiert keine Parallelität von softwaretechnisch-formaler Abbildung und organisatorisch-formalen Arbeitsabläufen in der Anwendung. Fortschreitende Softwarenutzung in einem Anwendungsbereich kann dazu führen, dass die Anwendenden weniger formalisiert arbeiten als vorher, weil sie komplexe Fälle lösen müssen, welche die Software nicht automatisiert verarbeiten kann. Die Software prozessiert tausendfach die formalisierten Abläufe im Hintergrund, ohne dass die Anwendenden eingreifen. Drittens ist, wie Hohlmann (2007) zeigt, stetig Softwaregestaltung notwendig, weil die Software zu wenig flexibel ist, als dass Anwendende mit ihr auf veränderte Anforderungen reagieren können (seien es neue Wünsche der Kundschaft, Ideen für effizientere Prozesse etc.). Da macht es dann einen Unterschied, ob die anwendende Organisation selbst fähig ist, die Software zu gestalten oder nicht. Es macht einen Unterschied, ob sich Software und Anwendung wechselseitig weiterentwickeln, und dies innerhalb einer Organisation oder in Abhängigkeit von einer Softwarefirma.

Die genannten Punkte sollen noch einmal verdeutlichen, dass Softwaregestaltung weder ein rein technologiegetriebener (Programmierung) noch ein rein anwendungskontextgetriebener (Formalisierung von Arbeitsabläufen) Prozess ist. Softwaregestaltung ist der stetige Prozess, Bedarfe eines industriespezifischen Anwendungsbereichs mit den Möglichkeiten der Softwareentwicklung abzugleichen. Viele Möglichkeiten der Softwaregestaltung ergeben sich erst iterativ im Austausch zwischen Anwendung, Gestaltung und Programmierung.

Anders als beim Konzept des Informationsraums von Boes et al. (2006) steht beim Verhältnis der Softwaregestaltung zur Softwareanwendung die Dynamik zwischen beiden Arbeitsprozessen im Fokus. Der Anwendungsbereich ist Gegenstand der Softwaregestaltung und ändert sich abhängig von den verwirklichten Möglichkeiten der Softwareentwicklung. Der Informationsraum hingegen ist nicht vorprogrammiert und ein »offener Raum, der sich erst durch das soziale Handeln seiner Nutzer konstituiert« (ebd. 24f.). Bei der Softwaregestaltung geht es hingegen um die Programmierung und welche Möglichkeiten der Softwaregestaltung zwischen Individual- und Standardsoftware Organisationen für einen Anwendungsbereich nutzen. Es geht, anders als beim Informationsraum, um die »programmierten und starren Informationssysteme« (Boes et al. 2020: 313). Die Fallstudien werden zeigen, dass Software und Arbeitsorganisation, anders als im folgenden Zitat, nicht als getrennt betrachtet werden:

> »Insgesamt zeigt sich sowohl in den mittel- als auch in den höherqualifizierten Bereichen, wie die Kombination eines digitalen ›Raums der Produktion‹ mit Lean und agilen Methoden auf dem Shopfloor die Kopfarbeit neuen Formen der Industrialisierung zugänglich macht.« (Boes et al. 2018: 180).

Vielmehr geht es immer darum, via digitalen Raum eine Software zu gestalten, und wie das die anwendende Organisation verändert. Bereits im Kapitel zur Softwareentwicklung (5.2.3) hat sich gezeigt, dass Firmen die Möglichkeiten des Internets für die Softwareentwicklung sehr unterschiedlich nützen. Weniger die Kombination von Internet und Arbeitsmethoden steht im Fokus der vorliegenden Arbeit. Vielmehr ist es das Verhältnis von Gestaltung und Anwendung, in dem das Internet nur eine von vielen Möglichkeiten darstellt, die beim Abgleich von energiewirtschaftlichen Bedarfen und technischen Möglichkeiten eine Rolle spielen.

So ist mit soziotechnischer Arbeitsgestaltung hier gemeint, dass sich Software, Arbeit und Organisation des Anwendungsbereichs durch Softwaregestaltung verändern. Die folgenden Kapitel arbeiten weiter heraus, welche Veränderungen sich auf die soziotechnische Arbeitsgestaltung zurückführen lassen und sich von den Folgen der reinen Software(funktionalität) unterscheiden.

6.5.3. Softwaregestaltung: inkrementell mehr Software in diversen Anwendungsbereichen

Folgt daraus eine allgemeine These für den Wandel der Arbeitswelt? In einem Artikel aus dem Jahr 2011 sieht der Netscape-Gründer und Silicon-Valley-Investor Marc Andreessen (als Tech-Nerd und durch IT Reichgewordener) eine unaufhaltsame Durchdringung der Wirtschaft mit Software.

> »Software is eating the world.« (Andreessen 2011)

Indizien sind für ihn die immer größer werdenden Softwarefirmen (Amazon, Google, Facebook, Spotify, Netflix, Pixar, LinkedIn) und die immer größere Rolle von Software in allen möglichen Industrien (er nennt die Logistik-Software von Walmart und die Betriebssoftware von E-Autos). Er konstatiert damit einen noch allgemeineren Trend, als ihn die Soziologie für die Digitalisierung feststellt: ob optimierte Überwachung und Verhaltenssteuerung (vgl. Zuboff 2018), gesteigerte Bedeutung der Distributivkraft (vgl. Pfeiffer 2021), neue Handlungsräume (vgl. Boes et al. 2016) oder allgegenwärtige Plattformen (vgl. Srnicek 2017, Staab 2019) – um nur eine Auswahl zu nennen.

Die vorliegende Arbeit betrachtet a) Digitalisierung wie andere Autoren als inkrementellen soziotechnischen Wandel (vgl. Hirsch-Kreinsen 2020: 40ff.). Wobei Disruptionen durch Start-ups nicht ausgeschlossen sind. Sie finden aber auch inkrementell statt. Sie sind nur innovativer, weil sie die Möglichkeiten der Technologie ausschöpfen, indem sie dem Primat der Softwareentwicklung folgen. Zudem geht es b) um eine neue Arbeitsteilung zwischen Anwendung und Entwicklung. Die Untersuchung geht von drei Seiten einer Organisation und Wertschöpfung aus: die Software selbst, die Anwendung der Software und Softwareentwicklung – ob innerhalb einer Firma oder verteilt auf mehrere Organisationen. Damit geht es c) um die Technokratie an Softwareprogrammierenden und -gestaltenden und deren zunehmende (Gestaltungs-)Macht.

Es geht also um soziotechnische Arbeitsgestaltung als inkrementellen Wandel hin zur Software als Basis von Organisation und Arbeit, getragen von einer Gruppe an Fachleuten und unterschiedlichen Formen des Arbeitsprozesses der Softwaregestaltung

als soziotechnische Netzwerkarbeit (und ermöglicht durch Breitband-Vernetzung, Fortschritte bei der Chipproduktion und mobilen Endgeräten). Damit folgt diese Arbeit anderen Forschenden, welche die Digitalisierung aus der Perspektive der Arbeit analysieren, mit ihrer je arbeitsweltspezifischen Ausprägung (vgl. Apitzsch et al. 2021: 20f.).

Die Ansätze von Hirsch-Kreinsen (2020) und Apitzsch et al. (2021) diskutiert das Schlusskapitel noch einmal, wenn es um den Beitrag der vorliegenden Untersuchung zum Verständnis der digitalen Transformation geht.

6.5.4. Folgen von Standardsoftware für die Arbeitsgestaltung der Softwareanwendung

Was sagt aber nun die Forschung konkret zu der Frage, welchen Unterschied es macht, ob Organisationen eine individuelle oder eine Standardsoftware gestalten? Welche Folgen einer Software auf die anwendende Organisation und die Anwendenden sind darauf zurückzuführen, dass sie individuell gestaltet ist? Dazu konnte keine Forschung gefunden werden. Dagegen ist die Forschung zu Standard-ERP-Software üppig. Sie zeigt zweierlei: Erstens gibt es Veränderung in der Softwareanwendung, die nicht auf den Standardcharakter zurückzuführen sind. Zweitens gibt es Veränderungen, die sich daraus ergeben, dass die Softwareanwendung eine Standardlösung einsetzt. Bei diesen Folgen gilt es allerdings in der Mehrzahl um jene einer fertigen Software und nicht um die Folgen eines Gestaltungsprozesses.

Zu den allgemeinen Folgen der Einführung von Standard-ERP-Lösungen, die nicht auf den Standardcharakter zurückgeführt werden können, gehören eine stärkere Automatisierung, standardisierte Arbeit und die Substitution von gering Qualifizierten durch höher Qualifizierte (vgl. Howcroft/Richardson 2012: 120). ERP-Systeme wirken allgemein disziplinierend auf Arbeit: durch Intensivierung, Arbeitsplatzabbau oder Integration jedes Arbeitsplatzes in ein IT-System (vgl. Dery et al. 2006: 207f.). Nach ihrer Einführung entstehen neue Tätigkeitsfelder, Berufsfelder und Abteilungen, die für die Koordination der Dateneingabe und -pflege zwischen allen involvierten Abteilungen wie IT, Logistik oder Einkauf zuständig sind (vgl. Walker 2016: 83f.). Die Zentralisierung des Managements durch ERP-Software kann das Problem mit sich bringen, dass z.B. dezentrale Mitarbeitende einer Organisation keine Lösungen erarbeiten können, weil das IT-System dafür zu unflexibel ist (vgl. Schwarz/Brock 1998: 76). Zu den im Absatz genannten Effekten kann aber auch eine individuell für eine Organisation entwickelte ERP-Software führen.

Es gibt aber einige Folgen, die sich auf den Standardcharakter zurückführen lassen, auch wenn das Verhältnis zwischen Software und Organisation von Fall zu Fall unterschiedlich ist. Standardsoftwarepakete wie die von SAP, die viele Einstellungs- und Anpassungsmöglichkeiten bieten, werden für Mormann zum Bezugspunkt der Arbeitsgestaltung.:

> »Die standardisierte Software fungiert als Referenzobjekt, mit dessen Hilfe sowohl über die Anpassung der Software als auch über die Anpassung der Organisation etwas ausgesagt werden soll.« (Mormann 2016: 150)

Laut ihr re-modellieren anwendende Organisationen der Standardsoftware ihre Arbeitsabläufe. Nur bei größeren unternehmerischen Konflikten passen die Organisationen die Software stärker an. So geschehen in einem Fall, bei dem Beschäftigte verhindert haben, dass Standorte in puncto Kennzahlen einfach verglichen werden können (vgl. Mormann 2016: 222). Eine andere Autorin kommt zu dem Schluss, dass Standardsoftware dazu dienen kann, Reorganisation in der anwendenden Organisation durchzusetzen (vgl. Hohlmann 2007: 333). Eine weitere Untersuchung zeigt, dass mehrere Projektzyklen notwendig waren, um die bei der ersten Einführung bereits umgesetzten, großzügigen Anpassungen am Standard zu reduzieren und näher an den Standard der Software zu rücken (in diesem Fall geht es um das ERP-System von Oracle). Die ERP-Implementierung erscheint als wiederkehrende, andauernde Aushandlung (vgl. Svejvig/Jensen 2013). Die Anpassung der Organisation an den Standard schlägt sich in der Partizipation der Anwendenden nieder. Bei einer SAP-Einführung nahmen während des Einführungsprojekts die Mitarbeitenden lediglich am Training teil und durften die Anwendung testen. Gestalten durften sie nicht (vgl. Tsamenyi et al. 2006: 425f.). Andere Studien sprechen von »marginalizing the users« im Verlauf einer ERP-Implementierung (vgl. Lyytinen und Newman 2015). In anderen Fällen ist die Anpassung gescheitert. Die Firmen haben die Implementierung abgebrochen, weil die Software wichtige Anforderungen nicht erfüllen konnte. Das Fazit der Autoren dreier Fallstudien ist, dass Firmen Effizienzen nur gehoben haben, wenn ihre bestehenden Praktiken zur Software passten (vgl. Grant et al. 2006: 13). In einem anderen Projekt bedeutete die Implementierung einer Standardsoftware, dass die Implementierung die vorher schon bestehende fehlende Integration von Abteilungen fortgeschrieben hat. Damit hat die Organisation Ziele wie eine zentrale Koordination durch die ERP-Software nicht erreicht (vgl. Elbanna 2007). Andere Autoren und Autorinnen sehen, sobald die ERP-Software implementiert ist, die Möglichkeiten eines starken Wandels von Software und Organisation als begrenzt an. Sie sprechen von »flüssige[m] Beton« (Brödner 2002 nach Remer 2008: 42) oder »enger Kopplung« (Hohlmann 2007: 34f.). Ein gelieferter Standard gilt aufgrund des Lock-in-Effekts als innovationshemmend, weil er erschwert, die Softwarezulieferfirma zu wechseln (vgl. Hohlmann 2007: 334, Zhu und Zhou 2012). Haben sich viele Organisationen an eine Standardsoftware angepasst, ist eine Verlagerung von Arbeit zwischen diesen leichter. Das ist eine weitere Veränderung in der Anwendung, auf welche die Forschung hinweist: Aufgrund von Standardisierung von Aufgaben und Fertigkeiten durch die Standardsoftware sind diese einfacher von einem lokalen Kontext zu lösen und zu verlagern (vgl. Howcroft/Richardson 2012: 124). Wer SAP in Firma X angewendet hat, kann es auch in Firma Y tun.

Neben den Fragen der Anpassung und der Flexibilität von Organisationen bei und nach Einführung einer Standard-ERP-Software geht es um die Veränderungen innerhalb der anwendenden Organisation, die nichts mit der reinen ausführenden Anwendung zu tun haben. Hohlmann (2007) stellt fest, dass nach der Einführung von SAP Gestaltungsnetzwerke innerhalb von Organisationen entstehen. Sie sind dafür da, den Standard anpassen zu können (siehe 6.3).

Letztendlich hat die Forschung vor allem untersucht, ob und wie sich eine Organisation einem fertigen Standardprodukt unterordnet. Nur Hohlmann geht ausführlich auf Strukturen ein, die dafür da sind, Software über die Implementierung hinaus zu ge-

stalten. Um solche Strukturen geht es auch beim Verhältnis von Softwaregestaltung und Softwareanwendung. Es geht um den Arbeitsprozess der Softwaregestaltung und weniger um dessen Produkt und wie es auf Arbeit und Organisation wirkt. Wenn es in den Fallstudien um Fragen von Partizipation, Reorganisation und Anpassbarkeit von Software geht, dann immer in Bezug auf eine noch zu gestaltende Individual- oder Standardsoftware. Es geht um den Arbeitsprozess der Softwaregestaltung und sein Verhältnis zum Arbeitsprozess der Softwareanwendung. Das kann dann auch einzelne Anwendungsbereiche in einem EVU betreffen und nicht gleich eine gesamte Organisation, wie dies bei der Einführung eines ERP-Systems wie jenem von SAP der Fall ist.

6.6. Zwischenfazit: Softwaregestaltung als soziotechnische Netzwerkarbeit und soziotechnische Arbeitsgestaltung

Das Kapitel hat Forschungsliteratur herangezogen, um die beiden Kernfragen der Arbeit konzeptionell einzubetten. Die erste Kernfrage nach der Kontrolle der Softwaregestaltung in unterschiedlichen Konstellationen adressiert das Konzept der soziotechnischen Netzwerkarbeit. Es stellt die konzeptionelle Lösung des Transformationsproblems im Arbeitsprozess der Softwaregestaltung dar. Für die zweite Kernfrage nach den Folgen der Softwaregestaltung für die Softwareanwendung konnte die Literatur nur dreierlei zeigen: (1.) Wie sich der Ansatz, die Folgen der Softwaregestaltung in ihrem Verhältnis zur Softwareanwendung zu untersuchen, von jenem der Informatisierung und des Informationsraums unterscheidet. (2.) Das Verhältnis von Softwaregestaltung und Softwareanwendung ist eines der Rationalisierung. (3.) Welche Folgen für die Softwareanwendung sich auf den Standardcharakter von Software zurückführen lassen.

Die analytische Grundlage für die **soziotechnische Netzwerkarbeit** sind weder Markt noch Hierarchie, sondern das Netzwerk. Mit ihm lassen sich auf verschiedene Organisationen, Teams oder Abteilungen verteilte Arbeitsprozesse besser analysieren, bei denen Wissen, Kooperation und Software zentral sind. Der hier verwendete Netzwerkbegriff folgt Ansätzen von Sydow/Windeler (2000), Apitzsch (2006) und Kalkowski/Mickler (2015), die Netzwerke als aus mehreren Ebenen bestehend auffassen.

(IT-)Projekte sind Beispiele für soziotechnische Netzwerkarbeit, weil auch bei ihnen die vier Ebenen aus Ablauf, Beziehungen, Software und Wissensarbeitenden für die Kontrolle der Arbeit sorgen (Heidling 2018, Ford/Randolph 1992, Kalkowski/Mickler 2005). Hohlmann (2007) zeigt, wie nach ERP-Einführungsprojekten Gestaltungsnetzwerke innerhalb der anwendenden Organisationen entstehen. Es existieren Abläufe, um weiter abteilungsübergreifend zusammenzuarbeiten und die Standardsoftware anzupassen, einzustellen und zu erweitern. Die Beziehungen zwischen den einzelnen Beteiligten des Gestaltungsnetzwerks untergraben alte Hierarchien und sind interdisziplinär. Die Software in Form des ERP-Systems wird für die Arbeit in den Organisationen zentral und das Wissen über die Software geht nun Hand in Hand mit dem jeweiligen branchen- und organisationsspezifischen Fachwissen. Die Softwaregestaltenden vermitteln zwischen Software und Fachabteilung. So wenden z.B. Key User:innen Software nicht nur an, sondern nehmen darüber hinaus Einstellungen an ihr vor.

Softwaregestaltung findet aber nicht immer innerhalb von Organisation statt und es geht nicht immer darum, eine Standardsoftware anzupassen. Zudem wird nicht immer Projektarbeit als Methode eingesetzt (mittlerweile weit verbreitet: Scrum). Deshalb ging es im weiteren Verlauf des Kapitels darum, allgemein notwendige Bedingungen der Kontrolle von Softwaregestaltung für drei der vier Ebenen der soziotechnischen Netzwerkarbeit herauszuarbeiten: Beziehungen, Software und Softwaregestaltende. Für die Ebene des Ablaufs wurden neben der Literatur zu IT-Projektarbeit keine theoretischen, passenden Bezüge gefunden.

Soziotechnische Netzwerkarbeit basiert unabhängig von der jeweiligen Konstellation auf kooperativen Beziehungen, die aber u.a. aufgrund von Marktbeziehungen, Hierarchien und Abteilungsgrenzen nicht selbstverständlich sind. Die Forschung zeigt, dass eine IT-Abteilung nicht automatisch kooperativ mit den Fachbereichen zusammenarbeitet. Regelmäßige Kooperation, Kommunikation und interdisziplinäres Wissen auf beiden Seiten sind Voraussetzungen dafür (vgl. Reich/Benbasat 2000, Chan/Reich 2007, Masak 2006, Schlosser et al. 2015, Valorinta 2011, Ko/Kirsch 2017). Je nach Organisationsformen der IT-Abteilung arbeitet diese mehr oder weniger eng mit den Fachabteilungen zusammen (vgl. Guillemette & Parè: 2012), ist sie mal mehr und mal weniger hierarchisch und zentralisiert (vgl. Sesay/Ramirez 2016, vgl. Masak 2006: 209) oder marktförmig (von Jouanne-Diedrich et al. 2005) organisiert. Dabei gibt es Forschung, die zeigt, dass Unternehmen, die komplexe IT-Wissensarbeit auslagern, dennoch in die interne IT-Organisation investieren müssen, damit die Zusammenarbeit funktioniert (vgl. Tiwana/Kim 2016).

Die soziotechnische Netzwerkarbeit muss mit unterschiedlichen Interessen und ungleicher Wissensverteilung und den daraus resultierenden Konflikten zurechtkommen. Ungleichgewichte entstehen u.a. dadurch, weil es bei der Auslagerung von IT-Arbeit in der auslagernden Organisation zu Kompetenzverlust, Kontrollverlust, einem Abbau des organisationalen Lernens oder der Innovationskapazitäten kommt (vgl. Miozzo/Grimshaw 2005: 1424). Es kommt zur Machtverschiebung zu IT-DL (vgl. Peled 2001, Flecker/Holtgrewe 2008: 314). Es kann zu Machtkämpfen zwischen IT-Dienstleistenden und den Auftraggebenden bzw. der Konzernzentrale kommen (vgl. Mezihorak 2018: 825ff.). Auch dann, wenn sie über weniger Wissen verfügen, müssen anwendende Unternehmen Wege finden, externe IT-Organisationen zu kontrollieren (vgl. Kaniadakis 2012: 270). Auch aufgrund der unterschiedlichen Konstellationen und Konflikte kann es für die Zusammenarbeitet von Organisationen unterschiedliche Steuerungsformen geben (vgl. Helfen/Wirth 2020: 14ff.). Die Forschung weist darauf hin, dass sie durch Lernprozesse entstehen, wofür Kapazitäten zum Wissensaustausch zwischen Firmen notwendig sind (vgl. Mola et al. 2017: 1293, vgl. van Fenema/Keers/Zijm 2014: 205). Auch innerhalb von Firmen kann es zu Machtkämpfen zwischen interner IT und Fachabteilungen kommen (vgl. Symon 2000: 400ff.). Dabei hat die IT-Abteilung eigene Machtquellen (vgl. Silva 2005: 56). Aber letztendlich bietet die Einführung und Entwicklung von Software für alle Beteiligten Möglichkeiten zur Mikropolitik (vgl. Ortmann et al. 1990).

Nicht nur die Beziehungen zwischen Organisationen sind Teil der soziotechnischen Netzwerkarbeit. Auch jene zwischen Personen sind es. Sie sind entscheidend, weil für die kooperative Zusammenarbeit in einem Netzwerk z. B. Projektstrukturen allein nicht aus-

reichen. Kooperationsbereitschaft, Sozialkompetenz, breit verankerte Verantwortungsbereitschaft und verändertes Führungsselbstverständnis/-verhalten müssen vorhanden sein (vgl. Rüegg-Stürm/Young 2001). Bolte und Porschen nennen Methoden, um informelle Strukturen der Kooperationen im Arbeitsalltag zu etablieren: Job Rotation, Hospitation, Promotoren oder Trainee-Programme für Einsteiger:innen (vgl. Bolte/Porschen 2007). Zudem ist es wichtig, dass langfristige Beziehungen bestehen, weil Vertrauen erst mit der Zeit aus reziproken Beziehungen und durch geteilte Interessen entsteht (vgl. Powell 1990, Uzzi 1997). Dabei ist das in Kooperationsnetzwerken schwierig: Denn es besteht in ihnen generell ein geringes Vertrauen und eine geringe Loyalität (vgl. Grimshaw et al. 2002: 200, Brinkmann/Dörre 2006: 139, Howcroft/Richardson 2012: 122, Holtgrewe 2014: 17ff.). Schnell entsteht Misstrauen und Wissen wird nicht mehr geteilt (vgl. Hirschfeld 2000: 277, 268) – insbesondere bei marktförmigen Beziehungen (vgl. Felin/Zenger/Tomsik 2009: 557).

Software beschränkt ihre Kontrolle in der soziotechnischen Netzwerkarbeit nicht auf Standardisierung, Formalisierung und Überwachung. Vielmehr ist bei der Softwaregestaltung ihre koordinierende, kooperationsermöglichende und wissensaktivierende Funktion charakteristisch. Software verlangt von den Anwendenden wissensevozierendes Engagement und Interaktion (vgl. Darr 2019, Rennstam 2012). Zudem koordiniert sie als gemeinsam gestaltbares, textbasiertes Bezugsobjekt die Wissensarbeit (vgl. Nicolini/Mengis/Swan 2012, vgl. Barrett/Oborn 2010, Carugati et al. 2018, Ponte/Rossi/Zamarian 2009, Bolici/Howison/Crowston 2009, 2016). Technische Kontrolle im Sinne Edwards (1981) ist bei der Softwaregestaltung sekundär. Literatur zur Softwareentwicklung zeigt, was auch für die Softwaregestaltung gilt: Menschliche Interventionen machen den digitalen Arbeitsprozess aus (vgl. Andrews et al. 2005, Barrett 2005). Gleichzeitig schränkt Software die Handlungsmöglichkeiten ein. Anwendende müssen der Programmlogik folgen (vgl. Degele 2000: 67f.) und sich an die in der Software fixierten Regeln halten (vgl. Heidenreich/Kirch/Mattes 2008: 7). Sie kann Arbeitsschritte als alternativlos vergeben, sie detaillierter und umfassender verregeln, vorstrukturieren und systematisieren (vgl. Schaeffer/Funken 2008: 13f.). Zudem verfestigt Software soziale Strukturen wie Rollen oder Routinen soziotechnisch, indem sie diese in Software abbildet (vgl. Mutch 2010, Volkoff et al. 2007). Sie gliedert den Einzelnen in einen Prozess ein (vgl. Sauer 2018: 196). Wobei es dann nicht immer nur um einfachen Input gehen muss, sondern auch darum, komplexere Aufgaben zu erledigen (vgl. Kleemann/Matuschek 2008). Zuletzt kontrolliert Software durch Transparenz: ob durch Zuboffs »electric panopticum« (1988), die organisationale Dezentralisierung durch die zentrale Steuerung dezentraler Einheiten via Kennziffern (vgl. Kocyba 1999) oder die Teamsteuerung über Kennzahlen (vgl. Boes et al. 2018).

Software ist aber nicht nur aufgrund ihrer Bedeutung für die Kontrolle eine der vier Ebenen der soziotechnischen Netzwerkarbeit. Zusätzlich prägt die Software die Kommunikation. Theoretischer Ausgangspunkt dabei ist Conways Law (1968), laut dem sich organisationale Kommunikationsstrukturen in der Software spiegeln. Neue Literatur weist darauf hin, dass es bei der Softwaregestaltung umgangen werden kann und trotzdem noch prägend ist (vgl. Colfer/Baldwin 2016, Hvatum/Kelly 2005). Das betrifft jedoch die Aufteilung einer Software z.B. in Module. Abgesehen davon können Abhängigkeiten, die durch den softwaretechnischen Zuschnitt entstehen, die Kommunikation prägen.

Wenn z.B. mehrere Teams mit einem gemeinsam genutzten (Standard-)Softwarebaustein arbeiten, müssen sie sich für die Gestaltung dieses Bausteins abstimmen (vgl. Remer 2008).

Die vierte Ebene des Netzwerks sind die Softwaregestaltenden, die zwischen Anwendung und Programmierung tätig sind. Als Wissensarbeitende sind die Softwaregestaltenden Teil der zunehmenden Zahl an höherqualifizierten Dienstleistungsarbeitenden in mittleren und höheren Dienstleistungsjobs, die zur Polarisierung der Qualifikationsstruktur beitragen (vgl. Overbeck 2017). Sie sind Träger der soziotechnischen Netzwerkarbeit, indem sie rollenbasiert Erwartungen erfüllen (vgl. Bechky 2006, vgl. Schimank 2010). Allgemeine Erwartungen unabhängig von spezifischen Rollen wie IT-Projektleitung oder Anforderungsmanagende sind: kooperativ zu sein (vgl. Williams 2002), selbstorganisiert zu arbeiten (vgl. Pongratz/Voß 1997, Sauer 2018, Lee/Edmondson 2017, Minssen 1999, Faraj/Sproull), aktiv mit Softwareobjekten zu interagieren und sich ausgehend von ihnen zu koordinieren (vgl. Rennstam 2012, Koblauch 1996, Zuboff 1988, Conrad 2012), mit Nicht-Wissen umgehen zu können (Schmiede 2006, Wilke 1998, Schulz-Schaeffer 1999) und sich im Netzwerk zu bewegen (vgl. Hohlmann 2007, Flecker/Holtgrewe 2008, Funken/Stoll/Hörlin 2011). Subjektivierung (vgl. Minssen 2011) und subjektivierendes Arbeitshandeln (vgl. Böhle 2010, Bolte 2017a, Bolte 2017b, Weishaupt/Hösl 2017) sind Kernbestandteile der Kontrolle ihrer Arbeitskraft.

Softwaregestaltende sind eine Gruppe, die versucht, Einfluss in einer Organisation zu gewinnen. Sie helfen, Kernprobleme des Kapitals (wie z.B. die Kontrolle von Arbeit) zu lösen, und können deshalb eine bestimmte Position für sich beanspruchen (vgl. Armstrong 1985). Ihre Position ergibt sich technikinduziert, weil sie notwendig sind, um die Software-Technologie einzusetzen, wie bspw. die Rolle von Key User:innen (vgl. Hohlmann 2007). Das empirische Kapitel wird zeigen, ob sie und ihre Qualifikationen tatsächlich Teil des Managements sind, welchen Teil des Managements sie ersetzen oder mit welchem Teil des Managements sie konkurrieren.

Die Ebenen und ihre Dimensionen der soziotechnischen Netzwerkarbeit zur Softwaregestaltung zwischen Anwendung und Programmierung im Überblick:

Tabelle 4: Ebenen und Dimensionen der soziotechnischen Netzwerkarbeit

Ebene	Dimension
Ablauf	Situativ Verhältnis von formalen und informellen Arbeitsschritten etablieren
	Gestaltungsnetzwerke und deren Rollen und interdisziplinäres Wissen einbeziehen
Beziehung	Kooperative Beziehung zw. IT-Abteilung/-Dienstleistungsfirma und Fachabteilungen
	Organisationsübergreifende Steuerungsstrukturen und kooperativer Umgang mit Interessengegensätzen und ungleichen Wissensständen
	Interpersonale Beziehungen basierend auf Vertrauen, Reziprozität, Kooperation

Ebene	Dimension
Software	Kontrolle: ermöglichend und einschränkend
	Strukturprägend durch ihre Architektur
Softwaregestaltende	Wissensarbeitende Vermittelnde zwischen Anwendung und Programmierung
	Rollen: erwartungsgeleitetes Handeln
	Erhalten Einfluss, weil sie eine Technologie einsetzen helfen

Die Diskussion über die flexible Ergänzung der vier Ebenen in Abschnitt 6.4.4 sollte verdeutlichen, dass es keine Best Practice dafür gibt, wie Ablauf, Beziehungen, Software und die Rollen der Softwaregestaltenden auszugestalten sind und Kommunikation stattfindet. So kann ein Common Ground aus Wissen und softwarebasierten Werkzeugen oder eine modulare Softwarearchitektur direkte Kommunikation ersetzen (vgl. Srikanth/Puranam 2014). Statt über den einen besten Weg zu kommunizieren, kommen vielfältige Kommunikationskanäle zum Einsatz (vgl. Heidenreich/Kirch/Mattes 2008). Es gibt Fälle, in denen direkte Kommunikation durch digitale Kommunikation (Chats, Ticketsysteme, online geteilte Dokumente) ersetzt werden kann (vgl. Shaikh/Henfridsson 2017). Zudem ist es abhängig von der Softwareentwicklungsplattform, wie ihre Nutzenden Wissensgrenzen überwinden (vgl. Foerderer et al. 2019).

Die Folgen der Softwaregestaltung auf die Softwareanwendung konzipiert die Untersuchung als **soziotechnische Arbeitsgestaltung** der Softwareanwendung durch die Softwaregestaltung. Diese soziotechnische Arbeitsgestaltung als Verhältnis zweier Arbeitsprozesse folgt Ansätzen einer inkrementellen Digitalisierung (vgl. Hirsch-Kreinsen 2020) im Sinne einer immer weiteren Ausbreitung von Software und einer Digitalisierung durch Arbeit und ihrer je arbeitsweltspezifischen Ausprägung (vgl. Apitzsch et al. 2021). Es handelt sich um eine Rationalisierung durch Technikgestaltung, was als technikentwicklungsbezogene Strategie die drei anderen von Menz/Nies/Sauer (2019) genannten vervollständigt. Indem Software für die Organisationen wichtiger wird, ihre Ziele zu erreichen, wird Softwaregestaltung für die Effizienz einer Organisation wichtiger. Das Konzept der soziotechnischen Arbeitsgestaltung arbeitet dann erst das Empirie-Kapitel detailliert aus.

Denn die Literatur zu den Folgen der Softwaregestaltung auf die Softwareanwendung betrifft nur Standardsoftware und liefert auch kein brauchbares Konzept, um das in den Fallstudien aus dem 8. Kapitel vorliegende Verhältnis von Softwareanwendung und Softwaregestaltung zu beschreiben. Was sind aber die Folgen von Standardsoftware für die Softwareanwendung? Eine Standardsoftware macht die Anwendung weniger flexibel in der Arbeitsgestaltung (vgl. Brödner 2002 nach Remer 2008: 42, Hohlmann 2007: 34, 334, Zhu/Zhou 2012). Sie kann als Durchsetzungsinstrument von Reorganisationen dienen (vgl. Hohlmann 2007: 333), re-modelliert Arbeitsabläufe und nur bei größeren unternehmerischen Konflikten passen die Organisationen die Software stärker an (vgl. Mormann 2016: 222). Andere ERP-Implementierungen sind wiederkehrende, andauernde Aushandlungen (vgl. Svejvig/Jensen 2013), werden abgebrochen, weil die Software wichtige Anforderungen nicht erfüllt (vgl. Grant et al. 2006: 13), oder schreiben die

vorher bestehende fehlende Integration von Abteilungen fort (vgl. Elbanna 2007). Zudem entstehen langfristige Gestaltungsnetzwerke für ERP-Software innerhalb von Organisationen (Hohlmann 2007). Ebenso stellen sich auch bei der Einführung einer Standardlösung Fragen der Partizipation der Anwendenden (vgl. Tsamenyi et al. 2006: 425f., Lyytinen/Newman 2015). Zuletzt erleichtert Standardsoftware es, Tätigkeiten von einem lokalen Kontext zu lösen und zu verlagern (vgl. Howcroft/Richardson 2012: 124). Andere Veränderungen lassen sich nicht auf den Standardcharakter zurückführen, wie zum Beispiel neue Tätigkeitsfelder, Berufsfelder und Abteilungen für die Koordination der Dateneingabe und -pflege (vgl. Walker 2016: 83f.); Automatisierung, Standardisierung, Dequalifizierung in der Anwendung (vgl. Howcroft/Richardson 2012: 120); Intensivierung, Arbeitsplatzabbau oder Integration jedes Arbeitsplatzes in ein System (vgl. Dery et al. 2006: 207f.); Zentralisierung des Managements einhergehend mit Handlungseinschränkungen für dezentrale Teams (vgl. Schwarz/Brock 1998: 76).

Bevor die Fallstudien die unterschiedlichen Aspekte der soziotechnischen Netzwerkarbeit und der soziotechnischen Arbeitsgestaltung konkretisieren, stellt das nächste Kapitel die Energiewirtschaft vor und beschreibt, wie sich die Industriestrukturen auf die Softwaregestaltung auswirken.

7. Industriespezifische Aspekte der Softwaregestaltung in der Energiewirtschaft

Eine Folge der Digitalisierung einer Branche ist, dass eine neue Arbeitsteilung existiert: jene zwischen Anwendung und Entwicklung von Software. Dabei ist die Arbeitsteilung geprägt durch Industriestrukturen, die das folgende Kapitel durch das Konzept der Industrie-Governance beschreibt. Letzteres dient dazu, in die Energiewirtschaft einzuführen und Anwendungsfelder und Treiber der Softwareentwicklung kennenzulernen. Die verschiedene Anwendungsfelder ergeben sich in erster Linie aus den unterschiedlichen Geschäftsfeldern der EVU und aufgrund der umfangreichen Regulierung der Branche durch den Staat. Energiewende und Liberalisierung sind wichtige Treiber für eine kontinuierliche (Weiter-)Entwicklung von Software. Charakteristisch für die Branche sind zudem die vielen Kooperationen zwischen den EVU und ein vielfältiger Markt an Zulieferfirmen für Software.

7.1. Industriestrukturen der Energiewirtschaft und ihr Verhältnis zur Digitalisierung

7.1.1. Ansatz der Industrie-Governance

Ob die Re-Regulierung der Energiewirtschaft seit 1998 oder der zunehmende Einsatz digitaler Technologien: Beide haben gravierende Folgen für die »Regulierung von Arbeit, Auswirkungen auf die Arbeitsgestaltung ebenso wie auf die Handlungsspielräume und Machtressourcen arbeitspolitischer Akteure« (Jürgens 2007: 119). Ebenso wie dies Jürgens 2007 für die Automobilindustrie konstatierte, gibt es aktuell in der Energiebranche Unsicherheit »hinsichtlich der Nachhaltigkeit neuer Struktur- und Strategiekonzepte« (ebd.). Das industriespezifische Ordnungsmodell ergibt sich, wie das Kapitel zeigen wird, im Wechselspiel aus politischer Regulierung und digitaler Technologie. Software ist Teil davon und prägt unabhängig von einzelnen Organisationen unternehmensübergreifende Struktur- und Handlungsmuster. Für die Analyse wird das Konzept der Industrie-Governance genutzt, das sowohl wirtschaftliche Strukturen als auch staatliche

Regulierung im Blick hat, wie zuletzt von Jürgens (2007) beschrieben. Es geht um Struktur- und Handlungsmuster, die unternehmensübergreifend gültig sind. Beziehungen zwischen Agierenden und Legitimationsfragen sind zentral. Das Konzept der Industrie-Governance geht über Fragen der Regulierung und der Produktion hinaus. Es geht darum, wie einzelne handelnde Personen zentrale Probleme lösen und welche Governance-Mechanismen sie anwenden. Ein solches Problem ist bspw., wie Wachstum geschaffen werden kann. Für die großen Konzerne hat es Chandler (1990) idealtypisch beschrieben. In den vertikal integrierten Unternehmen beruht es auf Massenproduktion (»scale«) und Diversifizierung (»scope«), wobei sie immer neue Geschäftsfelder erschließen (vgl. Jürgens 2007: 122). Eine andere Form ist im sogenannten Wintelismus (eine Wortschöpfung aus Windows und Intel) in der IT-Branche zu finden, wo Unternehmen unprofitable Bereiche abstoßen (es keine Querfinanzierung gibt) und Wachstum durch Fokus auf Kernkompetenzen verfolgen. Neue Märkte werden durch Firmenzukauf erobert und der Umsatz durch kürzere Produktzyklen gesteigert (vgl. Jürgens 2007: 124). An diesen beiden skizzierten Beispielen lässt sich bereits erkennen, dass nicht nur Unterschiede zwischen Ländern existieren, wenn es um eine wirtschaftliche Ordnung geht (wie der Ansatz der *Varieties of Capitalism* (Hall/Soskice 2001) es vorführt). Auch Industrien unterscheiden sich untereinander und verändern sich im Laufe der Zeit.

Um die Darstellung der Energiewirtschaft zu strukturieren, unterscheidet der Industrie-Governance-Ansatz vier Arenen (vgl. Jürgens 2007):

1. Corporate Governance: betrifft die Führung eines Unternehmens (z.B. grundsätzliche Ziele oder für was Unternehmen Gewinne verwenden)
2. Produktmarkt: Normen und Regelungen (inkl. Standards) von Staat und Unternehmen hinsichtlich Produkteigenschaften und Wettbewerbsstrukturen
3. Prozess: Ausgestaltung der interorganisationalen Beziehungen in den Wertschöpfungsketten (z.B. fokale Unternehmen, Outsourcing, Spezialisierung durch Aufteilung von Produktentwicklung und Fertigung, Fragmentierung, Rolle der Zulieferer, Etablierte vs. Neulinge)
4. Arbeitsbeziehungen: z.B. Standardisierung von Arbeit, Bedeutung von Erfahrungswissen, Organisationsgrad der Arbeitnehmer

Ausgeschlossen von der Analyse sind die Zulieferfirmen für Netze oder Kraftwerke (ABB, Hitachi, Siemens, Enercon, Yingli Solar, SMA Solar, Vestas, General Electric etc.) und Projektierende und Planende von Erzeugungsanlagen (Fichtner, Juwi etc.). Um die Darstellung zu vereinfachen und weil in den Fallstudien der Regulierung, neuen Geschäftsfeldern und allgemein in der zukünftigen Energiewirtschaft Strom eine große Bedeutung zukommt, wird im Folgenden der Schwerpunkt auf die Governance der Stromwirtschaft gelegt. Gas, Atomkraft, Öl oder Kohle bleiben außen vor.

Bevor das Kapitel die einzelnen Arenen detailliert behandelt, sei noch auf eine wichtige Besonderheit der Energiewirtschaft hingewiesen. Zentral für ihr Verständnis ist die Rolle des Staates. Er reguliert den gesamten Produktionsprozess. Er entscheidet mit über Gewinnchancen und Innovationsmöglichkeiten. Im Energiebereich nimmt die Governance ihren Ausgang in der EU und der Idee, einen Binnenmarkt für Energie zu etablieren. Das Wettbewerbsprinzip soll gelten. Durch die Umsetzung der EU-Vorgaben

löst sich das vor 1998 noch existierende »korporatistisch geregelte[…] Oligopol mit regionalem Gebietsmonopol« (Sack 2018: 83) in Deutschland auf. Bontrup und Marquardt fassen die Liberalisierung wie folgt zusammen:

> »Auf die erste Phase eines eher großzügigen Laisser-faire, der auf die Selbstdisziplinierung des Markts bzw. auf Verbändevereinbarungen fast schon blind vertraute, folgte eine Periode der extensiven Neugestaltung des Regulierungsrahmens für die Elektrizitätswirtschaft.« (Bontrup/Marquardt 2010: 23)

Die als effizient geltende Markt-Governance setzt einen Verwaltungsaufbau voraus (vgl. ebd: 72). Ein Autor konstatiert ein »ausuferndes Regulierungsrecht sowie einen erheblichen Behördenzuwachs« (Schöneich 2012: 79).

Wichtige Agierende in der Energiewirtschaft sind zudem die Verbände. Ob EEG oder Vorgaben der Bundesnetzagentur: Die Verbände BDEW (Bundesverband der Energie- und Wasserwirtschaft) und VKU (Verband kommunaler Unternehmen) liefern Anwendungshilfen für die Regulierung. Ob Verordnungen, Gesetze oder Datenformate: Es werden immer Dritte in sogenannten Konsultationsphasen hinzugezogen. In den Konsultationsphasen können Anmerkungen gemacht werden[1]. Unternehmensverbände machen zudem Studien zu neuen, digitalen Technologien und wie diese in der Energiewirtschaft genutzt werden können. Sie wirken auf die Regulierung der Digitalisierung ein, bspw. bei den Smart-Meter-Konsultationen.[2]

7.1.2. Corporate Governance: Zwischen Daseinsvorsorge und Wettbewerb

Die besondere Rolle des Staates zeigt sich daran, dass die Kernziele der Stromwirtschaft durch die Politik vorgegeben sind. Alle Unternehmen der Energiewirtschaft unterliegen gleichermaßen dem gesellschaftlichen Auftrag, der gesetzlich in § 1 des Energiewirtschaftsgesetzes festgelegt ist: Ihr Sinn und Zweck ist demnach »eine möglichst sichere, preisgünstige, verbraucherfreundliche, effiziente und umweltverträgliche leitungsgebundene Versorgung der Allgemeinheit mit Elektrizität und Gas, die zunehmend auf erneuerbaren Energien beruht.«

Inwiefern sie in ihrem Management öffentlich kontrolliert sind, hängt von ihrer Eigenständigkeit von der staatlichen Verwaltung (7.1.2.1), der Entflechtung (7.1.2.2), den Eigentümern (7.1.2.3) und der Marktrolle und den Geschäftsfeldern (7.1.2.4) ab.

---

1 Ein Beispiel für den Konsultationskreis: Jener für die Anreizregulierung am 25.08.2005, der laut BNetzA-Webseite aus folgenden Organisationen bestand: 8KU (Kooperation der acht großen kommunalen Energieversorgungsunternehmen aus Leipzig, Hannover, Köln, Frankfurt a.M., Darmstadt, Mannheim, Nürnberg und München), Bund der Energieverbraucher e. V., VKU – Verband kommunaler Unternehmen e. V., Verbraucherzentrale Bundesverband e. V. und noch zehn weiteren Verbänden: https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Energie/Unternehmen_Institutionen/Netzentgelte/Anreizregulierung/1_KK_Einladung.pdf;jsessionid=ADF8AE92C4D64C672E8EF59F444AAF2E?__blob=publicationFile&v=1, abgerufen am 16.05.2023.

2 Der Vollständigkeit halber sei erwähnt, dass – wie auch in anderen Industrien – Verbände wie Bitkom oder das Forum elektronische Rechnung Deutschland (FeRD) aktiv darauf hinwirken, die digitale Vernetzung der Gesellschaft voranzutreiben.

7.1.2.1. Rechtsform und eigenständige Unternehmensführung

Nicht erst durch die Liberalisierung hat sich die Unternehmensorganisation in der Stromwirtschaft geändert. Die für kommunale Unternehmen typische Rechtsform Eigenbetrieb ging bereits vorher stark zurück. 1952 hatten sie noch 94 %, 1989 noch 57 % und 2009 nur noch 23 % der VKU-Mitglieder (vgl. Ambrosius 2012: 42). Die Daten des Statistischen Bundesamtes zu allen Elektrizitätsversorgern zeigen, dass insgesamt die GmbHs (GmbH oder GmbH & Co. KG) mit 1156 von 1444 Betrieben 2017 dominieren. Im selben Jahr liefen nur noch 87 Versorger als Eigenbetrieb.

Tabelle 5: Rechtsformen Elektrizitätsversorger 2000 und 2017

	Insgesamt	Einzelfirma	OHG	KG	GmbH & Co. KG	GmbH	AG	Genossenschaften	Eigenbetrieb	Verband	Sonstige
2000	925	24	12	11	39	483	105	37	209	3	2
2017	1444	13	12	10	214	942	95	43	87	2	26

(Quelle: Statistisches Bundesamt)

Die Rechtsform macht für die Unternehmensführung einen Unterschied. Eigenbetriebsverordnungen wie jene in Niedersachsen lassen bspw. ein Weisungsrecht für den Bürgermeister bzw. Gemeindedirektor erkennen. Für Brede ist das aber nicht mehr zeitgemäß: »Die Werksleitung braucht Freiraum für die Gesamtheit der Unternehmenspolitik.« (Brede 2012: 308) Statt bspw. Werksausschuss und Werkleitung (primäre Leitungsorgane), Kommunalparlament und Hauptverwaltungsbeamte:r (sekundäre Leitungsorgane) wie beim Eigenbetrieb stellen GmbHs ein eigenständiges Unternehmen dar, dessen Geschäftsführung die gesellschaftende Kommune bzw. Stadt bestellt. Die Geschichte der SW München zeigt eine lange, über Jahrzehnte gehende Diskussion über die Rechtsform. Man wollte flexibleren Gestaltungsspielraum, weniger politische Einflussnahme und verbesserte Effizienz durch Mitbestimmung (vgl. Bähr/Erker 2017: 284).

7.1.2.2. Unternehmensstruktur: Entflechtung der Unternehmen

Ein Teil der Liberalisierung war es, die bis 1998 vertikal integrierten Unternehmen zu entflechten. Die Netze sind zwar weiterhin ein natürliches Monopol, »vor- und nachgelagerte Wertschöpfungsstufen allerdings werden als wettbewerbsfähig erachtet« (Bräunig 2012: 421). Bis 2005 sollten die EVU rechnerisch, informationell, eigentumsrechtlich und operationell entflechtet sein (vgl. Bräunig 2012: 422f.). Um diese als »Unbundling« bezeichnete Reorganisation zu verbessern, wurde die BNetzA (Bundesnetzagentur) ins Spiel gebracht. Sie wacht darüber, dass die Netze diskriminierungsfrei zugänglich sind und öffentliche und private Unternehmen gleichbehandelt werden. Bei der informationellen Entflechtung geht es darum, dass die Informationen für

Netz und Vertrieb diskriminierungsfrei zugänglich sein müssen (bspw. Zählerstände). Rechnerisch ist ein Unternehmen entflechtet, wenn es die Bilanzen nach Geschäftsfeld getrennt aufführt (z.B. Netz und Vertrieb). Bei der operationalen Entflechtung geht es vor allem um die personelle Trennung auf Leitungsebene (vgl. Bräunig 2012: 426f.). Die Entflechtung kann durch unterschiedliche Organisationsmodelle wie Holding oder Tochter-/Muttergesellschaften umgesetzt werden (vgl. Bräunig 2012: 432). Die Entflechtungspflicht gilt nicht für EVU mit weniger als 100.000 Kund:innen (vgl. § 7 Abs. 2 EnWG bzw. § 7a Abs. 7 EnWG). Dies hat zur Folge, dass kleine Stadtwerke integrierte Betriebe bleiben. Als zusätzlich zu entflechtende Marktrollen kamen 2005 der Messstellenbetrieb und mit dem Digitalisierungsgesetz von 2016 der Smart Meter Gateway Administrator dazu (der das Zähler-Datenmanagement übernimmt).

Die organisatorische Besonderheit kommunaler Unternehmen ist, dass es sich oft um Querverbundunternehmen handelt. 2009 bewirtschafteten 42 % der VKU-Unternehmen neben Strom mindestens noch eine weitere Sparte wie Wasser, Gas, Wärme oder Entsorgung. Strom allein bieten nur 16 von 1369 VKU-Mitgliedern an (vgl. Gottschalk 2012: 58). Das alleinstehende Stromlieferunternehmen oder der alleinstehende Netzbetrieb sind also selten.

Das mittelgroße Stadtwerk von Heidelberg hat zum Beispiel neun Gesellschaften im Konzern: Stadtwerke Heidelberg Bäder, Stadtwerke Heidelberg Energie, Stadtwerke Heidelberg Garagen, Stadtwerke Heidelberg Netze, Stadtwerke Heidelberg Technische Dienste, Stadtwerke Heidelberg Umwelt, Stadtwerke Neckargemünd, Stromnetz Neckargemünd, Heidelberger Straßen- und Bergbahn[3]. Einzelne EVU wie die Stadtwerke München überschreiten nationale Grenzen und haben Windparks in Norwegen oder Frankreich (vgl. Stadtwerke München GmbH 2023: 33f.).

7.1.2.3. Eigentümerstruktur und Unternehmensziele

Mit der Liberalisierung ging eine Privatisierung öffentlichen Vermögens einher. Komplett privatisiert wurde aber nur ein kleinerer Teil der Stromversorgung. In seinem Datensatz von 820 kommunalen Unternehmen (oftmals Strom und Gas gemischt) konnte Sander ermitteln, dass mehr als die Hälfte (430 Unternehmen) noch zu 100 % in kommunalem Besitz waren. Von den verbleibenden 390 Unternehmen blieben 63 in kommunaler Hand, wobei entweder weniger als 25 % der Anteile verkauft wurden oder ein anderes kommunales Unternehmen Anteile über 25 % erwarb (vgl. Sander 2009: 10f.). Von den großen Konzernen ist der Eigentümerstamm bei E.ON und RWE[4] gestreut, EnBW gehört dem Land Baden-Württemberg und einigen Kommunen dort[5], Vattenfall dem schwedischen Staat[6] und Uniper gehört seit Dezember 2022 der Bundesrepublik Deutschland.

Schwintowski macht als wesentlichen Unterschied zwischen privaten und öffentlichen Unternehmen aus, dass »öffentliche Unternehmen typischerweise gewinnbringen-

3 https://www.swhd.de/unternehmen, abgerufen am 02.05.2023.

4 Ca. 1/4 der Aktien von RWE gehören Kommunen (vgl. Meves 2021).

5 https://www.enbw.com/unternehmen/investoren/aktie/#aktionaersstruktur, abgerufen am 02.05.2023.

6 https://group.vattenfall.com/who-we-are/about-us, abgerufen am 02.05.2023.

de Bereiche mit solchen bündeln [können], die defizitär wirtschaften« (Schwintowski 2012: 330). Bei den Unternehmenszielen wird von »citizen value« anstelle des »shareholder value« gesprochen (vgl. Bähr/Erker 2017: 371). Aber inwiefern öffentliche Eigentümer, öffentliche Kontrollstrukturen und öffentliche Ziele wirklich zur Realisierung öffentlicher Politik führen, ist umstritten. So »ist empirisch festzustellen, dass der öffentliche Einfluss auf die Unternehmenspolitik mit zunehmender Größe der Unternehmen sinkt« (Monstadt 2004: 91). Es geht mehr um Kosten und Preise und nicht mehr um störungsfreie Versorgung. Das Management gleicht jenem in der Privatwirtschaft (vgl. Edeling/Stölting/Wagner 2004: 155f.). Schäfer konstatiert, dass die Daseinsvorsorge Teil des allgemeinen Wertschöpfungsprozesses wird und statt Bedürfnisbefriedigung und Nutzenstiftung Gewinn und Profit in den Vordergrund rücken (vgl. Schäfer 2014: 31). Unterschiede zwischen großen Konzernen und Stadtwerken gibt es weiterhin, was auch an der lokalen Verankerung Letzterer liegt. Ob wirtschaftliche Betätigungen außerhalb des Gemeindegebiets gesetzlich zugelassen sind, ist je nach Bundesland unterschiedlich (vgl. Püttner 2012: 144).

> »Während kommunale Energieversorger jedoch häufig aufgrund von politischem Kalkül und bestehenden Gemeindeordnungen in ihren Expansionsbemühungen beschränkt sind, haben andere regionale und überregionale Player den Markt der energienahen Dienstleistungen längst für sich entdeckt und treten als starke Konkurrenz auf.« (Rödl und Partner 2017: 59)

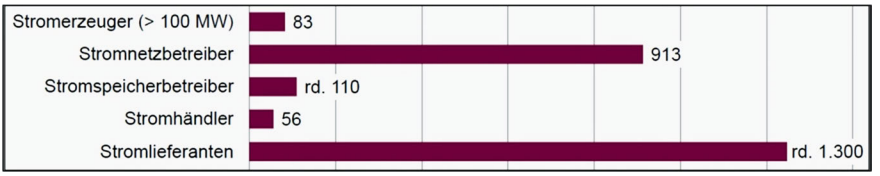
Neben diesen räumlichen Grenzen wirken sich die Vergaberegeln der EU einschränkend auf die wirtschaftliche Freiheit der kommunalen Unternehmen aus. Diese lässt nicht ohne weiteres zu, eine für die Leistungserbringung geeignet gehaltene Organisation auszuwählen (vgl. Brede 2012: 308). Gleichzeitig gilt die Selbstverwaltungsgarantie der Gemeinden laut Artikel 28(2) GG (vgl. Hellermann 2000: 350). Sie sichert den Kommunen die Selbstverwaltung zu. Die Selbstregelung lokaler Angelegenheiten und dezentrale politische Macht (Prinzip der Subsidiarität) ist ihnen gesetzlich garantiert (vgl. ebd. 21). Ob technischer Natur (in Form der Netze oder dezentraler Erzeugungsanlagen), im Sinne einer kommunalen Selbstverwaltung (Subsidiarität) (vgl. Edeling/Stölting/Wagner 2004: 20f.) oder durch Nähe zur Kundschaft, Dezentralisierung der Stromerzeugung und Nachhaltigkeit (vgl. Doleski 2016: 25f.): Die deutsche Energiewirtschaft hat einen starken regionalen Bezug. Eine aggressive Expansionspolitik oder kurzfristige Renditeorientierung ist nicht das Ziel kommunaler Unternehmen. Gleichwohl stehen regionale Versorger unter Druck, Gewinn zu erwirtschaften. Viele Kommunen finanzieren mit dem Geld andere Vorhaben ihrer Stadt.

Als Folgen der Liberalisierung haben sechs Feldstudien im Bereich Elektrizität (Belgien, Österreich, Polen, UK) festgestellt, dass der Wettbewerb von der lokalen Situation abhängt und im Vereinigten Königreich der intensivste Wettbewerb herrscht (vgl. Flecker/Hermann 2011: 526). Die EVU verfolgen neue Strategien: Diversifizierung, mehr Geld für Werbung, mehr Call-Center und damit reduzierte Betreuung vor Ort (vgl. ebd. 527).

7.1.2.4. Marktrollen, Geschäftsfelder und Regulierung

Die meisten stromwirtschaftlichen Unternehmen sind Netzbetreiber oder Stromlieferanten. 2018 gab es 913 Stromnetzbetriebe und rund 1300 Stromlieferanten in Deutschland (vgl. BDEW 2019).

Abbildung 7: Anzahl Elektrizitätsunternehmen 2018



(Quelle: BDEW 2019)

Firmen für Energiedienstleistungen wie Energieberatung, Energie-Contracting (Anlagen zur Verfügung stellen und betreiben) und Energiemanagement gibt es ca. 7400 (vgl. BfEE 2019: 103). Zusätzlich gibt es durch die erneuerbaren Energien viele kleine, private Erzeugungsanlagen von Genossenschaften, Landwirt:innen oder Privatleuten. Im Marktstammdatenregister[7] der BNetzA waren zum 28. April 2023 2.593.888 aktive Anlagenbetreibende aufgeführt.

Die Regulierung ist für jedes Geschäftsfeld unterschiedlich. Der Netzbetrieb ist viel stärker in seiner wirtschaftlichen Kalkulation von BNetzA-Verordnungen abhängig als ein Stromlieferant, dessen Herausforderung mehr im Wettbewerb mit anderen Lieferunternehmen liegt.

Bei den Netzen handelt es sich um natürliche Monopole. Es gibt Übertragungsnetzbetriebe (ÜNB) für die Fernübertragung, von denen es in Deutschland vier gibt (Tennet TSO, 50Hertz Transmission, Amprion, TransnetBW) und Verteilnetzbetriebe (VNB) für die letzten Kilometer bis zu den Verbrauchenden, von denen 913 existieren (siehe Abbildung oben) und die unterschiedlich groß sind. Die Übertragungs- und Verteilnetzbetriebe haben jeweils durch den Gesetzgeber zugewiesene Aufgaben (bspw. Stabilisierung des Stromsystems, Ausbau der Netze, Anschluss Erneuerbare-Energie-Anlagen). Um in dem natürlichen Monopol Anreize für Effizienzsteigerungen zu setzen, wurde 2007 die Anreizregulierungsverordnung verabschiedet[8]. Von nun an waren nicht mehr die Kosten Ausgangspunkt dafür, die Erlöse von Netzbetrieben zu bestimmen. Auch der Gewinn wurde nicht mehr pauschal aufgeschlagen. Es wurden nun individuelle Effizienzvorgaben ermittelt, von deren Erreichung abhängt, wie viel Rendite ein netzbetreibendes Unternehmen erwirtschaftet (vgl. Herrmann 2012: 289f.). Für Übertragungs- und Verteilnetzunternehmen liefert die Anreizregulierung einen Ansporn, die Effizienz zu steigern (bspw. durch Kooperation mit anderen Netzbetreiber, Erschließung neuer Geschäftsfelder, Einsatz moderner Technologien) (vgl. Herrmann 2012: 301). Laut einer Studie anhand von 109 VNB über die Anreizregulierung von 2009 hatte die Anreizregulierung ei-

7 https://www.marktstammdatenregister.de, abgerufen am 02.05.2023.

8 betreut durch die Beschlusskammer 8 der BNetzA, Verordnung StromNEV.

nen signifikanten positiven Effekt auf die Investitionsrate der VNB. Es stellte sich zudem heraus, dass es keinen Unterschied zwischen Firmen im öffentlichen und privaten Besitz gab (vgl. Cullmann et al. 2016).

Zwei Dinge gilt es noch zu bedenken: Für ca. 650 Stromnetzbetriebe (mit weniger als 30.000 Kund:innen) gibt es ein vereinfachtes Verfahren, bei dem die Berechnung der Anreizregulierung weniger aufwendig und somit kostengünstiger ist (vgl. Herrmann 2012: 295f.). Zum Zweiten müssen im Unterschied zu den ÜNB die VNB den Zuschlag für eine Netzkonzession erhalten, um wirtschaftlich tätig werden zu können. Wenn bspw. ein 20-jähriger Konzessionsvertrag ausläuft und der VNB den Zuschlag nicht mehr erhält, ist er seiner Geschäftsgrundlage beraubt und muss den Betrieb einstellen. Um die Konzession in Berlin zwischen Vattenfall und dem Land Berlin gab es gerichtliche Auseinandersetzungen, bis Vattenfall nachgab (vgl. Müller 2021).

Nicht nur im Netzbereich entstehen viele unternehmerische Risiken und wirtschaftliche Unsicherheiten durch die Regulierung oder politische Entscheidungen. Auch für den Messstellenbetrieb (den aktuell meistens das netzbetreibende Unternehmen übernimmt) gibt die BNetzA die Aufgaben vor: Installation, Inbetriebnahme, Konfiguration, Administration, Überwachung und Wartung von Zählern. Zusätzlich wurde im Zuge der Einführung von Smart Metering durch das Gesetz zur Digitalisierung der Energiewirtschaft (was die Installation von Smart Metern betrifft) eine neue Marktrolle bzw. ein neues Geschäftsfeld geschaffen. Der Smart Meter Gateway Administrator soll die Kommunikationseinheit für Messdaten zur Verfügung stellen und damit seine Geschäfte machen. Weil es so aufwendig ist, ein Zertifikat des Bundesamtes für Sicherheit in der Informationstechnik (BSI) zu erhalten, gibt es nicht viele Firmen, die diese Marktrolle übernehmen. Zudem begrenzt die Regulierung die Preise für Stromzähler für die Endkundschaft gestaffelt nach dem Stromverbrauch (für alle Verbrauchsstellen über 6000 kWh, bei denen der Einbau gesetzlich vorgeschrieben ist).

Die Stromlieferanten sind am stärksten am Markt ausgerichtet. Sie gab es vor der Liberalisierung noch nicht. Sie müssen sich um Marketing kümmern, Kund:innenbeziehungen pflegen und es gibt viel Konkurrenz (mittlerweile auch von branchenfremden Unternehmen wie Lidl[9], Deutsche Bahn[10] oder Volkswagen[11]).

Subventionierung Erneuerbare Energien

Erneuerbare Energieträger wurden vor allem zu Beginn durch ein Umlageverfahren staatlich gefördert – und zwar nicht nur für (große) Firmen, sondern auch für Privatpersonen. Die ökonomischen Anreize in Form von Fördersätzen wurden im Laufe der Jahre stufenweise abgesenkt. Es gab seit 2000 mehrere Novellierungen des Erneuerbare-Energien-Gesetzes. Über die Jahre und da es eine garantierte Vergütung von 20 Jahren (EEG § 21 (2)) gibt, haben sich Tausende von Vergütungskategorien (unterschiedlicher Jahre und Leistungsstufen) für die verschiedenen Energieträger Wasser, Biomasse, Windenergie, Solarenergie und Geothermie angesammelt. Für Kraft-Wärme-Kopplungs-Anlagen gibt es noch einmal gesonderte Vergütungskategorien. Mittlerweile gibt

9 https://www.lidl-strom.de/, abgerufen am 12.05.2023.

10 https://www.dbstrom.de/, abgerufen am 12.05.2023.

11 https://www.elli.eco/de/volkswagen/naturstrom, abgerufen am 12.05.2023.

es Ausschreibungsverfahren und die Direktvermarktung wird gefördert (bei großen Solaranlagen gibt es keine Einspeisevergütung mehr). Seit 2017 gibt es einen »Mieterstromzuschlag«, eine Subvention für Strom aus Solaranlagen auf Wohngebäuden, der an die Hausbewohner geliefert wird.

7.1.3. Produktmarkt-Governance: staatliche Regulierung und Digitalisierung

Unabhängig von ihrer Rechtsform oder ihrer Marktrolle finden sich die Unternehmen mit Normen und Regelungen konfrontiert, welche »die Produkteigenschaften und Nutzungsmöglichkeiten sowie Wettbewerbsstrukturen der Industrie betreffen« (Jürgens 2007: 128). Diese Produktmarkt-Governance wird geprägt durch Staat, Verbände und einzelne Firmen.

Eckpunkte für die Koordinierung und Steuerung durch den Staat sind die Harmonisierung der Kommunikationsstrukturen, die Durchsetzung des Wettbewerbsprinzips, das Setzen von Standards oder die Regionalisierung der Käufermärkte (vgl. Jürgens 2007: 129f.). Das zentrale Ziel ist es, Strukturen zu schaffen, die einen Markt mit Wettbewerbsmechanismen etablieren. Einen großen Anteil nimmt in der Stromwirtschaft die Produktgestaltung ein.

7.1.3.1. Wechsel Stromlieferant, Energierechnung und Stromhandel

Kund:innen konnten zwar schon ab 1998 theoretisch ihren Stromlieferanten wechseln. Aber erst 2007 gab es klare Vorgaben für den Wechsel des Lieferunternehmens. Im Zuge der Liberalisierung wurde die Strombörse eingeführt und Social Media ist wie für Firmen anderer Branchen ein wichtiger Kommunikationskanal für EVU geworden.

Um der Kundschaft ein vielfältigeres Angebot bieten zu können, den Strompreis transparent zu machen und informierte Kaufentscheidungen zu ermöglichen, müssen sämtliche Abgaben, der Energiemix und die Herkunft des gekauften Stroms auf der Rechnung angedruckt werden. Die Zusammensetzung des Preises muss gemäß EnWG § 40 Abs. 2 für die Letztverbrauchenden transparent sein, d.h. auf der Rechnung aufgeführt werden. § 42 des Energiewirtschaftsgesetzes (EnWG) verpflichtet Stromlieferanten, ihrer Kundschaft die Zusammensetzung der Energieträger des gelieferten Stroms (bspw. Kohle, Gas oder erneuerbare Energien) anzugeben. Seitdem § 66 Abs. 9 EEG und § 118 Abs. 5 EnWG am 1. Januar 2013 in Kraft getreten sind, gibt es das Herkunftsnachweisregister (HKNR), welches durch das Umweltbundesamt geführt wird. In dem Register sind sämtliche erneuerbaren Energiequellen aufgelistet. Ziel ist es, dem »gemischten« Strom aus der Steckdose genau eine Energiequelle zuordnen zu können, damit die Verbrauchenden sichergehen können, dass sie »grünen« Strom bekommen. Der Herkunftsnachweis für Strom aus Erneuerbaren-Energie-Anlagen ist in analoger Form nicht umsetzbar – das betrifft auch das Abbilden des Strommix auf der Stromrechnung, um so der Kundschaft transparent zu machen, welchen Strom sie

bezieht.[12] Für den Herkunftsnachweis und den Strommix mussten Entwickler:innen die entsprechende Software programmieren.

Für einzelne Stromkund:innen war zwar durch die Regulierung der Wechsel des stromliefernden Unternehmens möglich geworden. Es fehlte aber der Ort, wo sie die unterschiedlichen Angebote vergleichen konnten. Was in anderen Märkten durch ein neues Produkt im Warenregal geschehen kann, geschieht in der Stromwirtschaft über digitale Prozesse. Dafür betreiben einige Firmen Internetseiten, auf denen die Verbraucher:innen für ihren Wohnort die Preise der Stromlieferanten vergleichen und auch gleich einen Wechselprozess starten können (z.B. Verivox oder Check24).

Aber nicht nur auf den Vergleichsportalen müssen die EVU aktiv sein. Sei es Social Media, Apps, Online-Kund:innenportale oder Smart Home: Die Erwartung der ständigen Erreichbarkeit und jene der Selbstdarstellung und Aufmerksamkeitsgenerierung hat Einzug gehalten in die EW. Manche EVU richten Abteilungen für die Facebook-, Twitter, Instagram und Whats-App-Betreuung ein. Dabei vermischt sich Kund:innenservice, PR und Vertriebsarbeit, wenn über Facebook nicht nur Beschwerden eingehen, sondern auch Produkte erworben werden und über neue Aktionen in der Kommune des Stadtwerkes informiert wird.

Exkurs zu Netzentgelten[13]

Die Stromrechnung soll die Kundschaft nicht nur darüber informieren, wie der Strom produziert wird, sondern auch, wie sich der Strompreis zusammensetzt. Neben den Steuern und Abgaben fordert das EnWG, die Netzentgelte auf der Rechnung auszuweisen. Auf der Stromrechnung mag das nur ein Detail sein, das den meisten Verbrauchenden nie aufgefallen ist. Anhand dieses Beispiels kann aber exemplarisch die Verschränkung aus juristischer Regulierung und digitaler Abbildung aufgezeigt werden. Auch hier kann von einer unvollständigen Regulierung gesprochen werden. Es bleibt nämlich offen, wie bspw. die Netzentgelte auf die Rechnung kommen. Es gibt an die 1000 Netzbetriebe (VNB) und für ein Stromlieferunternehmen bedeutet das, dass es, je nachdem, in welches Netz es liefern will, andere Netzentgelte auf der Stromrechnung ausweisen muss. Eine Möglichkeit ist, die Beträge aus der Rechnung, die das netzbetreibende Unternehmen an das energieliefernde Unternehmen für die Nutzung seines Netzes schickt, einfach auf der Rechnung an die Endkundschaft auszuweisen. Der Zahlungsfluss ist nämlich so, dass das Lieferunternehmen das Geld der Kundschaft einzieht und dies dann an den VNB weiterleitet. Dafür schickt der VNB dem Lieferunternehmen eine Rechnung. Nach 1998 und bevor die Marktkommunikation (s.u.) funktionierte, geschah dies per Post. Folglich mussten Wäschekörbe mit Netzrechnungen in die IT-Systeme eingetippt werden. Mittlerweile passiert das digital und automatisiert. Wie kann nun das Lieferunternehmen sicher sein, dass die Rechnung stimmt? Und was passiert,

12 Aus der Steckdose kommt immer der gleiche Strom. Die Zuordnung des eigenen Stroms zu bestimmten Energiequellen ist physikalisch nicht möglich. Dazu müssten die Verbrauchenden Teil eines Inselnetzes sein, in das bspw. nur Strom aus erneuerbaren Energiequellen eingespeist wird.

wenn der Rechnungszeitraum des VNB mit jenem vom Lieferunternehmen nicht übereinstimmt? Soll das Lieferunternehmen immer auf die Rechnung des VNB warten, bis es die Rechnung an die Kundschaft schicken kann? Wenn das Lieferunternehmen nicht warten will, muss es die Netzentgelte der jeweiligen VNB kennen. Die BNetzA verpflichtet zwar die VNB, die Netzentgelte zu publizieren. Diese Veröffentlichung erfolgt allerdings nicht zentral in einer Datenbank, sondern auf den Webseiten der VNB. Die Zentralisierung der Daten haben zwei Unternehmen übernommen (ene't GmbH und GET AG). Sie haben eine Datenbank mit sämtlichen Netzentgelten angelegt und bieten diese den Lieferunternehmen zum Verkauf an. Damit bieten sie dem Vertrieb eine Grundlage, Preiskalkulationen für die unterschiedlichen Netzgebiete vornehmen zu können. Ohne die Netzentgelte je Netzgebiet kann das EVU seiner Stromkundschaft nämlich gar kein durchkalkuliertes Angebot machen. Die Alternative sind vom Netzgebiet unabhängige Preise. Dann muss der Vertrieb aber damit leben, dass er in dem einen Gebiet mehr und in dem anderen Gebiet weniger Marge hat. Die stellenweise sich in einer Straße ändernden Netzgebiete informationstechnisch zu integrieren und in jeder Abrechnung zu berücksichtigen, ist enorm aufwendig. Seit Jahren gibt es eine Diskussion, ob nicht der VNB seine Preise dem Lieferunternehmen über die Marktkommunikation (mittels eines EDIFACT Datenformats) zusendet, damit dieser dann eigene Kalkulationen machen kann. Genauso gut könnte die BNetzA auch eine zentrale Datenbank mit sämtlichen Preisen vorhalten und diese zur Verfügung stellen oder einen Web Service anbieten, der einem die Preise zu einem Netzgebiet zurückliefert. So gab und gibt es unzählige IT-Fachkräfte, die sich damit beschäftigen, die Netzentgelte auf die Rechnung der Verbrauchenden zu bringen.

Im Zuge der Liberalisierung wurde eine Strombörse eingeführt. Indizien, dass dort Manipulationen stattgefunden haben, gibt es viele (vgl. Becker 2010: 143ff.). Es mussten zusätzliche Regulierungen her. Die Bundesnetzagentur überwacht mittlerweile zusammen mit der bei ihr angesiedelten nationalen Markttransparenzstelle für den Großhandel von Strom und Gas das Verbot von Insiderhandel und Marktmanipulation. Die dafür geltende europäische Verordnung über die Integrität und Transparenz des Energiegroßhandelsmarkts (kurz: REMIT) ist seit 2011 wirksam. Zu den Aufgaben der Bundesnetzagentur zählen die Registrierung von Marktteilnehmenden, die Durchsetzung von Datenmeldepflichten sowie die Verfolgung von Verstößen. Neben der REMIT gibt es noch Verordnungen wie MiFID[14] I, MiFID II, MiFIR[15], EMIR[16]. Ihr Ziel »ist die Sicherstellung von fairem Wettbewerb, effizienteren Märkten und vor allem, das verlorene Vertrauen der Stakeholder in die Märkte zurückzugewinnen.« (Kolloch/Golker 2016: 45) Ein nach REMIT meldepflichtiger Marktteilnehmer ist dazu verpflichtet, sich bei der jeweiligen nationalen Behörde zu registrieren. In Deutschland wird diese Funktion durch die Bundesnetzagentur (BNetzA) übernommen.

13 Basierend auf Erfahrungen des Verfassers als IT-Berater in der Energiewirtschaft.

14 »Markets in Financial Instruments Directive«

15 »Markets in Financial Instruments Regulation«

16 »European Market Infrastructure Regulation«

7.1.3.2. Zentrale Systemsteuerung, IT-Sicherheit und Standards

Zentral gesammelte Daten über das gesamte Stromsystem, Vorgaben zur IT-Sicherheit und technische Standards kennzeichnen die Produktmarkt-Governance der EW.

Eine der Aufgaben der BNetzA ist es, zentral Daten zu sammeln. Sie führt eine Kraftwerksliste systemrelevanter Kraftwerke und das bereits erwähnte Marktstammdatenregister. Die Marktstammdatenregisterverordnung (MaStRV) verpflichtet dazu, sämtliche ortsfeste Stromspeicher unabhängig von ihrem Inbetriebnahme-Datum im Marktstammdatenregister zu registrieren. Es soll der Überblick behalten werden. Mittlerweile stellt die BNetzA eine Ladesäulenkarte[17] für E-Mobilität zur Verfügung. Über die Jahre werden immer mehr Daten gesammelt und aufbereitet und die zu sammelnden Daten werden komplexer.

Mit der steigenden Bedeutung der IT muss der Staat sich um die Sicherheit kümmern. Es gibt eine ganze Reihe von Gesetzen und Verordnungen, die für die IT-Sicherheit sorgen sollen: Im IT-Sicherheitsgesetz, das zum 25. Juli 2015 in Kraft getreten ist, geht es schwerpunktmäßig um kritische Infrastrukturen. Durch das BSI-Gesetz wird das BSI zur zentralen Meldestelle für die IT-Sicherheit. Der § 11 des EnWG ermächtigt die BNetzA, einen Katalog von Sicherheitsanforderungen vorzulegen. Dazu gehört u.a. die Einführung eines Informationssicherheits-Managementsystems (ISMS) oder, dass jedes Unternehmen Ansprechpersonen für IT-Sicherheit benennen muss. Wie weiter oben bereits erwähnt (7.1.2.4), zeigt die Einführung des Smart Meter Gateway Administrator, dass Sicherheitsvorkehrungen Innovationen erschweren und Barrieren für die Marktteilnahme errichten. Die Zertifizierung zur IT-Sicherheit ist sehr aufwendig.

Es gibt in der Stromwirtschaft unzählige Standards und mit der Digitalisierung werden es immer mehr. Die Deutsche Energie-Agentur GmbH (dena) hat in einer Studie von 2018 Schnittstellen und Standards für die Digitalisierung für Smart Grids, Virtuelle Kraftwerke (Näheres siehe 7.2.2.3), Smart Meter, Smart Home und Elektromobilität aufgelistet (vgl. Limbacher/Richard 2018). Im Rahmen des offenen Industrieforums VHPready e. V., einer Initiative für die Integration und Standardisierung von dezentralen Energiesystemen, wurde ein offener Industriestandard zur Steuerung von dezentralen Stromerzeugungsanlagen, Verbrauchern und Energiespeichern erarbeitet[18]. Mittlerweile werden in von der Europäischen Union geförderten Projekten wie z.B. DISPOWER, FENIX und MICROGRIDS ebenfalls Standards für eine einheitliche Informations- und Kommunikationstechnologie im Bereich dezentraler Energiesysteme entwickelt. Mit diesen Standards wird u.a. sowohl die internetbasierte Steuerung eines virtuellen Kraftwerkes möglich als auch der automatisierte Handel mit Strom. Virtuelle Kraftwerke unterliegen mittlerweile den IT-Sicherheitsstandards des BSI, weil sie Teil der kritischen Infrastruktur sind. Der BDEW listet acht Smart-Home-Funktionsstandards und zu jedem davon mindestens zwei unterschiedliche produktanbietende Firmen auf (vgl. BDEW 2016: 65). Dann gibt es noch Initiativen wie die Interessengemeinschaft

17 https://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/HandelundVertrieb/Ladesaeulenkarte/Ladesaeulenkarte_node.html, abgerufen am 16.05.2023.

18 https://www.smartgrids.at/VHPReady.html, abgerufen am 16.05.2023.

Geschäftsobjekte Energiewirtschaft e. V., die Standards für Energiedienstleistungen und die damit zusammenhängenden digitalen Prozesse etablieren will[19].

7.1.4. Prozess-Governance: Systemstabilität und regulierter Datenaustausch

In dieser Arena der Governance geht es um die »Ausgestaltung der interorganisationalen Beziehungen in den Wertschöpfungsketten« (Jürgens 2007: 132). Typisch für die Stromwirtschaft ist die interorganisationale Prozessgestaltung durch die BNetzA, die vielfältigen Kooperationen und Beteiligungen.

7.1.4.1. Regulierter Datenaustausch entlang der Wertschöpfungskette

Die BNetzA lieferte 2007 die ersten Prozessbeschreibungen für den Datenaustausch zwischen den Unternehmen. Damit die einzelnen Marktteilnehmenden ihre Rolle im System erfüllen können, ist genau definiert, welche Daten sie von wem in welcher Frist zu erhalten haben. Das legt die Beschlusskammer 6 der BNetzA[20] in den folgenden Prozessen fest:

- Geschäftsprozesse zur Belieferung der Kundschaft mit Elektrizität (GPKE)
- Wechselprozesse im Messwesen Strom (WiM Strom)
- Marktprozesse für erzeugende Marktlokationen (Strom) (MPES)
- Marktregeln für die Durchführung der Bilanzkreisabrechnung Strom (MaBiS)

Der Datenaustausch zwischen den Firmen für einen reibungslosen Ablauf der interorganisationalen Prozesse basiert auf dem EDIFACT-Format (Electronic Data Interchange for Administration, Commerce and Transport), das wiederum auf EDI (Electronic Data Interchange) basiert. Dafür gibt es je Vorgang unterschiedliche Formate (bspw. für Stammdatenaustausch das Datenformat UTILMD, für den Austausch von Zählerständen MSCONS), die über eine Webseite einsehbar sind[21]. Es erscheinen zweimal im Jahr neue EDI-Formate für die Marktkommunikation[22]. Den rechtlichen Rahmen zwischen Marktteilnehmenden regeln Verträge.

Wie bereits oben erwähnt, ist der Übertragungsnetzbetreiber (ÜNB) für die Netzstabilität zuständig. Maßstab ist hierfür die Netzfrequenz von 50 Hertz. Diese Frequenz darf nur minimal über- oder unterschritten werden, weil sämtliche technischen Anlagen (ob Industrie oder Haushalt) darauf eingestellt sind. ÜNBs können mehrere Maßnahmen ergreifen, um diese einzuhalten: Kraftwerke abschalten (Redispatch), verbrauchende Einrichtungen abschalten (abschaltbare Lasten) und auf Regelenergie zurückgreifen, um unvorhergesehene Leistungsschwankungen in ihren Stromnetzen

19 https://www.bo4e.de/, abgerufen am 16.05.2023.

20 https://www.bundesnetzagentur.de/DE/Beschlusskammern/BK06/BK6.html, abgerufen am 16.05.2023.

21 https://www.edi-energy.de/, abgerufen am 16.05.2023.

22 Auf der Webseite https://www.edi-energy.de ist das Archiv einsehbar. Bspw. beginnt es für das Datenformat UTILMD (Übermittlung Daten der Kundschaft) bereits 2007. Bis 2022 gibt es dafür 26 Versionen. Jede neue Version macht eine Softwareanpassung notwendig.

auszugleichen. Mithilfe des Einspeisemanagements kann der verantwortliche Netzbetrieb (VNB) unter besonderen Voraussetzungen die Einspeisung aus EE- und KWK-Anlagen vorübergehend abregeln (§ 13 Abs. 2, 3 EnWG).

Um diese physikalischen Voraussetzungen zu gewährleisten, fließt neben Strom Geld. Am 1. Juni 2010 traten die Marktregeln für die Durchführung der Bilanzkreisabrechnung Strom (MaBiS) in Kraft. Sie regulieren die Bilanzierung der in einem Monat im Stromnetz verteilten Energiemengen und die Abrechnung der Bilanzkreise. Damit regelt die MaBiS den kaufmännischen Ausgleich zwischen Stromlieferanten und VNB. Der Hintergrund ist, dass das Lieferunternehmen für seine Kundschaft den Verbrauch prognostizieren muss, damit der entsprechende Strom geordert werden kann. Die Abweichungen zwischen dieser Prognose und dem tatsächlichen Verbrauch werden finanziell ausgeglichen. So ist das Lieferunternehmen zu einer möglichst genauen Prognose angehalten.

Zur Durchsetzung ihrer Entscheidungen stehen der BNetzA umfangreiche Mittel zur Verfügung. Die Aufsicht umfasst u.a. die ordnungsgemäße Abwicklung der Zahlungen von den VNB an die Betreibenden von Erneuerbaren-Energie-Anlagen oder Maßnahmen für die Systemsicherheit wie bspw. Redispatch-Maßnahmen, durch die Erzeugungsanlagen abgeschaltet werden.

7.1.4.2. Typisch für die Branche: Kooperationen und Beteiligungen

Die rechtlichen Besonderheiten kommunalen Wirtschaftens führen zu vielfältigen überkommunalen Kooperationen. Bontrup und Marquard sehen für die kommunalen Unternehmen keine andere Möglichkeit: »Kooperation oder Ausverkauf« (Bontrup/Marquardt 2010: 362). Die Thüga Holding GmbH & Co. KGaA hat 90 gesellschaftende Unternehmen aus der kommunalen Energie- und Wasserwirtschaft[23]. Zu ihr gehört das IT-Dienstleistungsunternehmen (IT-DL) Thüga SmartServices. Es bietet IT-Dienstleistungen für über 250 Organisationen an[24]. Südweststrom hat 59 gesellschaftende EVU[25]. Ein IT-DL der Branche wie rku.it hat 19 kommunale Anteilsinhabende[26] und die items GmbH neun[27]. Trianel, als Erzeugungskooperation gestartet, bietet mittlerweile digitale Handelsplattformen an und hat 57 Gesellschaftende (vgl. Trianel GmbH 2022).

Typische Formen von Kooperationen sind Shared Service Center wie jene der Thüga SmartService, rku.it, items oder Trianel. Sie bieten IT-Outsourcing an und betreuen ERP-Systeme (zum überwiegenden Teil von SAP) für Stadtwerke.

Eine Untersuchung hat 65 Innovationskooperationen von Stadtwerken unter die Lupe genommen. Die Autoren stellen fest, dass nicht nur die großen EVU, sondern vor allem mittelgroße für Innovationen kooperieren. Von den 65 untersuchten Kooperationen finden 57 zwischen Firmen mit weniger als 250 Mitarbeitenden statt (vgl. Lütjen et al. 2014).

---

23 https://thuega-cdn-copy.s3.eu-central-1.amazonaws.com/Thuega/documents/Th %C3 %BCga-Gruppe-Feb-2023.pdf, abgerufen am 28.04.2023

24 https://smartservice.de/thuega-smart-service/daten-und-fakten/, abgerufen am 28.04.2023.

25 https://www.suedweststrom.de/gesellschafter/, abgerufen am 28.04.2023.

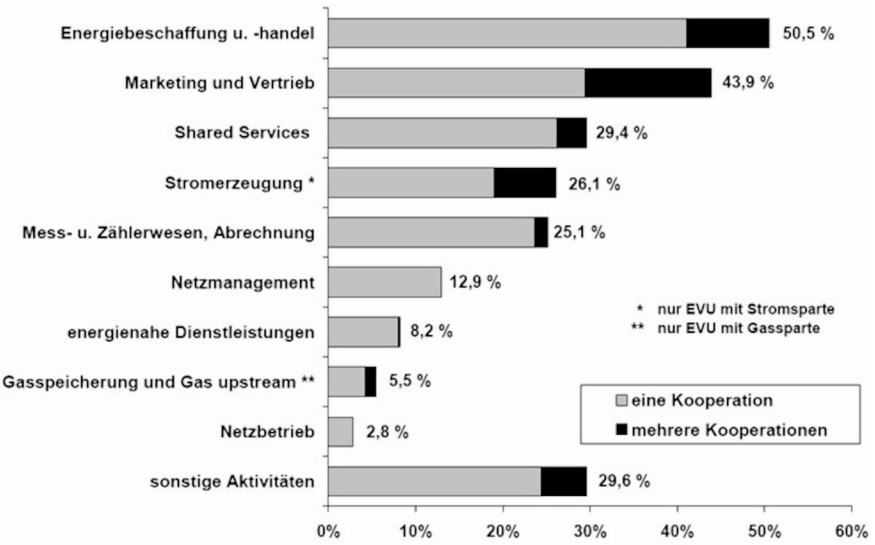
26 https://rku-it.de/ueber-uns/profil/gesellschafter, abgerufen am 28.04.2023.

27 https://itemsnet.de/ueber-uns/, abgerufen am 28.04.2023.

In einer anderen Studie hat Sander (2009) 820 kommunale EVU untersucht und 277 Kooperationen identifiziert. Er kommt zu dem Ergebnis, dass drei Viertel aller kommunalen EVU an mindestens einer Kooperation und über 40 % an mindestens zwei Kooperationen beteiligt sind. Die Wahrscheinlichkeit zu kooperieren und die Anzahl an Kooperation nehmen mit der Größe des Energieversorgers zu. Es gibt noch weitere förderliche Umstände: EVU im Alleinbesitz einer Kommune bevorzugen Kooperationen mit anderen kommunalen Unternehmen. Sie kooperieren nur zu einem geringen Anteil mit großen Energiekonzernen. Abbildung 9 zeigt, dass die Hälfte der kommunalen EVU in den Bereichen Energiebeschaffung und -handel sowie knapp 44 % im Bereich Marketing und Vertrieb kooperieren.

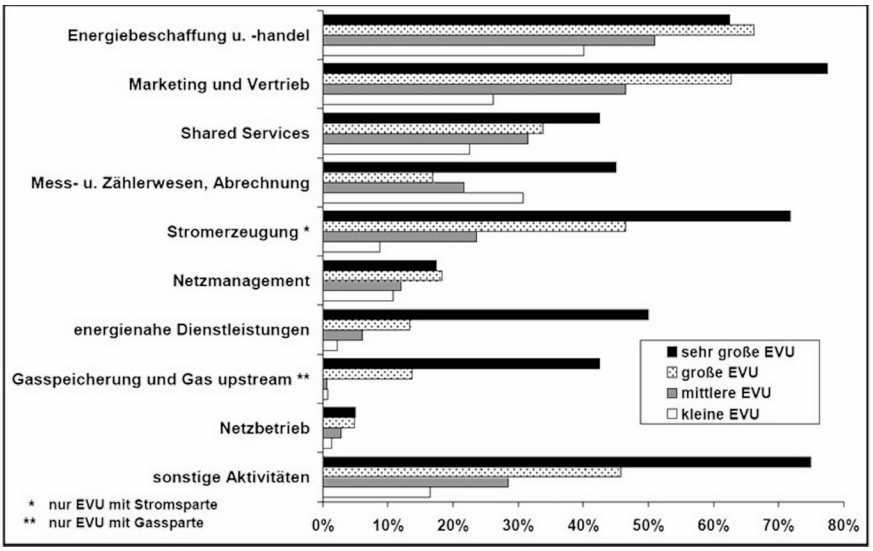
Kleine und mittlere EVU kooperieren relativ häufig in den Bereichen Shared Service (IT, Materialbeschaffung, Personal etc.) sowie im Mess- und Zählerwesen und der Abrechnung (Abbildung 10). In diesen Bereichen erfordert die Einführung und Integration neuer, komplexerer Datensysteme einen hohen finanziellen, organisatorischen und personellen Aufwand (vgl. ebd. 27). Gleichzeitig gaben bei einer Studie der Unternehmensberatung PWC über 60 % der befragten Geschäftsführenden von Stadtwerken an, ihre Ziele mit der letzten Kooperation »eher nicht« oder nur »teils, teils« erreicht zu haben (vgl. Sander 2009: 37f.).

Abbildung 8: Kooperationsgrad und -häufigkeit je Wertschöpfungsbereich



(Quelle: Sander 2007: 25)

Abbildung 9: Kooperationsgrad nach Wertschöpfungsbereich und Unternehmensgröße



(Quelle: Sander 2007: 26)

In der Untersuchung von Sander zu Kooperationen von kommunalen Unternehmen ist nicht klar, welche Rolle die IT spielt. Es ist auch nicht klar, auf welche Bereiche der IT sich das Outsourcing bezieht. Genaue Daten über den Umfang des IT-Outsourcings in der Branche fehlen.

7.1.5. Governance industrieller Beziehungen: Betriebsräte und Akademisierung

In dieser Arena der Industrie-Governance geht es um die Machtverteilung zwischen Belegschaft und Management, um Arbeitsplatzsicherheit und den Organisationsgrad der Beschäftigten (vgl. Jürgens 2007: 141ff.). Betriebsräte und Gewerkschaften sind in der Stromwirtschaft stark verankert. Die Beschäftigungsreduktion im Zuge der Liberalisierung konnte die Branche »im Wesentlichen ohne betriebsbedingte Kündigungen« (Bontrup/Marquardt 2010: 291) gestalten.

Leider gibt es für die Stromwirtschaft allein keine Daten zum Organisationsgrad. Allerdings ist sie Teil des Wirtschaftszweigs Energie/Wasser/Abfall/Bergbau, für den Daten des IAB-Panels aus dem Jahr 2016 vorliegen. Laut der Studie arbeiten 82 % der Beschäftigten in Betrieben mit Betriebsrat (vgl. Ellguth/Kohaut 2017: 283). 75 % unterliegen dem Branchentarifvertrag in West- und 44 % in Ostdeutschland (vgl. Ellguth/Kohaut 2017: 280). 65 % der Betriebsräte geben in einer Studie von 2010 an, dass sich die Arbeitsbedingungen verschlechtert oder stark verschlechtert haben. Ein Drittel der Betriebsräte geben an, dass ihre Vorschläge gleichberechtigt in unternehmerische Entscheidungsprozesse einfließen (»trifft zu« oder »trifft voll zu«) (vgl. Marquardt/Bontrup 2010: 295). Die beiden großen Gewerkschaften der Branche sind Verdi und IG BCE. Letztere ist eher in Ostdeutschland und bei den fossilen Kraftwerksbetrieben vertreten. Erstere bezieht ihre Stärke vor allem daraus, dass sich die Branche aus dem

öffentlichen Dienst heraus entwickelt hat (wo Verdi einen Schwerpunkt hat). Bontrup und Marquard kommen in ihrer Befragung von 53 Unternehmen bei 38 % der Unternehmen auf einen gewerkschaftlichen Organisationsgrad von über 50 %. Dabei sind in der Erzeugung am meisten Beschäftigte organisiert. Der Organisationsgrad nimmt mit der Unternehmensgröße zu. Nur 6 % der Unternehmen unterliegen dem Mitbestimmungsgesetz von 1976. Sie zitieren einen Verdi-Mitarbeiter, der meint, dass es schwieriger geworden ist, Kandidat:innen als Arbeitsdirektor:in gegen die Kapitalseite durchzusetzen (vgl. Bontrup/Marquardt 2010: 273f.). Die IT-Branche hat eine andere Tradition, was Mitbestimmung und gewerkschaftlichen Organisationsgrad anbelangt. Bei der ersten Betriebsversammlung von SAP stimmten 90 % gegen einen Betriebsrat (vgl. Siegele/Zepelin 2009: 151).

Obwohl der Organisationsgrad im Vergleich zu anderen Branchen hoch ist, sind die durch die Liberalisierung entstandenen Gewinne der Arbeitsproduktivität nicht bei der Belegschaft angekommen. Das betrifft zwischen 1998 und 2006 sowohl die kleinen als auch die großen Stromversorger.

> »Stattdessen haben sich die Bezieher von Gewinneinkommen, Mieten und Pachten überproportional stark bedient.« (Bontrup/Marquardt 2010: 107)

Für einzelne EVU wie die Stadtwerke München konnte eine Akademisierung der Belegschaft verzeichnet werden. Dort sind auch wieder so viele Beschäftigte angestellt wie vor der Liberalisierung (vgl. Bähr/Erker 2017: 371). Wie auch in anderen Firmen wurden im Zuge der Liberalisierung Leiharbeitende in Call-Centern eingesetzt und Busfahrer:innen zu schlechteren Bedingungen in Tochterfirmen eingestellt (vgl. Bähr/Erker: 353). Die Qualifikationsstruktur hat sich allgemein im Zuge der Liberalisierung verschoben von »gewerblichen zu höher bzw. hochqualifizierten Verwaltungs-, Gestaltungs- und Managementarbeitsplätzen« (Bontrup/Marquardt 2010: 97). Zudem ist ein Outsourcen »weniger [benötigter] Unternehmensbereiche […] und ein Abschieben minderer Arbeitsqualifikationen in unternehmerische Randbereiche« (ebd.) zu beobachten. Zu der Akademisierung tragen zusätzlich die vermehrten Kooperationen bei.

> »Je mehr ein Stadtwerk mit anderen Unternehmen zusammenarbeiten muss, desto mehr Dienstleistungsmanagement ist gefordert.« (Bontrup/Marquardt 2010: 366f.)

Explizit in Bezug auf die Energiewirtschaft stellt eine Studie fest, dass die Entflechtung von Netz und Vertrieb zu mehr bürokratischer Arbeit führt (vgl. Flecker/Hermann 2011: 539). Die digitale Transformation trägt zusätzlich zur Akademisierung bei. Es »nehmen komplexe Tätigkeiten zu, die in der Regel auch tiefere Kenntnisse im Umgang mit der digitalen Technik voraussetzen.« (Roth 2018: 78)

> »Die Arbeit wird zunehmend unter Verwendung digitaler Arbeitsmittel und mit der Unterstützung digitaler Assistenzsysteme (Software- wie Hardwarekomponenten) erbracht und organisiert werden. Dies umfasst nahezu alle Tätigkeitsbereiche in der Energieversorgung, von den Monteuren über die Meister, Techniker und Ingenieure bis

> hin in den Vertrieb, das Marketing und die Querschnittsbereiche wie Personalwesen, Buchhaltung, Controlling und die kaufmännische Sachbearbeitung.« (Roth 2018: 77)

Neben der veränderten Zusammensetzung der Belegschaft lösen sich einzelne Gruppen stärker räumlich von der Organisation als andere.

> »Die Digitalisierung ermöglicht die zeitliche, örtliche und organisatorische Flexibilisierung der Arbeit und verstärkt sie.« (Roth 2018: 64)

Es zeigt sich eine veränderte Zusammensetzung der Belegschaft (vgl. Roth 2018):

- Insgesamt weniger Bedarf gibt es im Rechnungswesen, Controlling, kaufmännische Sachbearbeitung, Personalwesen, für Ingenieur:innen (Erzeugung), Mechaniker:innen, Monteur:innen (Erzeugung), Monteur:innen (klassisch, Netze), Produktentwickler:innen (Commodity).
- Gleichbleibenden Bedarf gibt es für Monteur:innen (Bau) und Koordinator:innen (Bau).
- Mehr Beschäftigte sind in den Bereichen WFM Montage (Netze), IT-Produktentwicklung, Ingenieur:innen (Netze), Online-Marketing, Projektmitarbeit und Datenanalyse (Vertrieb) zu erwarten.

Die benötigten IT-Fachkräfte sind aber nicht leicht zu finden. Laut einer Studie[28] von BDEW und EY sehen Führungskräfte ein Haupthemmnis für die Digitalisierung im Fehlen personeller Ressourcen und der Qualifikation der Mitarbeitenden (63 %) sowie in nicht ausreichenden IT-Ressourcen im Haus (54 %) (vgl. BDEW/EY 2018: 16).

Eine Studie (Flecker und Hermann 2011) zu den Folgen der Liberalisierung mit sechs Fallstudien in europäischen Ländern im Bereich Elektrizität (drei in Belgien, jeweils eine in Österreich, Polen, UK) kommt zu ähnlichen Ergebnissen: Personalabbau zwischen 25 und 50 % seit Mitte der 90er. Es gibt weniger »blue collar«-Tätigkeiten (Erzeugung, Instandhaltung, Administration) und mehr »white collar«-Tätigkeiten (Handel, Einkauf, Controlling, IT) (vgl. ebd. 531). Zusätzlich zeigt die Studie, dass eine Reaktion auf die Liberalisierung eine Restrukturierung (Outsourcing, Konzentration) war (vgl. ebd. 528). Vor allem Beschäftigte aus Österreich, UK und Polen betonen gestiegene Arbeitsintensität und Unterbesetzung. Call-Center-Beschäftigte haben die schlechtesten Arbeitsbedingungen (vgl. ebd. 538). Ein Beschäftigter aus Österreich war der Ansicht, dass die Einführung von SAP größere Auswirkungen auf das Unternehmen hatte als die Liberalisierung (vgl. ebd. 530).

28 193 Geschäftsführende und Vorstände von Stadtwerken und EVUs in Deutschland, Österreich und der Schweiz wurden im Februar/März 2018 anhand eines standardisierten Fragebogens telefonisch befragt.

7.2. Folgen der Industriestrukturen für die Softwareentwicklung

So vielfältig und zahlreich die Betriebe in der Energiewirtschaft sind, so vielfältig sind die Softwarestrategien und -anwendungsbereiche, so zahlreich die Möglichkeiten, um mit Software Geld zu verdienen. Die Regulierung beeinflusst Produkte und Prozesse der Branche und kommt ohne Software nicht aus. Verändert sich die Regulierung, braucht meist auch die Software ein Update. Dabei gibt es, wie es in der Branche allgemein üblich ist, viele Kooperationen. Die Beschäftigten müssen interdisziplinär arbeiten und über staatliche Regulierung, Softwarelandschaften/-pakete und energiewirtschaftliche Technik und Geschäftsprozesse Bescheid wissen. Aufgrund des hohen Organisationsgrades spielt der Betriebsrat auch bei der Softwareentwicklung eine Rolle.

7.2.1. Digitalisierungsstrategien zwischen Anwendung und Entwicklung

Selbst Software zu entwickeln ist bereits seit langem Teil der Branche. Zugleich stellen die vielen EVU und die zunehmende Bedeutung von Software einen großen Markt für Standardsoftware und IT-Dienstleistungen dar. Sowohl Softwarefirmen als auch IT-DL und die EVU selbst verdienen mit Software Geld. Aber weder dominiert eine Softwarefirma noch ein EVU mit einer bestimmten Digitalisierungsstrategie die Branche.

7.2.1.1. Vielfältige Zulieferindustrie für Standardsoftware

Bei den Stadtwerken München galt bereits seit 1977 die Datenverarbeitung als Managementwerkzeug (vgl. Bähr/Erker 2017: 279ff.). Ab Ende 1997 gab es eine gezielte IT-Strategie bei den SWM, »die die datentechnische Durchdringung sämtlicher Arbeits- und Geschäftsprozesse umfasste und als integraler Bestandteil des Transformationsprozesses begriffen wurde« (Bähr/Erker 2017: 326).

Obwohl Software seit längerem die Arbeits- und Geschäftsprozesse in den EVU durchdringt, nehmen die Investitionen in Software weiter zu. Laut Statistik des Statistischen Bundesamts geben Stromversorger mit mehr als 250 Beschäftigten mehr Geld für Software aus (siehe Tabelle 5). Haben diese Unternehmen 2009 noch 86 Mio. Euro investiert, sind es 2020 302 Mio. EVU mit weniger als 50 Beschäftigten haben 2009 noch 8 Mio. und dann 2020 10 Mio. Euro investiert. Wie die Tabelle zeigt, hängt diese Veränderung nicht mit einer veränderten Anzahl an Unternehmen zusammen. Zahlen dazu, in was die EVU genau investiert haben, z.B. wie viel in die Programmierung (Anpassung eines Standards oder eigene Lösungen), gibt es leider nicht.

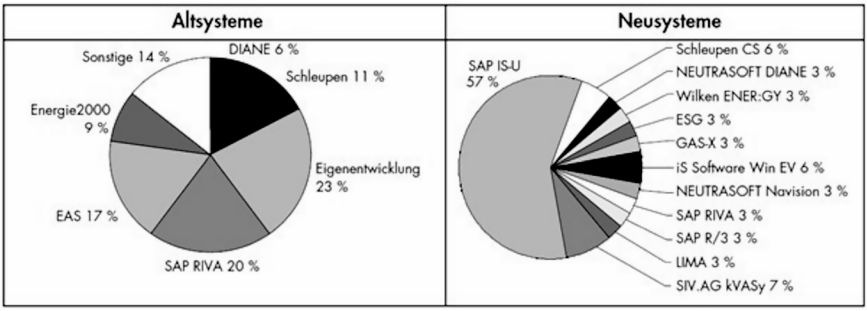
Tabelle 6: Vergleich Investitionen in Software in Millionen Euro bei Stromversorgern zwischen 2009 und 2020

	Investitionen in Software			Anzahl Unternehmen		
Unternehmen von bis Beschäftigte	**2009**	**2020**	**Veränderung**	**2009**	**2020**	**Veränderung**
0 bis 9	2	2	0 %	379	62.056[29]	+16.274 %
10 bis 19	1	2	+100 %	103	297	+188 %
20 bis 49	5	6	+20 %	223	241	+8 %
50 bis 249	28	45	+61 %	320	421	+32 %
250 und mehr	86	302	+251 %	148	185	+25 %

(Quelle: Statistisches Bundesamt 2011 und 2022)

Ein Teil der Investitionen fließt auf jeden Fall in Standardsoftwarepakete. EVU haben ihre Kern-Softwarepakete für industriespezifische Prozesse über die Jahrzehnte immer wieder einmal ausgetauscht. 1995 haben die Stadtwerke München für 13,5 Millionen D-Mark die Version R/2[30] des ERP-Systems von SAP eingeführt (vgl. Bähr/Erker 201: 326). Laut einer Studie mit 36 EVU sind zwischen 1999 und 2005 die untersuchten Firmen mehrheitlich auf die neue Branchenlösung von SAP IS-U (»Industry Solution for Utilities«) umgestiegen. Die alte Branchenlösung von SAP hieß RIVA. Die Abbildung 11 zeigt, dass von den Altsystemen 23 % Eigenentwicklungen waren (vgl. Sarshar/Loos/Weber 2006).

Abbildung 10: Vorher (1995) und nachher (2005) bei den industriespezifischen Softwarepaketen der EVU



(Quelle: Sarshar/Loos/Weber 2006: 122)

29 Die Statistik spricht seit 2018 nicht mehr von »Unternehmen«, sondern von »rechtlichen Einheiten«. Damit sind wahrscheinlich auch kleinere Anlagenbetreibende für Wind, Biomasse oder Photovoltaik Teil der Statistik.

30 Seit 1981 gab es R/2, seit 1991 R/3 und seit 2015 S/4 (https://www.sap.com/germany/about/company/history/, abgerufen am 4. Mai 2023).

Für die Ablösung alter Software gibt es vielfältige Alternativen. Weil es keine anderen Daten gibt, steht hier die Zulieferindustrie für Standard-Abrechnungssoftware[31] im Fokus. Eine Übersicht des Bundesverbandes der Energiemarktdienstleister (BEMD) sieht die industriespezialisierten Softwarefirmen von Schleupen, SIV und IS-Soft vorne, was die Größe ihrer Kundschaft anbelangt (gemeinsam 912 EVU). Tabelle 6 zeigt, dass die Softwarefirmen für die industriespezifische Software vor allem ab den 90er Jahren aktiv geworden sind. Was auffällt, ist, dass trotz der Komplexität der Branchenprozesse und der vielfältigen Anwendungsfelder die IS-Soft mit nur 21 Programmierenden 255 EVU mit ihrer Software beliefern kann. Allerdings gehört keines der belieferten Unternehmen zu den großen mit mehr als 200.000 Marktlokationen bzw. Verbrauchenden[32]. Neben IS-Soft zeichnet sich Schleupen durch seine Spezialisierung auf kleine EVU aus. Bei den großen EVU liegen SIV und SAP vorne. Die Tabelle zeigt, dass sich jüngere Softwarefirmen wie Powercloud, IQone oder Quanto schwertun, den Markt zu erobern.

Tabelle 7: Firmen Abrechnungssoftware Stand 2023

Softwarefirma	Markteintritt	Mitarbeitende für Produkt	Entwickelnde für Produkt	EVU	bis 20.000 MaLo*	20.000-200.000	ab 200.001 MaLo
Schleupen	1977	376	189	357	173	7	4
SIV	1992	399	121	300	90	83	37
IS-Soft	1996	82	21	255	190	65	0
Wilken Neutrasoft	2000	208	95	207	107	100	0
MSU	2003	90	40	181	130	50	1
Wilken ENER:GY	1999	208	95	128	35	90	3
SAP	1988	k. A.	k. A.	84	10	47	27
SDK	1994	34	30	69	32	33	4
Rhenag	1998	85	70	52	18	30	4
Somentec	1995	78	27	46	35	11	0
Klafka&Hinz	2000	115	50	41	30	10	1
robotron	1999	325	120	40	19	15	6

31 Mit Abrechnungssoftware ist gemeint, dass eine Software Zählerdaten verarbeiten und eine Rechnung an die Verbrauchenden versenden kann. Bei der BEMD-Studie der folgenden Seite ist von »Meter to Cash« die Rede. Die Softwarepakete der Firmen haben meist aber auch noch andere Funktionalitäten wie Forderungsmanagement oder Finanzbuchhaltung mit dabei und stellen vollständige ERP-Systeme dar.

32 Von »großen« EVUs wird hier gesprochen, wenn ein EVU mehr als 200.001 Kund:innen hat oder, wie es in der Tabelle heißt: MaLo – siehe die Beschriftung der letzten beiden Spalten der Tabelle 7. Die Abkürzung steht für Marktlokationen und ist vereinfacht gesagt eine technische Bezeichnung für Kund:innen der EVUs.

Softwarefirma	Markteintritt	Mitarbeitende für Produkt	Entwickelnde für Produkt	EVU	bis 20.000 MaLo*	20.000-200.000	ab 200.001 MaLo
Powercloud	2012	60	45	35	10	21	4
AKTIF	2001	33	19	28	24	4	0
Iqone	2014	8	7	5	4	1	0
Quanto	2018	30	20	5	5	0	0

(Quelle: BEMD https://www.bemd.de/download/BEMD-ITLM2C_Anbietermatrix_Abrechnungssoftware_2020.pdf, zuletzt abgerufen am 2. Mai 2023)
* Marktlokation bezeichneten eine eindeutige ID für Stromabnehmer oder -einspeiser. Vereinfacht könnte man sagen: Kund:innen der EVUs. Die Messlokation ist ein Stromzähler.

SAP begann 1995 die Entwicklung des branchenspezifischen Moduls IS-U mit dem Ziel, eine international einsetzbare Standardlösung zu programmieren (vgl. Frederick/Zierau 2011: 15f.). Die Anpassungen an den deutschen Markt konnten sich eher größere EVU leisten. Die Besonderheit war und ist immer noch, dass SAP eine Entwicklungsplattform bietet, damit EVU selbst oder IT-DL Programmierungen am Standard vornehmen können. Wie bereits unter 4.3 geschrieben, gibt es allein für die Versorgungswirtschaft über 200 Firmen, die mit SAP kooperieren. Diese große Anzahl ist ein Indiz dafür, wie umfangreich die Arbeit war und ist, um SAP IS-U zu implementieren und anzupassen.

Es ist unklar, welche Rolle andere Entwicklungsplattformen spielen und wie häufig sie die EVU nutzen. Zum Beispiel gibt es noch andere, kleinere branchenspezifische Entwicklungsplattformen wie eine Open-Source-Datenplattform für Smart-City-Anwendungen für Stadtwerke (vgl. ZfK, 02.07.2022). Die Entwicklungsplattformen AWS von Amazon[33] oder Azure von Microsoft[34] nutzen Organisationen der Branche auch. Allerdings bieten die zwei Unternehmen keine industriespezifischen Bausteine an.

7.2.1.2. Unterscheiden sich große und kleine EVU?

Auch wenn größere EVU mehr Geld für Software ausgeben, heißt das nicht, dass nicht ebenso kleine EVU agile Teams etablieren und selbst Software entwickeln. Untersuchungen zur Digitalisierung in großen und kleinen EVU zeichnen ein unscharfes Bild und zeigen eine Heterogenität an Digitalisierungsstrategien. Ob groß oder klein: Auch beim Thema Digitalisierung kooperieren die EVU.

Die Studien vom Verband BDEW, A. T. Kearney und IMP³Prove (2018 und 2019) konnten Unterschiede in der Digitalisierung zwischen großen und kleinen Versorgern[35] nur

33 https://aws.amazon.com/de/developer/tools/, abgerufen am 4. Mai 2023.

34 https://azure.microsoft.com/de-de/products/visual-studio/, abgerufen am 4. Mai 2023.

35 »Die insgesamt 80 teilnehmenden EVU aus dem Jahr 2018 umfassen Strom-, Gas- und Wassernetzbetreiberwie auch Querverbundunternehmen, darunter kleine und sehr große Unternehmen, Netzbetreiber und Vertriebsunternehmen. In ihrer Gesamtheit bilden sie alle Wertschöpfungsstufen und Unternehmensgrößen der Branche ab und lassen daher Rückschlüsse auf den generellen Digitalisierungsstand in der Energiewirtschaft zu.« (BDEW et al. 2019: 6)

für ein Jahr feststellen. In einer Studie ermittelten sie einen Digitalisierungsindex je Firma, der sich aus sieben Analysebereichen zusammensetzt (u.a. Digitalisierungsstrategie, Kostenersparnis durch digitale Interaktion mit der Kundschaft, ob agile Methoden genutzt werden, Kooperationen mit Plattformen oder IT-Fachkräften). 2018 konnten zwischen großen und kleinen EVU noch keine Unterschiede festgestellt werden. Es gab sowohl bei kleinen als auch großen »Top-Digitalisierer« wie Nachzügler. In der Studie von 2019 war dann ein leichter Zusammenhang auszumachen: je höher der Umsatz, umso weiter in der Digitalisierung. Zu einem ähnlichen Ergebnis kommt die Kanzlei Rödl & Partner. Sie messen den IT-Reifegrad[36] (vgl. Rödl/Partner 2017: 42).

Unterschiede gibt es zwischen neuen und etablierten EVU. Bei einer Befragung von elf kommunalen Versorgern mussten sich die Forschenden eingestehen, dass die »klassischen Digitalisierungsthemen etablierter Energieversorger möglicherweise nicht gänzlich auf Stadtwerke-Neugründungen zu übertragen sind« (Berlo et al. 2018: 32). Sie wägen Risiken und Nutzen der neuen Technologien stärker ab und setzen sie sehr spezifisch vor allem zum Auf- und Ausbau von Flexibilisierungsoptionen ein: flexibles Einspeisemanagement, Kontrolle von Lastverschiebungen, Bereitstellung von Speicherkapazität, Erweiterung von virtuellen Kraftwerken (vgl. Berlo et al. 2018: 31f.).

Die typische Kooperationsbereitschaft der Branche zeigt sich auch bei der Digitalisierung. EVU kooperieren: mit Start-ups (23 %), mit Mittelständlern/Großunternehmen (36 %), anderen kooperierenden Unternehmen (Beratungsunternehmen, Softwarehäuser) (36 %), mit Universitäten oder Forschungseinrichtungen (35 %) (vgl. BDEW/A.T. Kearney, IMP³Prove 2019: 23).

7.2.1.3. Software als Geschäftsfeld der EVU

Eine IT-Strategie, die sich in der Branche durchsetzt, ist nicht zu erkennen. Auf jeden Fall gibt es EVU, die mit Software Geld verdienen.

Manche Unternehmen bieten die Standardsoftwarepakete als Anwendungsplattformen an. Darunter ist zu verstehen, dass ein IT-DL oder ein EVU eine Standardsoftware implementiert und dann anderen zur Verfügung stellt. Die kommunale Kooperation Thüga will bis 2024 30 Unternehmen für ihre Abrechnungsplattform gewinnen (vgl. ZfK, 17.06.2022). Der Energiekonzern EnBW stellt eine Anwendungsplattform für Stromlieferanten zur Verfügung, auf der sie ihre Prozesse abwickeln können (vgl. ZfK, 25.02. 2019). Arvato Systems (Teil des Bertelsmann-Konzerns) bietet eine Anwendungsplattform (einzelne Module und Applikationen) als Pay-Per-Use-Service an[37].

Eine Sonderform von Anwendungsplattformen ist White-Label-Software. Was ist das? Eine Organisation entwickelt eine Software, stellt sie zur Verfügung (ob via Cloud oder On-Premises[38]) und versieht sie in der Darstellung nach außen mit der entspre-

36 Der wie folgt gemessen wird: Es gibt eine IT-Strategie. Mitarbeitende sind an die IT-Infrastruktur angebunden. Wie zufrieden Mitarbeitende mit IT-DL bzw. der IT-Abteilung sind und wie Kosten-Nutzen-Verhältnis der Unternehmens-IT eingeschätzt wird.

37 https://www.arvato-systems.de/branchen/branchen-im-ueberblick/energie-versorgungswirtschaft/aep, abgerufen am 4. Mai 2023.

38 Lizenzmodell, bei dem ein Unternehmen eine Softwarelizenz erwirbt und diese auf eigenem Server betreibt.

chenden Aufmachung des anwendenden EVU (bspw. Logo und Name Stadtwerk). Der Endkundschaft fällt nicht auf, dass es eine Standardsoftware einer anderen Firma ist. Solche White-Label-Lösungen gibt es für Kund:innenportale der Wohnungswirtschaft, mit der Anwendende Zählerstände erfassen können (vgl. ZfK, 01.02.2022). Vattenfall und EnBW kooperieren für eine White-Label-Software für den Smart-Home-Bereich (vgl. ZfK, 25.02.2019). Auch Softwarefirmen wie die oben genannte SIV bieten bspw. White-Label-Apps für Smartphones an[39].

EVU sind (auch) Softwarefirmen, wie es Doleski (2016) für eine zunehmende Anzahl von ihnen prognostiziert. Die Stadtwerke Schwäbisch Hall haben die Softwarefirma Somentec[40], das Stadtwerk Enercity aus Hannover Lynqtech[41]. EVU sind Gesellschaftende von IT-Dienstleistungsunternehmen, die selbst programmieren und Software gestalten (bspw. Thüga Smartservice GmbH, rku.it GmbH oder items GmbH).

Leider fehlen genaue Daten: Wer passt wie stark Standardpakete an? Wer nutzt einen Softwarestandard ohne Anpassung? Welche EVU bieten White-Label-Software an oder nutzen sie? Wie viel Geld verdienen EVU mit Softwareentwicklung oder mit dem Anbieten von Abrechnungsplattformen? Es gibt auch keine Übersicht über sämtliche Zulieferfirmen für industriespezifische Software.

Statt dass sich in der Energiewirtschaft softwareentwickelnde Firmen durchsetzen wie in anderen Branchen, gibt es sowohl Softwarefirmen für Standardsoftware als auch EVU, die für einzelne, industriespezifische Anwendungsbereiche Software für andere entwickeln oder betreiben. Bei den vielen EVU ist eine zur Branchenstruktur passende Strategie, diese EVU zu beliefern oder mit ihnen zu kooperieren.

Auf jeden Fall gibt es kein disruptives Unternehmen wie Amazon, das Marktführer in der Branche geworden ist und gleichzeitig viel Geld mit Software oder IT-Dienstleistungen verdient. Vielmehr koexistieren die Strategien: Einerseits etablieren sich neue Stromlieferanten mit viel Werbung am Markt wie Yello Strom von EnBW (seit 1999). Andererseits gewinnt Octopus Energy (seit 2015) Marktanteile: eine Firma, die nicht nur Strom, sondern auch Software(dienstleistungen)[42] anbietet.

39 https://www.siv.de/de/referenzen/referenzen-energielieferanten/team-energie-gmbh-co-kg/, abgerufen am 4. Mai 2023.

40 https://www.somentec.de/ueber-somentec/unternehmen/, abgerufen am 4. Mai 2023

41 https://www.enercity.de/presse/pressemitteilungen/2020/Lynqtech, abgerufen am 4. Mai 2023.

42 eigene Softwarelösung https://kraken.tech/, abgerufen am 4. Mai 2023.

7.2.2. Wechselspiel von Regulierung und Softwareentwicklung

7.2.2.1. Markt-Governance dank Software: Wechsel Lieferfirma, Transparenz und Datenaustausch

Auch wenn der Staat Software nicht selbst entwickelt, hat er durch die Regulierung einen maßgeblichen Anteil an der stetigen Softwaregestaltung in der Branche. Die Regulierung für den Wettbewerb und die Produktgestaltung gibt der Staat vor. Die dafür notwendige digitale Technik entwickelt jemand anderes.

Ein Ziel der Liberalisierung war, dass die Privatkundschaft das stromanbietende Unternehmen möglichst einfach wechseln kann. Aufgrund von Entflechtung, Auflösung der Gebietsmonopole und um solche Wechsel zu ermöglichen, müssen die Unternehmen vielfältige Daten austauschen (bspw. über Kundschaft oder Zähler). Die BNetzA übernimmt die Regulierung der Kommunikationsstrukturen. In diesem als Marktkommunikation bezeichneten System hat jedes EVU und jedes zählende Gerät eine ID (siehe oben 7.1.4.1). Die vorgeschriebenen Marktprozesse können bei der Koordination so vieler Marktakteur:innen nur effizient sein, wenn sie automatisiert ablaufen. Müssten Stromversorger langfristig eine Vielzahl an Sachbearbeitenden einstellen, um die Marktprozesse abzubilden: Wie wäre es um die Legitimität der Regulierung dann bestellt?

Vor allem für die Wertschöpfungsstufen Vertrieb und Netze müssen Unternehmen Daten zu Zahlungen, Strombilanzierung, Anlagen und Personen austauschen. Das ist kostenintensiv (vgl. Seeliger et al. 2019), birgt Risiken, macht die Industriestrukturen komplexer und erhöht den Koordinations- und Informationsbedarf. Rohracher kommt zu dem Schluss, dass »transaktionskostentheoretische Analysen auf die Kosten und Imperfektionen des Ersatzes vertikaler Netzintegration durch Preissignale« (Rohracher 2007: 144) hinweisen. Andere Autoren stellen fest, dass der »betriebene Aufwand vermuten lässt, dass ein vertikal integriertes Energieversorgungsunternehmen in öffentlicher Trägerschaft und mit Gebietsmonopol sogar effizienter und effektiver wirtschaften« (Bräunig 2012: 435) kann. Renate Mayntz konstatiert, dass allgemein die Liberalisierung von Telekommunikation, Bahn, Elektrizität zu sehr komplexen Strukturen und komplexen Abhängigkeiten zwischen Handelnden, Prozessen und System führt. Es kommt dann nicht nur zu Konflikten zwischen Regulierer und Regulierten, sondern auch zwischen den unterschiedlichen Zielen der Regulierung (vgl. Mayntz 2009: 139). Die Regulierung soll schließlich nicht nur einen Markt ermöglichen. Sie soll zudem effizient sein und eine hohe IT- und System-Sicherheit garantieren, was bei zunehmender Komplexität immer schwieriger wird.

> »Die Komplexität der Koordination, die erfüllt werden muss, um Systemstabilität zu gewährleisten, ist in allen Sektoren ein wichtiger Ansatzpunkt der Gegner der Liberalisierung« (Voß/Bauknecht 2007: 121).

Indem die Koordination an die IT-Systeme delegiert wird, entlastet sie das Handeln der Marktteilnehmenden wie z.B. einzelner Händler:innen am Energiemarkt oder der Stromkundschaft (die beide per Mausklick handeln können) und belastet die IT-Systembetreibenden.

Dass Software einen Markt ermöglicht, ist auch in weniger stark regulierten Branchen zu beobachten. In ihrer historischen Darstellung der Literatur zum Forschungsfeld Information Systems sehen Björn-Andersen und Raymond die Funktion der IT ab den 2000ern darin, Preistransparenz herzustellen und es neuen Firmen zu erleichtern, in bestehende Märkte vorzudringen (vgl. Bjørn-Andersen/Raymond 2014: 190). Crowston und Myers zeigen dies in ihrer Untersuchung der Immobilienwirtschaft (vgl. Crowston/Myers 2004: 16ff.). Dort macht es die IT der Kundschaft einfacher, sich zu informieren. Dadurch kann eine transparente, marktförmige Vermittlung stattfinden und auch Neulinge in der Branche haben eine Chance.

7.2.2.2. Softwareentwicklung für den Stromhandel: Markt ermöglichen, Geld verdienen und neue Risiken

Am Stromhandel zeigt sich, wie Software Markt ermöglicht, Regulierung Software-Innovationen erzwingt und gleichzeitig für neue Risiken durch Manipulationsmöglichkeiten und IT-Sicherheit sorgt.

Eine neue Institution, welche die Liberalisierung eingeführt hat und die auf digitaler Technologie basiert, ist die (europäische) Energiebörse. Der informationstechnische Aufwand für die Preisbildung dort ist sehr hoch. Damit der tägliche Handel gewährleistet ist, muss ein hoher Datenverkehr mit entsprechender IT-Sicherheit ermöglicht werden. Die einzelnen Marktakteur:innen müssen am gleichen IT-System angeschlossen sein. Für die Software der Energiebörse in Großbritannien wurden 1,5 Mrd. Euro ausgegeben (vgl. Rohracher 2007: 144). Für die Regulierung des Handels durch die REMIT-Verordnung der BNetzA von 2005 wird ein hoher Digitalisierungsgrad attestiert, weil der Datenverkehr eine hohe Quantität und Varianz aufweist (vgl. Kolloch/Golker 2016: 45). Andererseits bieten die durch die Regulierung erzwungenen technischen Innovationen in Form von Software Firmen die Chance, die von ihnen entwickelten Lösungen anderen anzubieten.

> »Dies kann besonders vor dem Hintergrund des schwindenden Kerngeschäfts, der Belieferung von Endkunden mit den Medien Strom und Gas, von immanenter Bedeutung für die wirtschaftliche Zukunft der EVU sein. Der Wandel vom reinen Medienvertrieb zum integrierten Medien- und Dienstleistungsvertrieb wird einer der entscheidenden Faktoren sein, die über das Überleben der EVU auf dem (Energie-)Markt entscheiden.« (Kolloch/Golker 2016: 53)

Es gibt eine ganze Reihe digitaler Handelssysteme und Beschaffungsplattformen wie EnPortal[43] (seit 2007) oder Enmacc[44] (gegründet 2016), die den Handel an der Börse oder Over the Counter (direkt zwischen zwei Parteien, nicht via Börse) anbieten. Sie eröffnen für kleine Stadtwerke und Energielieferfirmen nicht nur Einnahmequellen, sondern die Möglichkeit, an der Börse teilzunehmen.

Beim Stromhandel kann es zu Manipulationen und erhöhtem Risiko kommen. Zum einen wegen technischer Fehler: An der europäischen Strombörse kam es schon einmal

43 https://www.enportal.de/, abgerufen am 4. Mai 2023.

44 https://enmacc.com/, abgerufen am 4. Mai 2023.

zu einem Ausfall, weil Datenpakete fehlerhaft waren (vgl. Päffgen/Sperling 2019). Neben der Börse für Lieferunternehmen gibt es noch einen Regelenergiemarkt für den notwendigen Lastenausgleich im Stromnetz. Dort wird der Strom gehandelt, der eingesetzt wird, um die Netzfrequenz stabil zu halten. 2019 fiel die Netzfrequenz schlagartig auf 49,8 Hz, was einem Blackout nahekommt. Es wird vermutet, dass eine fehlerhafte Datenübertragung und die darauffolgende Reaktion einer vollautomatischen Regeleinrichtung der Grund waren. Auf jeden Fall stiegen die Preise für Regelenergie sprunghaft an (vgl. Sperling 2019).

Als Grund für die Instabilitäten des Stromsystems macht Weyer Interessengegensätze aus:

> »Ferner kann die Frage nach der Sicherheit und Zuverlässigkeit komplexer technischer Systeme gestellt werden, die unter den Bedingungen verschärften Wettbewerbs operieren müssen und zudem von unterschiedlichen Handlungslogiken geprägt werden. Für einen Stromhändler beispielsweise hat die Stabilität des Systems nicht oberste Priorität, so dass seine Transaktionen das Energienetz zusätzlich belasten können.« (Weyer 2010: 844)

Netze bekommen so einen prekären und riskanteren Status, anders als das bei den »High-Reliability-Organisationen der 1980er Jahre der Fall war« (ebd.).

Das Beispiel soll verdeutlichen, dass es ein langwieriger Regulierungsprozess ist, um ein stabiles System herzustellen, bei dem auch die Softwareentwickelnden gefragt sind und die Regulierung nicht mit dem Erlassen einer Verordnung erledigt ist. Institutionen und Technologien entwickeln sich wechselseitig in einem Prozess der Koevolution (vgl. Rohracher 2007: 148). Wobei die IT selbst institutionalisiert und »Part of the Furniture« (Silva/Backhouse 1997) der Energiewirtschaft wird.

7.2.2.3. Dezentrale Erzeugungsanlagen gebündelt vermarkten: virtuelle Kraftwerke

Mit der Energiewende wurde die Dezentralisierung der Stromversorgung wieder zum Thema. Einige Autoren haben die Hoffnung, dass mithilfe von IT und weniger kapitalintensiven Erzeugungsanlagen eine wettbewerbliche Organisation der Stromwirtschaft möglich ist (vgl. Voß/Bauknecht 2007: 119). Weil es den Regelenergiemarkt gibt und die Möglichkeit, mehrere Erzeugungsanlangen zusammen dort zu vermarkten, wurde dafür eine informationstechnische Lösung gefunden: virtuelle Kraftwerke.

Mittlerweile haben schon einige EVU (bspw. RheinEnergie[45] aus Köln oder die Stadtwerke Rosenheim[46]) solche Kraftwerke aufgebaut. Lichtblick (1999 gegründet) war eine der ersten Firmen, die solch ein virtuelles Kraftwerk geschaffen haben. Darunter versteht man die Steuerung mehrerer dezentraler Anlagen, als wären sie ein einziges Kraftwerk (im Falle von Lichtblick Gasmotoren, die Strom und Wärme produzieren – sogenannte Mikro-Blockheizkraftwerke/-BHKWs, Batterien und PV-Anlagen). Die

45 https://www.rheinenergie-trading.com/de/produkte_2/services_strom/virtuelleskraftwerk.html, abgerufen am 13.07.2023.

46 https://www.swro.de/de/dienstleistungen/energievermarktung, abgerufen am 13.07.2023.

Steuerung des Kraftwerkpools, zu dem auch die vielen kleinen, in den Kellern von Privatpersonen stehenden Mini-BHKWS gehören, übernimmt die von Lichtblick entwickelte Software »SchwarmDirigent« (vgl. von Petersdorff 2013). Es gibt mittlerweile einen technischen Standard für virtuelle Kraftwerke (siehe 7.1.3.2) und weitere Softwareprodukte (bspw. von Next Kraftwerke[47]), mit denen EVU unterschiedliche Erzeugungsanlagen zusammengeschalten können.

7.2.3. Softwaregestaltende: gesteigerte Interdisziplinarität und Intervention Betriebsrat

Die Energiewirtschaft verlangt von Softwaregestaltenden im besonderen Maße, interdisziplinär zu arbeiten, und der Organisationsgrad der Branche, dass sie sich mit Betriebsräten arrangieren.

Der Anteil der Akademiker:innen steigt und es muss intensiv interdisziplinär gearbeitet werden. Neben den komplizierten energietechnischen Anlagen (Netze, Kraftwerke etc.) sorgen die ständigen Änderungen der Regulierung dafür. Bei deren Übersetzung in Algorithmen müssen juristische Texte interpretiert werden, energiewirtschaftliches Wissen und Kompetenzen in der Softwareentwicklung vorhanden sein. Wie kompliziert die Umsetzung der Regulierung ist, zeigt sich daran, dass die Verbände für größere Gesetze und Verordnungen immer Anwendungshilfen herausgeben[48]. Obwohl die EVU selbst Fachleute haben, um zwischen gesetzlicher Regulierung und Software zu übersetzen, ziehen sie und Softwarefirmen für die Programmierung bei Bedarf zusätzlich Jurist:innen zurate, wofür es spezialisierte Rechtsanwaltskanzleien gibt (bspw. Becker Büttner Held). So sparen sie Zeit, weil nicht immer das Know-how vorhanden ist, um juristische Texte zu verstehen, und sie gehen sicher, nichts falsch zu machen.

Wenn auch der hohe Organisationsgrad nicht den Personalabbau und Restrukturierung verhindern konnte (wie oben beschrieben), ist doch der Betriebsrat entscheidend dafür, eine mögliche digitale individuelle Leistungs- und Verhaltenskontrolle zu verhindern:

> »Inwiefern die Möglichkeiten der zunehmenden Transparenz für eine Leistungs- und Verhaltenskontrolle genutzt werden, hängt stark davon ab, ob es im Unternehmen einen Betriebsrat gibt.« (Roth 2018: 76)

7.3. Fazit: Software und Softwareentwicklung als Bausteine der Industrie-Governance

In der Stromwirtschaft gibt es für Software vielfältige Anwendungsbereiche in den Wertschöpfungsstufen Vertrieb, Netz, Handel und Erzeugung. Die Besonderheiten des Softwareeinsatzes sind, dass Software tragend für die Regulierung ist (bspw. Marktkommunikation), aus systemstabilisierenden Notwendigkeiten resultiert (bspw. Strommengen-

47 https://www.next-kraftwerke.de/unternehmen/technologie, abgerufen am 13.07.2023.

48 https://www.bdew.de/service/anwendungshilfen/, abgerufen am 13.07.2023.

bilanzierung) oder die Marktintegration von dezentralen Erzeugungslangen ermöglicht (bspw. virtuelle Kraftwerke).

Der Ansatz der Industrie-Governance zeigt, dass die Softwaregestaltung für alle Arenen relevant ist: Die Unternehmen müssen sich eine Strategie überlegen. Dabei steht die reine Anwendung einer Standardsoftware für viele EVU im Mittelpunkt. Es stehen viele Firmen zur Verfügung, die dafür Softwarelösungen anbieten. Die EVU sind dann von den Lebenszyklen der Softwarepakete abhängig. Gleichzeitig profitieren sie von softwaretechnischen Innovationen. Einige machen sich auf, selbst Software zu gestalten und zu programmieren – für sich und für andere. Genaue Zahlen dazu gibt es nicht. Für die Arenen der Produktmarkt- und Prozess-Governance ist Software entscheidend, um diese in der Form, wie sie die Regulierung vorgibt, umzusetzen: sei es für den Datenaustausch zwischen Firmen, den aus Sicht der Verbrauchenden unkomplizierten Wechsel des Stromlieferanten, die Strommengenbilanzierung, neue Geschäftsfelder wie virtuelle Kraftwerke oder die Vermarktung von Ökostrom. Für die Beschäftigten bedeutet der zunehmende Softwareeinsatz zunehmende Akademisierung und ruft die Betriebsräte auf den Plan, zumindest bei der individuellen Verhaltenskontrolle einzuschreiten. Oft ist die Umsetzung ohne Anwendende und allein mit Expert:innen möglich – ob für Regulierung oder Energietechnik (bspw. virtuelle Kraftwerke).

Software hilft, existierende Strukturen zu erhalten. Dabei gibt es sowohl in der Software- als auch der Stromindustrie keine Anzeichen für eine Monopolisierung: In der energiewirtschaftlichen Softwareindustrie gibt es einerseits noch viele Anbietende von Software und neue Verdienstmöglichkeiten vor allem für große EVU und die großen, kooperativen IT-DL durch Softwareentwicklung und den Betrieb von Anwendungsplattformen. Andererseits gibt es viele kleinere Softwarefirmen und auch kleinere EVU programmieren selbst oder haben gar eigene Softwareunternehmen. In der Stromindustrie bleiben die vielen kleinen Unternehmen erhalten und damit auch die kommunalen Strukturen, weil sich Standardprozesse und die umfangreiche Regulierung digital durch Standardsoftwarepakete und Anwendungsplattformen abwickeln lassen. Um die genannten Thesen auf soliden empirischen Boden zu stellen, fehlen allerdings die quantitativen Zahlen.

Unklar bleibt, welche Rolle die Spannung zwischen Profitorientierung und kommunaler Daseinsvorsorge beim Softwareeinsatz spielt. Es sei vorweggenommen, dass auch die Fallstudien das nicht aufklären können.

8. Formen und Folgen der Softwaregestaltung – die Empirie

Darstellung und Vergleich der Fallstudien

8.1. Einführung: Vorgehen und Kurzvorstellung der sieben Fallstudien

Die Forschungsarbeit beleuchtet, was zwischen Anwendung und Programmierung passieren muss, damit Firmen die Möglichkeiten industriespezifischer Softwareentwicklung für sich nutzen können, und welche Folgen dies hat. In den hier vorgestellten sieben Fallstudien lösen die Organisationen die beiden Kernprobleme der Softwaregestaltung (jene der softwaretechnischen Gestaltungsmöglichkeiten und der Interdisziplinarität) jeweils unterschiedlich, um mal gemeinsam einen Standard zu gestalten oder für sich eine individuelle Software herzustellen. Beides sind Wege, auf denen die EVU die Möglichkeiten der Softwaregestaltung für sich nutzen und ihre Effizienz steigern. Beides sind Wege, welche die Fallstudien nicht immer in ihrer Reinform darstellen. In einigen Fällen liegt eine Mischung zwischen Individual- und Standardsoftwaregestaltung vor. Darüber hinaus reorganisieren sich die EVU in unterschiedlichem Maße, aber nie vollständig, um die Möglichkeiten der Softwaregestaltung zu nutzen. Nur in einem Fall ist eine Organisation von Anfang an auf die Softwaregestaltung ausgerichtet und schöpft ihre Möglichkeiten voll aus: STARTUP gestaltet für sich eine individuelle Software und bietet diese als (Standard-)Produkt anderen Organisationen zur Anwendung an. Bei ihr gilt der Primat der Softwareentwicklung (4.1) und die Anwendung ist nicht nur von einer Software, sondern auch von einer kontinuierlichen Softwaregestaltung abhängig und den Bedarfen der Softwaregestaltung untergeordnet, damit diese ihre Möglichkeiten voll ausschöpfen kann.

Dank der Fallstudien konnte die Forschungsarbeit für die beiden Konzepte der soziotechnischen Netzwerkarbeit (Kontrolle der Softwaregestaltung) und soziotechnischen Arbeitsgestaltung (Verhältnis von Softwaregestaltung zur Softwareanwendung) einen **kategorienbasierten Analyserahmen** ausarbeiten. Zudem hat die Analyse der Fallstudien ergeben, dass es wichtig ist, zwischen Konstellation, Softwaregestaltung und Folgen für Softwaregestaltende und Softwareanwendung zu unterscheiden.

Die **soziotechnische Netzwerkarbeit** als bestehend aus vier Ebenen zu konzeptionieren, war ein erster, vereinfachender Schritt (siehe 6. Kapitel). Nun kommen für die Softwaregestaltung spezifische Kategorien hinzu und die Differenzierung zwischen Konstellation, Arbeitsprozess und Arbeit der Softwaregestaltenden, um die Zusammenhänge zwischen diesen zu untersuchen. Dabei setzt sich die soziotechnische Netzwerkarbeit aus dem Arbeitsprozess der Softwaregestaltung und der Arbeit der Softwaregestaltenden zusammen.

Für die **soziotechnische Arbeitsgestaltung** geht es um die Zusammenhänge zwischen Konstellation und Verhältnis von Softwaregestaltung zur Softwareanwendung und welchen Einfluss Gestaltung und Anwendung aufeinander haben und welche Konflikte zwischen ihnen bestehen.

Der Analyserahmen besteht aus vier Teilen:

1. Der Teil der **soziotechnischen Konstellation** beschreibt die Bedingungen, unter denen im jeweiligen Fall die Organisationen die Möglichkeiten der Softwaregestaltung nutzen. Sie ist deshalb ein eigenständiger Teil des Analyserahmens, weil die Softwaregestaltung auszeichnet, dass sie in sehr unterschiedlichen Kontexten stattfindet. Die soziotechnische Konstellation hat Folgen dafür, wie die Softwaregestaltung stattfindet, die Softwaregestaltenden arbeiten und welche und wie weitgehend Firmen die Möglichkeiten der Softwaregestaltung nutzen können.
2. Der **Arbeitsprozess der Softwaregestaltung** selbst findet unter diesen Bedingungen der soziotechnischen Konstellation statt und zeigt, wie die Organisationen diesen Arbeitsprozess kontrollieren. Was müssen die Organisationen und Arbeitenden können, um eine individuelle oder eine Standardsoftware zu gestalten? Er findet entweder dezentral in den softwareanwendenden Organisationen statt oder zentralisiert in einer Organisation, die für mehrere anwendende Organisationen eine Software gestaltet.
3. Für die **Arbeitsbedingungen der Softwaregestaltenden** sind soziotechnische Konstellation und Arbeitsprozess der Softwaregestaltung mehr oder weniger förderlich. Entweder arbeiten sie in einer Matrixorganisation, d.h., die Softwaregestaltung ist Teil einer Hierarchie oder einer Marktbeziehung zwischen Organisationen. Oder sie arbeiten in einem reinen organisationalen Netzwerk, in dem weder formale Hierarchie noch Markt die Softwaregestaltung beeinflussen. Unabhängig davon, ob es sich um eine Matrixorganisation oder ein reines Netzwerk handelt, stellen die Softwaregestaltenden in allen Fallstudien eine eigenständige Beschäftigtengruppe dar, die hinsichtlich des Beschäftigungssystems, der Kontrolle und der Wissensverteilung ihre Besonderheiten aufweist.
4. Zuletzt hat es für die **soziotechnische Arbeitsgestaltung der Softwareanwendung** Folgen, wie die Fallstudien die Möglichkeiten der Softwaregestaltung nutzen. Sie hat Folgen vor allem dahingehend, ob die softwareanwendenden Organisationen die Software unabhängig gestalten können oder abhängig sind von einer Softwarefirma, einem IT-Dienstleistungsunternehmen (IT-DL) oder anderen EVU. Es geht darum, wie die Organisationen die Möglichkeiten der soziotechnischen Arbeitsgestaltung durch die Softwaregestaltung nutzen. Dabei geht es um das Verhältnis der beiden Arbeitsprozesse: Welchen Einfluss haben die Arbeitsprozesse von Softwaregestaltung

und Softwareanwendung aufeinander und welche Konflikte bestehen zwischen beiden?

Als Ergebnis veranschaulicht der Schluss des Kapitels die Unterschiede der Fallstudien in puncto soziotechnischer Netzwerkarbeit und soziotechnischer Arbeitsgestaltung jeweils anhand von vier Idealtypen. Die vier Typen der soziotechnischen Netzwerkarbeiten machen deutlich, dass vor allem der Koordinationsaufwand, aber auch die Möglichkeiten, einen Standard basierend auf einem breiten Konsens zu etablieren, sich je nach Netzwerk unterschieden. Jene vier Typen der Arbeitsgestaltung verdeutlichen dabei noch einmal, dass Organisationen die Möglichkeiten der Softwaregestaltung dann voll ausschöpfen, wenn sie individuell für sich eine industriespezifische Software gestalten und diese gleichzeitig als Standardsoftware anderen Firmen anbieten. Vor allem zeigt sich, dass es um soziotechnische Möglichkeiten geht: nicht nur um individuelle Softwaregestaltung, sondern auch um eine individuelle Gestaltung einer gesamten Organisation, bei der die gestaltete und angewendete Software im Zentrum steht.

Zudem stellt der Schluss einen neuen Rationalisierungstyp vor, der sich aus den Gemeinsamkeiten der Fallstudien ergibt. Kapitel 6 hat ihn bereits theoretisch von anderen abgegrenzt und als Typ der technikentwicklungsbezogenen Rationalisierung bezeichnet. Er basiert auf der Arbeitsteilung von Softwareanwendung, -gestaltung und -programmierung, ist rollen- und softwarebasiert und hat die Kommunikation und den Wissensaustausch zwischen Anwendung und Programmierung als Gegenstand. Mit ihm wird noch einmal das Argument unterstrichen, dass, indem der Arbeitsprozess der Softwaregestaltung Arbeit und Organisation in EVU gestaltet, er Anwendungsbereiche rationalisiert. Unabhängig davon, welche Form die Softwaregestaltung in den einzelnen Fallstudien annimmt: Es zeigen sich doch immer die gleichen Elemente einer technikentwicklungsbezogenen Rationalisierung.

Zuerst führt das Kapitel kurz in die sieben Fallstudien ein: Wie lösen sie jeweils die Kernprobleme der Softwaregestaltung von softwaretechnischen Gestaltungsmöglichkeiten und Interdisziplinarität? Inwiefern gestalten sie einen Standard oder eine individuelle Software? Zuletzt stellt die Einleitung des Kapitels den Analyserahmen vor und zeigt, wie er mit den Konzepten der soziotechnischen Netzwerkarbeit und der soziotechnischen Arbeitsgestaltung (beide aus Kapitel 6) in Verbindung steht. Die nächsten vier Abschnitte behandeln dann je einen Teil des Analyserahmens, für den sie jeweils die sieben Fälle detaillierter darstellen. Zudem fassen die vier Abschnitte Gemeinsamkeiten und Unterschiede zwischen den Fallstudien zusammen. Der letzte der vier Abschnitte bzw. Teil des Analyserahmens zur soziotechnischen Arbeitsgestaltung diskutiert darüber hinaus drei Thesen: Inwiefern handelt es sich um einen intervenierenden Betriebsrat? Inwieweit kann von einer direkten Partizipation der Anwendenden gesprochen werden? Inwieweit bewegen sich EVU auf einen neuen Typus von Prozessorganisation zu? Das Fazit dieses Kapitels fasst die Ergebnisse zusammen, stellt Bezüge zu bestehenden Forschungsarbeiten aus Kapitel 6 her, geht auf den Typ der technikentwicklungsbezogenen Rationalisierung und auf zwei Punkte gesondert ein: inwiefern die Softwaregestaltung eine Konkurrenz zum Management darstellt und ob es eine industriespezifische Softwaregestaltung gibt.

Um die Fallstudien besser zu verstehen, haben bereits vorhergehende Kapitel in Scrum (5.2.4) und die Entwicklungsplattformen von SAP (4.3) eingeführt. Am Ende der Kurzvorstellung der sieben Fallstudien steht eine kurze Erläuterung zur Holokratie, die bei der Fallstudie STARTUP eine Rolle spielt. Sonstige Fachbegriffe sind in Fußnoten erklärt.

8.1.1. Kurzvorstellung der Fallstudien: Wie sie die Kernprobleme der softwaretechnischen Gestaltungsmöglichkeiten und Interdisziplinarität lösen

Zur Einführung stellt der nächste Abschnitt die sieben Fallstudien kurz vor. Dabei steht im Fokus, wie jeder Fall die Kernprobleme der softwaretechnischen Gestaltungsmöglichkeiten und Interdisziplinarität löst und ob sie eher die Möglichkeiten einer individuellen oder einer Standardsoftware nutzen. Zudem nennt er die Eckdaten der Fälle: Um welche Unternehmen handelt es sich, welche Software entwickeln sie, welchen Anwendungsbereich hat die Software, wer sind die Anwendenden, Programmierenden und Softwaregestaltenden? Die unterschiedlichen Fälle werden dann die nächsten Abschnitte differenzierter und detaillierter mithilfe der vier Teile des Analyserahmens und ihrer jeweiligen Kategorien analysieren.

8.1.1.1. INTERN1: erweitertes Scrum für die mobile Auftragssteuerung der Netz-Instandhaltung

Die Softwaregestaltung findet in dem Fall innerhalb eines EVU statt. Wie löst das EVU das Problem der softwaretechnischen Gestaltungsmöglichkeiten? Es ist softwaretechnisch sowohl auf die Anwendung als auch auf die Entwicklung von Software ausgerichtet. Weil die Softwaregestaltung parallel zur bestehenden Organisation des EVU besteht, existiert eine Matrixorganisation. Iterativ übergeben Rollen wie Anforderungsmanagende und Product Owner:innen im Scrum-Ablauf die Anforderungen aus dem anwendenden Fachbereich an die programmierende IT-Abteilung. Der softwaretechnische Zuschnitt ist individuell, weil das EVU zwar die ERP-Software von SAP erweitert, es die mobile Lösung für die Instandhaltung aber allein für den Fachbereich gestaltet. Die Interdisziplinarität besteht in diesem Fall aus einem Netzwerk der Softwaregestaltung zwischen IT-Abteilung und dem Fachbereich Instandhaltung.

Damit gelingt dem EVU etwas, was anderen misslingt: Es kann intern einen funktionierenden, netzwerkförmigen, kommunikativen Arbeitsprozess für die Softwaregestaltung quer zu bestehenden Abteilungen und Teams etablieren. Es fehlt weder an softwaretechnischem Know-how, noch vereiteln Kommunikationshürden wie Abteilungssilos, hierarchische Konflikte oder Widerstände in den Fachabteilungen die Gestaltung der Software. Einzig die Frage ist offen, wer zukünftig die Hoheit über die Softwaregestaltung hat: der Fachbereich oder die IT-Abteilung.

Das betreffende EVU gehört zu den größeren in Deutschland. Es deckt die gesamte Wertschöpfung der Branche mit unterschiedlichen Tochterfirmen ab. Neben einer zentralen IT gibt es für die Geschäftsfelder Vertrieb und Verteilnetz eigene IT-Abteilungen. Bei dem Unternehmen handelt es sich um einen Verteilnetzbetreiber (VNB) für Strom und Gas, das Teil eines Konzerns ist und mehrere Tausend Mitarbeiter hat. Der Anwen-

dungsbereich der mobilen Lösung ist die Netz-Instandhaltung mit 600 Monteur:innen und 100 Dispatchenden. Das EVU engagiert externe Programmierende. Für den Arbeitsprozess der Softwaregestaltung existiert ein angepasstes Scrum mit Scrum Master und mehreren Product Owner:innen aus der IT-Abteilung und mehreren Anforderungsmanagenden und Key User:innen in den Fachbereichen.

Tabelle 8: Steckbrief Fallstudie INTERN1

Allgemeine Eckdaten	Unternehmen	VNB Strom und Gas (ca. 4000 MA), Teil eines Konzerns
	Anwendungsbereich	Instandhaltung Netze
	Software	Mobile Lösung für Monteure und Auftragsverarbeitung für Dispatcher (beides integriert in ERP-System)
	Anwendende	600 Monteure, 100 Dispatcher, Strom und Gas, alle intern
	Programmierende	8 für die mobile Lösung (Backend[1], Frontend, Middleware), 2 für die ERP-Integration der Auftragsverarbeitungssoftware; Teil der IT-Abteilung, 4 davon sind Externe
Softwaregestaltende	Key User:innen	52 für Monteure, 4 für Dispatcher
	Product Owner:innen	4 (einer davon führend), Teil des Innovationsbereichs der IT-Abteilung
	Anforderungsmgmt.	5 bis 6 aus dem Fachbereich
	Scrum Master	1 (Teamleiter)
	Ausgewählte Anwendende	für Tests, Resonanzgruppen, Workshops und andere Treffen

8.1.1.2. INTERN2: zentrale Anforderungsrunde mehrerer Fachbereiche für die Auftragsverarbeitung

Auch bei INTERN2 findet die Softwaregestaltung innerhalb eines EVU statt. Auch in diesem Fall löst das EVU das Problem der softwaretechnischen Gestaltungsmöglichkeiten damit, dass es sich softwaretechnisch nicht komplett reorganisiert, sondern ergänzend zu bestehenden Strukturen der Softwareanwendung den Arbeitsprozess der Softwaregestaltung etabliert. In dieser Matrixorganisation gestaltet das EVU individuell eine Software. Aber in diesem Fall tut es dies abteilungsübergreifend und anders als bei INTERN1 für mehrere Fachbereiche, die sich regelmäßig in einer Anforderungsrunde treffen. Dabei erweitert das EVU die SAP-ERP-Software nicht um eine mobile App, sondern passt ein von SAP geliefertes Modul an. Hier besteht das interdisziplinäre Netzwerk der Softwaregestaltung aus der IT-Abteilung und den dortigen Programmierenden, mehreren Fachbereichen und Abteilungen, von denen einzelne eigene, kleine IT-Teams haben.

1 Datenzugriffsschicht einer Software (Backend), im Gegensatz zur Präsentationsschicht (Frontend) als Quellcode zur Darstellung der Daten. Eine Middleware ist eine Software, die den Datenaustausch zwischen zwei ansonsten nicht verbundenen Softwarekomponenten ermöglicht.

Der Fall zeigt, wie es einem größeren EVU gelingt, einen Kommunikationsprozess zu etablieren, der eine ganzheitliche, abteilungsübergreifende Betrachtungsweise eines Prozesses erlaubt. Für diese abteilungsübergreifende Softwaregestaltung ist die Kooperation mehrerer Firmenbereiche notwendig, die an einem digitalen Prozess arbeiten und sich die dafür zuständigen Programmierenden teilen.

Das betreffende EVU ist eines der größten in Deutschland. Es deckt die gesamte Wertschöpfung der Branche mit unterschiedlichen Tochterfirmen ab. Neben einer zentralen IT hat der Netzbereich seine eigene IT-Abteilung. Die Tochterfirma für den Netzbetrieb Strom und Gas hat mehrere Tausend MA. Der Anwendungsbereich betrifft die Auftragsverarbeitung bzw. das Work-Management, wie das SAP-Modul heißt. Mehrere Fachbereiche nutzen es. Für die Softwaregestaltung existieren Scrum Master, Key User:innen und Product Owner:innen in den unterschiedlichen Fachbereichen. Dort gibt es auch eigene kleine IT-Teams. Die Abstimmungsrunde bearbeitet ca. 10 Tickets pro Woche. Die Programmierenden sind größtenteils externe Freelancer, welche die IT-Abteilung steuert.

Tabelle 9: Steckbrief Fallstudie INTERN2

Allgemeine Eckdaten	Unternehmen	VNB für Strom und Gas (ca. 4000 MA), Teil eines Konzerns
	Anwendungsbereich	Work-Management (Auftragsverarbeitung): betrifft mehrere Fachbereiche (u.a. Netzanschluss, Zählerwesen, Abrechnung)
	Software	Teil der ERP-Software von SAP
	Anwendende	keine genauen Zahlen vorhanden, aber auf jeden Fall mehr als 300 Anwendende für das Work-Management
	Programmierende	Programmierende des Work-Managements: 2 Interne, 7 Externe
Arbeitsprozess Softwaregestaltung	Abstimmungsrunde	bestehend aus den betroffenen Fachbereichen (u.a. Netzanschluss, Zählerwesen, Abrechnung), der IT des Zentralbereichs, Programmierende und Scrum Master
	Product Owner:innen/ Anforderungsmgmt.	mehrere in den Fachbereichen und der IT der Abteilung Zentralbereich
	Scrum Master	1
	Key User:innen	20–30 im Fachbereich Netzanschluss (für die anderen FB liegen keine Zahlen vor)
	Tickets	pro Woche min. 10 Tickets in der Abstimmrunde, ca. 300 im Backlog

8.1.1.3. KOOP1: kooperativ verhandelte, industriespezifische Erweiterung und Anpassung einer Standard-ERP-Software

In dem Fall überlassen die EVU den größten Teil der SAP-Erweiterung und -Anpassung einem IT-DL. Wobei sie im Zuge des vom IT-DL organisierten Anforderungsmanage-

ments kooperativ über den Zuschnitt der Software verhandeln: was sie in einen gemeinsamen Standard aufnehmen und was einzelne EVU individuell gestalten. Die interdisziplinären Netzwerke der Softwaregestaltung erstrecken sich zwischen IT-DL und mehreren anwendenden EVU, wobei die größeren EVU auch noch eigene IT-Abteilungen haben und selbst Software gestalten.

In dem Fall gelingt es den beteiligten Organisationen, die widersprüchlichen Ziele von gemeinsamen Synergien und individueller Gestaltung zu verhandeln. Dabei setzt das IT-DL einen Mediator ein:

> »Ich glaube, eines der größten Learnings ist, dass die größten Herausforderungen, einer der großen Kostenpunkte, die Abstimmung überhaupt und den Austausch über den Kunden (unv.). Also dieses ganze Anforderungsmanagement und alles was nichts mit Technik zu tun hat, […] sondern wirklich miteinander reden und abstimmen und irgendwie sich auf einen gemeinsamen Nenner zu einigen. Dass das eigentlich die größten Hürden sind und das über Jahre hinweg irgendwie so am Laufen zu halten. Dafür haben wir eine Lösung gefunden, auch sogar irgendwann extern moderiert und auch immer noch. Das tut auch gut, wenn ein neutraler Externer dabei ist, der zwischen Kunden und auch uns als Dienstleister irgendwie vermittelt und irgendwie schaut, dass einem da gerecht wird.« (Digitalisierungsmanager IT-DL)

Tabelle 10: Steckbrief Fallstudie KOOP1

Allgemeine Eckdaten	Unternehmen	mehrere EVU (vor allem Netz, Vertrieb, Strom, Gas) und IT-DL
	Anwendungsbereich	sämtliche energiewirtschaftlichen Kernprozesse (Marktkommunikation, Abrechnung, Geräteverwaltung etc.)
	Software	SAP-ERP-Standardsoftware, industriespezifisch erweitert
	Anwendende	mehrere Tausend in unterschiedlichen EVU
	Programmierende	15–20 für den industriespezifischen Teil des ERP beim IT-DL, vereinzelt Programmierende in EVU (z.B. 3 bei EVU2)
Softwaregestaltende	Key User:innen	verteilt auf die EVU
	Key Account Managende	mehrere des IT-DL, zuständig für einzelne EVU
	Anforderungsmgmt.	mehrere im IT-DL und den EVU
	IT-Projektmanagende	7 beim IT-DL (2 auf Tests spezialisiert), vereinzelt bei EVU
	IT-Beratende	28–30 für den industriespezifischen Teil des ERP beim IT-DL
	Prozessmanagende	bei zwei der befragten EVU
	IT-Koordinierende	bei einzelnen EVU
	Digitalisierungsmanager	bei einem befragten EVU
	Applikationsbetreuer	bei einem befragten EVU

Die Gesellschaftenden des IT-DL sind mehrere EVU. Die durch das IT-DL betriebene und weiterentwickelte ERP-Software bildet sämtliche Kernprozesse der EVU ab (Marktkommunikation, Abrechnung, Geräteverwaltung etc.). Die Anwendenden und Key User:innen arbeiten in den EVU. Dort findet vereinzelt auch Softwaregestaltung statt (z.B. durch Applikationsbetreuende oder IT-Koordinierende). Programmierende, IT-Beratende, IT-Projektmanagement und Key Account Managende sind beim IT-DL angestellt.

8.1.1.4. KOOP2: prekäre Kooperation für eine industriespezifische Erweiterung und Anpassung einer Standard-ERP-Software

Wie auch bei KOOP1 lösen das Problem der softwaretechnischen Gestaltungsmöglichkeiten mehrere Organisationen. Nur gibt es nicht das eine institutionalisierte Treffen, in dem sich EVU darüber verständigen, welche anstehenden Anforderungen an die IT sie gemeinsam umsetzen und welche nicht. In diesem Fall überlassen die EVU nur einen kleinen Teil der SAP-Erweiterung und -Anpassung einem IT-DL. Den anderen Teil übernehmen die EVU (vor allem die größeren) selber. Das geht so weit, dass sie selbst Softwaregestaltungsprojekte zusammen mit anderen EVU machen. Über den Zuschnitt der Software der EVU entscheidet damit nicht zentral eine Organisation: Es gibt Standard-Erweiterungen und -Anpassungen der SAP-Software durch das IT-DL, durch einzelne EVU und auch Softwaregestaltung mit anderen Softwarefirmen. Die interdisziplinären Netzwerke der Softwaregestaltung erstrecken sich zwischen IT-DL und mehreren EVU. Die größeren EVU haben in diesem Fall eigene IT-Abteilungen.

Der Fall zeigt, dass sich die beteiligten EVU immer mehr aus einer Kooperation lösen, wenn die zentrale Steuerung der Softwaregestaltung nicht gelingt. Im Zeitverlauf wandern koordinative und operative Aufgaben zwischen EVU und IT-DL hin und her. Die uneinheitliche Lage drückt sich darin aus, dass es schwierig war, sich durch die Interviews einen Überblick zu verschaffen: welches EVU nun was selber macht (ob Anwendung, Programmierung oder Softwaregestaltung) oder inwieweit sich die industriespezifische Standardsoftware von EVU zu EVU unterscheidet.

Das IT-DL ist im Eigentum einiger EVU und bietet sowohl Softwareentwicklung als auch -anwendung (inkl. Business Process Outsourcing) an. Dabei soll es nicht nur für die gesellschaftenden Unternehmen Aufgaben übernehmen, sondern mit der Betreuung anderer EVU Geld verdienen. Kernanwendungsbereich der Software ist die Datenverarbeitung für die energiewirtschaftlichen Geschäftsprozesse (Marktkommunikation, Kund:innenservice, Energiemengenbilanzierung etc.). IT-Beratende und Programmierende arbeiten für das IT-DL. Größere EVU haben ihre IT-Projektleitenden, IT-Koordinierenden, Key User:innen und Prozessmanagenden und arbeiten mit unterschiedlichen Softwarefirmen direkt zusammen.

Tabelle 11: Steckbrief Fallstudie KOOP2

Allgemeine Eckdaten	Unternehmen	IT-DL, EVU, diverse Softwarefirmen
	Anwendungsbereich	Datenverarbeitung Geschäftsprozesse EVU
	Software	SAP-ERP-Software, Softwarelösungen anderer Firmen: Marktkommunikation, Kund:innenservice und Netzleitstelle
	Anwendende	verteilt auf mehrere EVU und beim IT-DL
	Programmierende	vereinzelt in EVU, beim IT-DL, Freelancer, Softwarefirma
Softwaregestaltende	Key User:innen, Prozessmanagende	EVU2: ERP, Marktkommunikation, Kund:innenservice Netz
	Teamleitung Marktkommunikation	EVU2 Netzbetrieb
	IT-Projektleitung	EVU2 Netzbetrieb
	IT-Koordinierende Fachbereich	EVU2: zuständig für ca. 60 Mitarbeitende Netzbetrieb
	Manager Digitalisierung	EVU3: Vertrieb
	IT-Beratende	IT-DL

8.1.1.5. PAKET: industriespezifische ERP-Standardsoftware entwickelt durch eine Softwarefirma

Die EVU der Fallstudie lösen das Problem der softwaretechnischen Gestaltungsmöglichkeiten damit, dass sie ihre Organisation auf die Anwendung einer Standardsoftware ausrichten und den Standardsoftware-Zuschnitt der Softwarefirma überlassen. Ausgewählte EVU und ihre Expert:innen nehmen an Arbeitskreisen und Projekten mit der Softwarefirma teil. Diese interdisziplinären Netzwerke ergänzen die in der Softwarefirma vorhandene Interdisziplinarität, die notwendig ist, damit sie die industriespezifische Standardsoftware entwickeln kann.

Damit ist, anders als in den anderen Fallstudien zuvor, die entwickelte Software eine Standardware, bei der die Softwarefirma auf Skalierung setzt und die EVU vorwiegend auf Kosteneffizienz und Auslagerung der Softwareentwicklung. Dennoch ist die Softwarefirma auf das Feedback und den Input der EVU zur Qualitätssicherung und Weiterentwicklung des Standards angewiesen.

In dem Fall geht es um eine Softwarefirma und mehrere EVU. Die Software bildet die energiewirtschaftlichen Kernprozesse wie Marktkommunikation, Abrechnung oder Energiedatenmanagement ab. An der Softwaregestaltung sind im wesentlichen Key User:innen, Fachexpert:innen (via Arbeitskreise), Anwendungsbetreuende, Prozessmanagende oder Führungskräfte der EVU sowie Fachexpert:innen, IT-Projektleitende, Führungskräfte und IT-Beratende der Softwarefirma beteiligt. Die Programmierenden arbeiten allesamt in der Softwarefirma. Die Anwendenden sitzen in den EVU, wobei die Softwarefirma auch BPO (Business Process Outsourcing) anbietet und dafür eigene Anwendende beschäftigt.

Tabelle 12: Steckbrief Fallstudie PAKET

Allgemeine Eckdaten	Unternehmen	eine Softwarefirma, mehrere EVU
	Anwendungsbereich	energiewirtschaftliche Kernprozesse wie Marktkommunikation, Abrechnung, Energiedatenmanagement
	Software	ERP-Lösung inklusive industriespezifischer Prozesse
	Anwendende	verteilt auf mehrere EVU, bei EVU6 z.B. ca. 380 Anwendende
	Programmierende	konzentriert auf eine Softwarefirma, genaue Anzahl unbekannt
Softwaregestaltende	Key User:innen	verteilt auf diverse EVU
	Fachexpert:innen	verteilt auf diverse EVU und innerhalb Softwarefirma
	IT-Projektleitende	in EVU oder Softwarefirma
	IT-Beratende	mehrere der Softwarefirma
	Anwendungsbetreuung	in manchen EVU
	Prozessmanagende	in manchen EVU
	Führungskräfte	in manchen EVU an Softwaregestaltung beteiligt

8.1.1.6. KOOP3: Ko-Produktion einer IoT-Software für stadtwerksnahe Anwendungen

Wie auch bei PAKET richten sich in dem Fall die EVU drauf aus, die Kernmodule der IoT-Software anzuwenden. Sie überlassen den softwaretechnischen Zuschnitt des Standards der Softwarefirma. Das Problem der softwaretechnischen Interdisziplinarität in Bezug auf die energiewirtschaftlichen Anwendungsfälle löst KOOP3 dadurch, dass das IT-DL die Rolle des Vermittlers in die Energiewirtschaft übernimmt. Es führt die Implementierungsprojekte in den EVU durch und gibt bei deren Durchführung aufkommende Anforderungen an die Softwarefirma weiter.

Im Gegensatz zu den Fällen von KOOP1, KOOP2 und PAKET hat die Kooperation stark informellen Charakter (rudimentäre Verträge, flache Hierarchien, Konflikte werden persönlich gelöst). Weder Marktmechanismen noch hierarchische Befehlsketten oder Abteilungsgrenzen stellen Hemmnisse dar, um das notwendige Wissen aus dem Anwendungsbereich mit jenem der Programmierung zusammenzubringen.

Die beteiligten Unternehmen sind die IoT-Softwarefirma, das IT-DL und die EVU. Schwerpunkt der Anwendungsfälle ist die Installation von Sensoren. Die IoT-Lösung liest diese über LoRaWan[2] aus, stellt ihre Daten über Schnittstellen zur Verfügung, so dass verschiedene Anwendungen der EVU die Daten weiterverarbeiten bzw. darstellen können. Die Programmierenden arbeiten in der Softwarefirma und zum Teil beim IT-DL oder in EVU, wenn diese die Standardsoftware um eigene Module erweitern. Für die Softwaregestaltung gibt es IT-Projektmanagende in der Softwarefirma, dem IT-DL

2 Form der kabellosen Datenübertragung in einem Netzwerk, die sich durch geringen Energiebedarf auszeichnet.

und den EVU. Für die Zusammenarbeit mit der Softwarefirma hat das IT-DL einen extra Product Owner.

Tabelle 13: Steckbrief Fallstudie KOOP3

Allgemeine Eckdaten	Unternehmen	IoT-Softwarefirma, IT-DL, EVU
	Anwendungsbereich	IoT, LoRaWan
	Software	Datenplattform: Daten diverser Sensoren sammeln; bietet Schnittstellen zu anderen Systemen
	Anwendende	keine konkreten Anwendenden, weil diverse Anwendungen die gesammelten Daten verwenden können
	Programmierende	zentral in der Softwarefirma (ca. 15); IT-DL und manche EVU programmieren kleinere Lösungen, die sie über Schnittstellen mit der IoT-Software verbinden
Softwaregestaltende	IT-Projektmanagement	Softwarefirma, IT-DL, EVU
	Account Management	Softwarefirma
	Product Owner	IT-DL
	Teamleiter	EVU1

8.1.1.7. STARTUP: Primat der Softwareentwicklung für den digitalen Emissionshandel (E-Mobilität)

Das ist der einzige Fall, in dem die Organisation klar auf Softwareentwicklung ausgerichtet ist und ihre eigene Software herstellt. Damit löst STARTUP das Problem der softwaretechnischen Gestaltungsmöglichkeiten in diesem Fall mit dem Primat der Softwareentwicklung (siehe dazu 4.1). Die Anwendung der Software beschränkt sich allerdings nicht auf STARTUP selbst. Die entwickelte Software bietet die Firma anderen Unternehmen als Anwendungsplattform in Form einer White-Label-Software[3] an (z.B. Firmen der Automobilbranche). Die Interdisziplinarität besteht intern zwischen unterschiedlichen Rollen und es gibt keine Grenzen durch Abteilungen oder abgeschottete Teams.

Das Start-up organisiert den Quoten- bzw. Emissionshandel von E-Autos für einzelne Autobesitzer:innen und für Ladesäulen. Um die Emissionen im Verkehrssektor zu reduzieren, hat der Staat die Treibhausgasminderungsquote (kurz: THG-Quote) eingeführt. Unternehmen mit hohen Emissionen, wie z.B. die Mineralölindustrie, können ihre Quoten erfüllen, indem sie über den Quotenhandel Zertifikate von Unternehmen erwerben, die emissionsarme Kraftstoffe für den Verkehr herstellen, wie z.B. Stadtwerke, die Strom z.B. über PV-Anlagen produzieren und über Ladesäulen anbieten. Die vom

3 Wie bereits unter 7.2.1.3 ausgeführt, handelt es sich bei einer White-Label-Software darum: »Eine Organisation entwickelt eine Software, stellt sie zur Verfügung (ob via Cloud oder On-Premises) und versieht sie in der Darstellung nach außen mit der entsprechenden Aufmachung des anwendenden EVU (bspw. Logo und Name Stadtwerk).«

Start-up entwickelte Software hat im Wesentlichen zwei Teile, die auch den zwei angebotenen Dienstleistungen entsprechen: einen für die Anmeldung von E-Autos (B2C) und einen für den Quotenhandel (B2B).

Die Programmierenden arbeiten alle für das Start-up. An der Softwaregestaltung nehmen vor allem all jene teil, die an den zwei Kreisen für Handel und Anmeldung zum Handel mitarbeiten, wozu die Unternehmensleitung, Product Owner, Solution Architect, aber auch Programmierende und andere Beschäftigte gehören.

Tabelle 14: Steckbrief Fallstudie STARTUP

Allgemeine Eckdaten	Unternehmen	organisiert Quotenhandel für E-Mobilität
	Anwendungsbereich	Quotenhandel: Anmeldung von E-Autos (B2C) und Handel mit diesen Zertifikaten, Handel mit Mengen aus Ladesäulen (B2B)
	Software	a) Webseite und App, um E-Autos anzumelden, b) Software für Quotenhandel
	Mitarbeitende insgesamt	ca. 20–25, ca. 13 Vollzeitäquivalent (schwankt im Befragungszeitraum)
	Anwendende	E-Auto-Besitzende, interne Mitarbeitende
	Programmierende	2 für das Modul Anmeldung, 3 für das Modul Zertifikate-Handel
Softwaregestaltung	Geschäftsführung	2
	Product Owner:innen	2
	Kreise	2 (je Produkt: Handel und Anmeldung)
	Solution Architect	1

Exkurs: Holokratie

In der Fallstudie STARTUP organisieren u.a. Kreise die Zusammenarbeit. Woher kommt die Idee von Kreisen? Der Verfasser führt dies auf das Organisationskonzept der Holokratie zurück. Es stammt wie das agile Manifest aus der Softwareentwicklung. Bei der Holokratie sind Rollen zentral und es soll eine hierarchiefreie Organisation geschaffen werden. Das Organisationskonzept soll hier zumindest erwähnt werden und der Exkurs auf die Kernelemente hinweisen, um den Fall STARTUP besser zu verstehen und zu zeigen, aus welchen unterschiedlichen Ansätzen sich Organisationen bedienen, um Softwaregestaltung zu organisieren.

Auf der Webseite www.holacracy.org sind auf der Seite Verfassung (https://www.holacracy.org/constitution/5) als Organisationsstrukturen aufgeführt:

- Role Definition
- Responsibility of Role Leads

- Circles
- Circle Leads

Dabei werden Kreise wie folgt definiert:

»A ›Circle‹ is a container for organizing Roles and Policies around a common Purpose. The Roles and Policies within a Circle make up its acting ›Governance‹.« (HolacracyOne 2023)

Die Befragten sprachen in den Interviews nie über »Policies«. Doch ist unverkennbar, dass in der Fallstudie STARTUP Rollen und Kreise zentral sind. Die Verantwortlichkeiten der Rollen verhandeln die Beteiligten untereinander. Diese Treffen werden in der Holacracy »governance meetings« genannt:

»Governance meetings help define how we will work together – they facilitate uncovering and assigning the roles needed to reach the circle's aim.« (Robertson 2007)

Kreise haben konkrete Zwecke (z.B. ein Modul der Software zu gestalten) und es gibt Regeln, nach denen sie arbeiten (z.B. wer Treffen organisiert). Obwohl die Anleihen aus dem Organisationskonzept der Holokratie unverkennbar sind, konnten die Befragten nicht bestätigen, dass ihre Organisation davon inspiriert ist. Manchmal sprechen sie von Teams, manchmal von Kreisen.

8.1.2. Unterschiedliche Möglichkeiten der Softwaregestaltung: zwischen Standard- oder Individualsoftware und Überblick über die Fallstudien

Letztlich lassen sich die Unterschiede zwischen den Fallstudien in Bezug auf die Nutzung der Möglichkeiten der Softwaregestaltung auf zwei diametral entgegengesetzte Typen reduzieren: Gestalten sie eine individuelle Software oder einen Standard? Welche Form der Softwaregestaltung am effizientesten für ein EVU ist, lässt sich nicht so einfach beantworten. Denn sowohl eine individuell auf die jeweiligen Bedürfnisse zugeschnittene Software kann effizient sein als auch eine Software, die viele EVU einsetzen und die einen Standard darstellt. Andererseits kann es unnötig hohe Kosten verursachen, etwas Individuelles zu gestalten, oder es kann ineffizient sein, sich auf eine Standardsoftware auszurichten, wenn eine Organisation dadurch vorher effiziente und optimierte interne Prozesse ändern muss.

In den Fallstudien liegt, was die gesamte IT-Landschaft der EVU betrifft, meistens eine Mischung aus industriespezifischen Standardlösungen oder -softwarebausteinen und individuell gestalteter Software vor. Für den Arbeitsprozess der Softwaregestaltung, der den Kern der jeweiligen Fallstudie bildet, lässt sich jedoch sagen, dass bei INTERN1, INTERN2 und STARTUP Unternehmen eine individuelle Software gestalten. Bei PAKET und KOOP3 gestalten Softwarefirmen eine Standardlösung. Bei KOOP1 ist ein kooperativer Standard das Ziel, wenn es auch individuelle Abweichungen gibt. Bei KOOP2 ist es gar nicht so einfach zu sagen, welches EVU einem gemeinsamen Standard folgt und

welches nicht, weil kein zentralisiertes Entscheidungsgremium dafür existiert. Das restliche Kapitel macht die Unterschiede zwischen den Fallstudien deutlicher und vertieft sie.

Die obige Kurzvorstellung der Fallstudien und wie sie die beiden Kernprobleme der Softwaregestaltung der softwaretechnischen Gestaltungsmöglichkeiten und der Interdisziplinarität lösen, sind in der untenstehenden Tabelle zusammengefasst. Wie ist die Tabelle zu lesen? Zum Beispiel ist bei INTERN1 das EVU durch eine Matrixorganisation aus Softwareanwendung (SA) und Softwaregestaltung (SG) auch auf die Softwaregestaltung ausgerichtet. Dabei besteht der Arbeitsprozess der Softwaregestaltung im Kern aus einem um einige Methoden erweiterten Scrum-Arbeitsprozess. Die Software hat einen individuellen Zuschnitt für den Fachbereich Instandhaltung und die softwaretechnische Interdisziplinarität ist in einem EVU integriert, weil beteiligte Softwaregestaltende, -programmierende und -anwendende Mitarbeitende eines EVU sind. Bei KOOP1 hingegen gibt es sowohl EVU, die nur anwenden, als auch diejenigen, die mitgestalten und dafür den entsprechenden Arbeitsprozess der Softwaregestaltung haben. Den Zuschnitt der Software verhandeln die EVU untereinander und die softwaretechnische Interdisziplinarität ist nicht in einer Organisation integriert, sondern verteilt sich auf EVU und IT-DL, auch wenn einige EVU über softwaretechnisches Wissen verfügen und das IT-DL tiefergehendes energiewirtschaftliches Wissen hat.

Tabelle 15: Überblick über die Fallstudien – softwaretechnische Gestaltungsmöglichkeiten und Interdisziplinarität

Fall	**Softwaretechnische Gestaltungsmöglichkeiten**		**Softwaret. Interdisziplinarität**
	softwaretechnische Ausrichtung anwendende Organisation (EVU)	**softwaretechnischer Zuschnitt**	**organisatorische und interpers. Netzwerke der Softwaregestaltung**
INTERN1	**SG**: Matrix Fachbereich – IT-Abteilung, erweitertes Scrum	**individuell** für einen Fachbereich	**integriert**: IT-Abteilung und ein Fachbereich
INTERN2	**SG**: Matrix Fachbereiche – IT-Abt., Anforderungsrunde	**individuell** für mehrere Fachbereiche	**integriert**: IT-Abteilung, mehrere Fachbereiche inkl. deren IT-Teams
KOOP1	**SA/SG**: organisationsübergr. Matrix EVU – IT-DL, zentrales Anforderungsmanagement	kooperativ verhandelt: **Standard – individuell** für mehrere EVU	**desintegriert**: IT-DL und mehrere EVU; Fachbereiche und IT-Abteilung innerhalb der EVU
KOOP2	**SA/SG**: organisationsübergr. Matrix EVU – IT-DL, verteiltes Anforderungsmanagement	verteilte Entscheidungshoheit: **Standard – individuell**	**desintegriert**: IT-DL und mehrere EVU; Fachbereiche und IT-Abteilung innerhalb der EVU

Fall	Softwaretechnische Gestaltungsmöglichkeiten		Softwaret. Interdisziplinarität
	softwaretechnische Ausrichtung anwendende Organisation (EVU)	**softwaretechnischer Zuschnitt**	**organisatorische und interpers. Netzwerke der Softwaregestaltung**
PAKET	**SA**: Anwendung Standardsoftware inkl. Einstellungen vornehmen	**Standard** für viele EVU	**desintegriert**: Arbeitskreise und IT-Projekte zw. Softwarefirma und EVU, intern in Softwarefirma
KOOP3	**SA**: Anwendung IoT-Software	**Standard**-Kernmodul IoT für viele EVU	**desintegriert**: IT-DL, Softwarefirma, EVU
START-UP	**SG:** Primat der Softwareentwicklung	**individuell** für die Organisation und als **Standard** für andere	**integriert**: intern, rollenbasiert (keine Abteilungsgrenzen), Kreise

Der Analyserahmen hilft, die soziotechnische Netzwerkarbeit der Softwaregestaltung in ihren Zusammenhängen und wesentliche Elemente zu untersuchen.

8.1.3. Der Analyserahmen

8.1.3.1. Die vier Teile und ihre Kategorien

Die vier Teile des Analyserahmens wurden bereits kurz in der Einleitung vorgestellt. Der folgende Abschnitt geht ausführlicher auf sie ein, indem er die jeweiligen Kategorien der Teile vorstellt.

1. Ausgangsbedingungen für die Softwaregestaltung (soziotechnische Konstellation)

Die soziotechnische Konstellation von Anwendung und Entwicklung bestimmt den Spielraum und die Grenzen, welche Möglichkeiten der Softwaregestaltung Organisationen überhaupt in einem Anwendungsbereich verwirklichen können. Indem die Analyse der Fallstudien die Ausgangsbedingungen der Softwaregestaltung berücksichtigt, kann sie zeigen, wie in verschiedenen Kontexten die Beteiligten Wissen austauschen und kommunizieren und welche Folgen der Kontext für den Arbeitsprozess der Softwaregestaltung, die Arbeit der Softwaregestaltenden und die soziotechnische Arbeitsgestaltung der Softwareanwendung hat. Die empirische Untersuchung hat vier zentrale Kategorien der soziotechnischen Konstellation ausgemacht: Anwendungsbereich der Software, Softwarearchitektur, Arbeitsteilung und die Grundkoordination zwischen Anwendung und Entwicklung (Gestaltung und Programmierung). Vom Anwendungsbereich hängt der Digitalisierungsbeitrag ab: Inwieweit kann Software zur Arbeit beitragen oder diese ersetzen? Welches energiewirtschaftliche Domänenwissen ist relevant und wie ist es verteilt? Die Softwarearchitektur prägt die Arbeitsteilung zwischen Anwendung und Programmierung: Welche technischen Gestaltungsmöglichkeiten gibt es außerhalb einer Softwarefirma (Schnittstellen, Anpassungsmöglichkeiten)? Wie ist die Software technisch aufgeteilt und wer gestaltet welchen Teil? Die Arbeitsteilung beschreibt, wie sich Arbeit und Wissen von Anwendung und Entwicklung verteilen und wie die Wis-

sensgrenzen verlaufen. Sind die Programmierenden Teil der IT-Abteilung oder einer Softwarefirma? Sind die Anwendenden Teil eines Fachbereichs eines EVU oder eines IT-DL? Von der Grundkoordination ausgehend müssen Organisationen die kooperative Zusammenarbeit zwischen Anwendung und Entwicklung etablieren. Arbeiten die beteiligten Teams, Abteilungen oder Organisationen primär via Markt (z.B. zwischen Softwarefirma und EVU), Hierarchie (z.B. innerhalb eines EVU) oder Netzwerk (z.B. projektförmig) zusammen?

2. Folgen der Softwaregestaltung für die Arbeit der Softwaregestaltenden

Wie die Softwaregestaltenden als Beschäftigtengruppe in einer Organisation arbeiten, hängt mit der soziotechnischen Konstellation und dem Arbeitsprozess der Softwaregestaltung zusammen. Dabei sind Beschäftigungssystem, Kontrolle und die Wissensverteilung Gegenstand der Analyse und wie sie sich unterscheiden in Fällen von Matrix- oder reinen Netzwerkorganisationen. Mit Beschäftigungssystem ist in erster Linie die Allokation von Arbeitskräften inklusive ihrer Karrierewege gemeint. Den Vergleich mit Anwendenden und Programmierenden nutzt die Zusammenfassung des Abschnitts, um die Eigenarten der Arbeit der Softwaregestaltenden zu verdeutlichen.

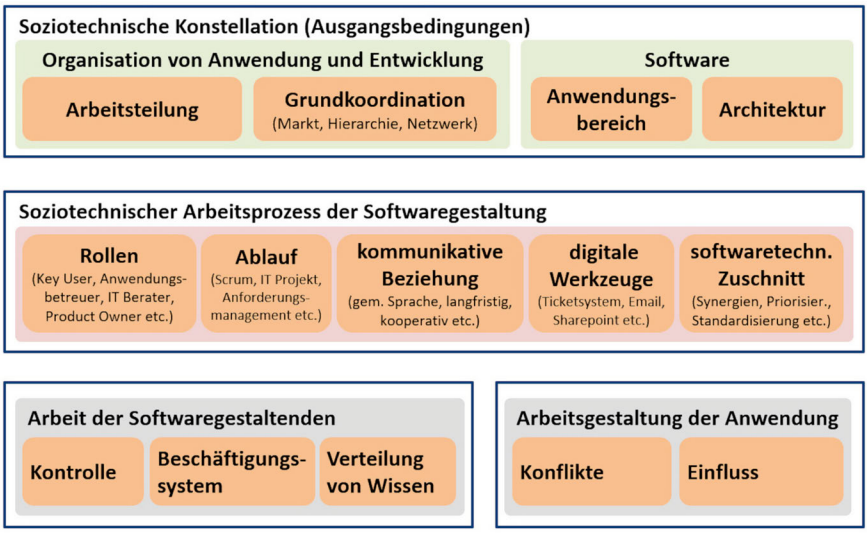
3. Folgen der Softwaregestaltung für die soziotechnische Arbeitsgestaltung der Softwareanwendung

Die Folgen für die soziotechnische Arbeitsgestaltung der Softwareanwendung ergeben sich im Verhältnis von Softwaregestaltung und Softwareanwendung. Erst in diesem Verhältnis zeigen sich Einfluss und Konflikte zwischen beiden. Zentrale Elemente dabei sind, inwiefern Anwendende an der Softwaregestaltung partizipieren, welche Ziele die EVU verfolgen (bspw. die Entwicklung einer Softwarefirma überlassen), ob die Softwaregestaltung die Softwareanwendung kontrollieren kann und ob eine Reorganisation stattfindet (bspw. damit ein EVU selbst Software entwickeln kann). Die soziotechnische Arbeitsgestaltung der Softwareanwendung durch die Softwaregestaltung ist damit nur ein Teil der Arbeitsgestaltung in EVU – allerdings kein unwichtiger. Denn die gestaltete Software prägt je nach Anwendungsbereich weitgehend Arbeitsinhalte und Arbeitsprozesse und vom Arbeitsprozess der Softwaregestaltung hängt die Organisation der anwendenden Organisation ab. Die Fallstudien unterscheiden sich dahingehend, wie unabhängig die EVU Software und damit die Arbeit der Anwendenden gestalten können.

Der Analyserahmen teilt die vier Ebenen der soziotechnischen Netzwerkarbeit von Ablauf, Beziehungen, Software und Softwaregestaltenden aus dem 6. Kapitel auf die drei Teile soziotechnische Konstellation, Arbeitsprozess der Softwaregestaltung und Arbeit der Softwaregestaltenden auf, um die Zusammenhänge zwischen ihnen zu untersuchen. Der Arbeitsprozess der Softwaregestaltung besteht aus den Kategorien Rollen, Ablauf, kommunikative Beziehungen, digitale Werkzeuge und softwaretechnischer Zuschnitt.

Der Rahmen verdeutlicht die soziotechnischen Strukturen der Fallstudien: Softwarearchitektur und Anwendungsbereich der Software als Teil der soziotechnischen Konstellation, softwarebasierte Werkzeuge und softwaretechnischer Zuschnitt als Elemente des Arbeitsprozesses der Softwaregestaltung.

Abbildung 11: Analyserahmen für die Formen und Folgen der Softwaregestaltung – soziotechnische Netzwerkarbeit und soziotechnische Arbeitsgestaltung



8.1.4. Was sind große, mittlere und kleine EVU?

In der Untersuchung steht zwar nicht im Fokus, welche Auswirkung die Organisationsgröße auf die Softwaregestaltung hat. Jedoch gibt es Auffälligkeiten, die mit der Größe der Firmen zu tun haben. Um diese einzuordnen, unterscheidet die Untersuchung der Fallstudien drei Größen:

- kleine EVU: weniger als 800 Mitarbeitende
- mittlere EVU: zwischen 800 und 5000 Mitarbeitende
- große EVU: über 5000 Mitarbeitende

Die Einteilung geht darauf zurück, welche EVU in den Fallstudien welche Auffälligkeiten zeigen (z.B. selbst Software zu gestalten, mitzugestalten oder nur Software anzuwenden). Das ist eine ganz andere Einteilung als jene des Statistischen Bundesamtes, das sämtliche EVU mit mehr als 250 Mitarbeitenden in eine Kategorie steckt (siehe 7.2.1.1). Eine andere Unterscheidung zwischen EVU wäre jene der BNetzA, die EVU mit weniger als 100.000 Kund:innen von einigen regulatorischen Vorgaben befreit (siehe 7.1.2.2.). Der Frage, welche Folgen diese Grenze für die Softwaregestaltung hat, konnte die Arbeit nicht nachgehen. Es scheint aber, dass jene mit weniger als 100.000 Kund:innen tendenziell auf andere Standardlösungen vertrauen (siehe 7.2.1.1).

8.2. Soziotechnische Konstellation als Ausgangssituation der Softwaregestaltung

Die beiden Kernprobleme der softwaretechnischen Gestaltungsmöglichkeiten und Interdisziplinarität lösen die EVU je Fallstudie unterschiedlich. Die folgenden Abschnitte vertiefen mithilfe des Analyserahmens und seiner Kategorien das Verständnis darüber, wie dies geschieht.

Der in 8.2 behandelte erste Teil des Analyserahmens nimmt die soziotechnische Konstellation (Anwendungsbereich, Arbeitsteilung, Grundkoordination, Softwarearchitektur, s.o.) in den Blick. Sie stellt die Ausgangsbedingungen und Grenzen des Arbeitsprozesses der Softwaregestaltung dar, in denen sich die Möglichkeiten der Softwaretechnik verwirklichen lassen. Sie legt bereits im Wesentlichen fest, ob Organisationen eine individuelle Software gestalten oder einen Standard, ob innerhalb eines EVU oder zentralisiert in einer Softwarefirma oder einem IT-DL. Nach der Falldarstellung stellt die Zusammenfassung noch einmal die Unterschiede und Gemeinsamkeiten der Fallstudien heraus und stellt kurz die Zusammenhänge mit den anderen Teilen des Analyserahmens dar. Auf die Folgen der dargestellten Ausgangsbedingungen gehen erst die darauffolgenden Punkte 8.3.ff. ein, die sich den anderen Teilen des Analyserahmens widmen.

8.2.1. Darstellung der Fallstudien

8.2.1.1. INTERN1: Erweiterung ERP-Software durch IT- und Fachabteilung zur Steuerung der Instandhaltung, hierarchisch

Welche Folgen haben in diesem Fall die Ausgangsbedingungen der soziotechnischen Konstellation von Arbeitsteilung, Grundkoordination, Softwarearchitektur und Anwendungsbereich? Die Arbeitsteilung trennt Anwendung und Programmierung klar voneinander. Die so bestehenden Wissensgrenzen zwischen den Abteilungen muss der Arbeitsprozess für die individuelle Softwaregestaltung überwinden, mehrere Programmierenden-Teams einbinden und viele Anwendende mit unterschiedlichen Spezialisierungen berücksichtigen. Die zehn Programmierenden gehören zur IT-Abteilung. Die Anwendenden der Software umfassen ca. 600 Monteur:innen und 100 Dispatcher:innen und sind selbst noch einmal fachlich spezialisiert für Hoch-/Nieder-/Mittelspannung und Strom-, Gas- und Wassernetze.

Aufgrund der Grundkoordination muss der Arbeitsprozess der internen Softwaregestaltung im EVU quer zu den existierenden hierarchischen Strukturen und Abteilungsgrenzen stattfinden. Dies zeigt sich neben den Hierarchieebenen von IT- und Instandhaltungsabteilung an den hierarchisch vorgegebenen Zielen für die Softwaregestaltung: Kostenreduzierung (weniger Personal bzw. mehr Arbeit mit gleichem Personal) und Qualitätssicherung in der Instandhaltung.

Dabei bringt die Softwarearchitektur eine Spezialisierung und damit zusätzliche Wissensgrenzen zwischen den Programmierenden mit sich, die der Arbeitsprozess der Softwaregestaltung berücksichtigen muss. Denn die Software besteht aus vier Teilen, die allesamt für den Arbeitsprozess der Instandhaltung notwendig sind: die Software auf den mobilen Endgeräten, die dazugehörige Software für die Anbindung an das ERP-

System (Middleware[4]), die ERP-Software selbst und eine Software für die Auftragsverarbeitung (Dispatching). Diese dezentrale Softwaregestaltung im EVU, unabhängig vom anbietenden Unternehmen der ERP-Software SAP, ermöglicht die SAP-Architektur, die vielfältige Erweiterungsmöglichkeiten z.B. durch diverse Schnittstellen bietet.

Vom Anwendungsbereich hängt ab, für was das EVU die von ihm gestaltete Software einsetzen kann. In dem Fall ist nicht der gesamte Arbeitsablauf der Netz-Instandhaltung in der Software abgebildet, sondern nur die Steuerung der Monteur:innen. Ihre Steuerung soll möglichst automatisiert ablaufen. Früher haben Meister:innen den Monteur:innen Zettel mitgegeben, auf denen stand, was zu tun ist. Jetzt nimmt ein Team von Auftragsverarbeitenden zentral Aufträge auf (z.B. von Baufirmen) und gibt sie digital an Monteur:innen weiter, die von ihrem mobilen Gerät über Aufträge informiert werden. Sie geben in das mobile Gerät Daten wie Arbeitsstunden oder Zählerstände ein. Wie weit die Automatisierung der Auftragsverteilung geht, ist noch offen. Aktuell gibt es intern im Büro noch sogenannte Dispatchende. Sie stehen für Rückfragen der Monteur:innen zu Aufträgen zur Verfügung und verteilen Aufträge, welche die Software nicht automatisch zuordnen kann.

8.2.1.2. INTERN2: Anpassung ERP-Software durch mehrere Fachbereiche und die IT-Abteilung zur Auftragsdatenverarbeitung, hierarchisch

In diesem Fall führt die soziotechnische Konstellation dazu, dass der Arbeitsprozess der individuellen Softwaregestaltung im Gegensatz zu INTERN1 Wissensgrenzen nicht nur zwischen IT und einem Fachbereich, sondern zwischen mehreren Fachbereichen und IT-Teams überwinden muss. Die mehreren Hundert Anwendenden für das SAP-Modul Work-Management, welches zur Auftragsverarbeitung im Netzbereich dient, sind auf einige Fachbereiche verteilt. Im Unterschied zu INTERN1 sind mehrere IT-Teams auch außerhalb der IT-Abteilung involviert: Zum einen ist eine Abteilung (»Zentralabteilung«) für das gesamte Modul des Work-Managements zuständig. Sie kümmert sich um Weiterentwicklung, Fehlerbehebung und Datenbereinigung. Zum anderen gibt es dezentrale IT-Teams in den Fachbereichen wie Netzanschluss, die automatisieren und digitalisieren sollen. Dann gibt es noch die zwei internen und sieben externen Programmierenden, die im zuständigen IT-Team der zentralen IT-Abteilung sitzen.

Der interne Arbeitsprozess der Softwaregestaltung im EVU muss althergebrachte interne Hierarchien, deren jeweilige Ziele und Vorgaben und, im Unterschied zu INTERN1, mehrere Abteilungsgrenzen berücksichtigen. Dies zeigt sich neben den Hierarchieebenen auch in den hierarchisch vorgegebenen IT-Budgetzielen für die Softwaregestaltung: Das Management gibt jedem Fachbereich einzeln ein IT-Budget. Im Fachbereich Netzanschluss gibt das Management Ziele vor: Orientierung an der Kundschaft, Effizienz und Ausrichtung am Standard des SAP-Softwarepakets.

Dabei ermöglicht die Softwarearchitektur wie bei INTERN1 eine dezentrale Softwaregestaltung unabhängig vom anbietenden Unternehmen der ERP-Software SAP. Wobei es in dem Fall vor allem um die Anpassung des SAP-Standards durch Customi-

4 Eine Middleware ist eine Software, die den Datenaustausch zwischen zwei ansonsten nicht verbundenen Softwarekomponenten ermöglicht.

zing[5] und Programmierung geht sowie Schnittstellen zu anderen Softwarelösungen herzustellen und weniger darum, eine eigenständige Erweiterung zu gestalten wie bei INTERN1.[6]

Neben Arbeitsteilung, Grundkoordination und Softwarearchitektur stellt der Anwendungsbereich eine Ausgangsbedingung für die Softwaregestaltung dar. Anders als die Instandhaltung bei INTERN1 lässt sich die gesamte Auftragsverarbeitung in Software abbilden. Die EVU können mit dem Modul Work-Management der ERP-Software von SAP anstehende Arbeiten wie den Einbau von Stromzählern oder das Verlegen eines Netzanschlusses koordinieren und dann entsprechende Rechnungen stellen. Es geht um den Datenverarbeitungsprozess und die Prozessintegration mehrerer, auf Teams verteilter Arbeitsschritte: vom ersten Kontakt mit der Kundschaft über das Verlegen des Anschlusses bis zur Rechnungsstellung. In diesem Prozess erfasst der Fachbereich Netzanschluss dann z.B. die Dokumentation zu den Anschlüssen, Strom-/Gas-/Wasser-Verteilern und Häusern im ERP-System, und der Fachbereich Abrechnung schickt die Rechnung an die Kund:innen.

8.2.1.3. KOOP1: Erweiterung und Anpassung ERP-Software durch IT-DL und EVU zur Datenverarbeitung, marktbasiert

Die soziotechnische Konstellation hat in dem Fall zur Folge, dass mehrere EVU darüber kooperativ verhandeln, was sie als gemeinsamen Standard und was sie individuell gestalten. Dabei muss aufgrund der Arbeitsteilung der Arbeitsprozess für die Verhandlung zwischen Individual- und Standardsoftwaregestaltung mehrere Wissensgrenzen überwinden: nicht nur zwischen dem IT-DL und den EVU, sondern auch innerhalb einiger EVU und innerhalb des IT-DL selbst. Denn es besteht in diesem Fall zwar eine starke Zentralisierung der Softwareprogrammierung und -gestaltung aufgrund der Auslagerung vieler IT-Tätigkeiten an das IT-DL. Einige wenige EVU haben jedoch eigene Programmierende, Dienstleistungsunternehmen oder Softwarefirmen, mit denen sie zusammenarbeiten. Bezogen auf die umfangreichen Einstellungsmöglichkeiten an der SAP-Standardsoftware haben neben den Programmierenden auch andere Mitarbeitende die Möglichkeit, über vorgegebene Einstellungsmöglichkeiten Änderungen an der Software vorzunehmen.

Anders als in vorhergehenden Fällen muss sich der Arbeitsprozess der Softwaregestaltung mit einer Marktbeziehung als Grundkoordination zwischen den EVU und dem IT-DL arrangieren. Die Marktbeziehung zeigt sich an Verträgen, Kostenkalkulation und Monitoring durch monatliche Berichte. Zwei der befragten EVU sagen explizit, dass das IT-DL mit anderen IT-DL konkurriert und sie Aufträge auch an andere vergeben. Sie ha-

5 Unternehmensspezifische Einstellung einer Standardsoftware.

6 Zugleich bleibt die Softwaregestaltung auf einen Teil der von Softwareanwendenden verwendeten Softwarelösungen beschränkt. Der befragte Anwendende aus dem Fachbereich Netzanschluss setzt neben dem Modul Work-Management von SAP noch mehrere weitere Systeme ein, weil die Standardsoftware nicht alle notwendigen Funktionalitäten abbildet. Dadurch beschränkt sich die individuelle Softwaregestaltung auf einen Teil der verwendeten Software. Für die anderen Softwarepakete muss der Fachbereich mit anderen Teams der IT-Abteilung zusammenarbeiten.

ben intern wieder Know-how aufgebaut, um Aufwandsschätzungen für Anforderungen des IT-DL hinterfragen zu können. Die Marktbeziehung geht mit Misstrauen einher.

> »Also, Beraterfirmen sind auch Vertriebler. Vertriebler verkaufen ihre Seele. Das ist einfach so. […] Das ist immer alles toll. Die können auch immer alles, was fehlt, noch weiterentwickeln, klar.« (Teamleiter Fachbereich EVU2)

Doch ist das Besondere, dass die beteiligten Organisationen die Marktbeziehungen durch kooperative Beziehungen ergänzen. Es gibt Treffen unterschiedlicher Hierarchieebenen von EVU und IT-DL für operative und strategische Abstimmungen. Es gibt langjährige, persönliche Beziehungen zwischen Beschäftigten der Organisationen. Vor allem die größeren EVU und das IT-DL arbeiten intern über Hierarchien und Abteilungen hinweg zusammen, z.B. in Projekten, wie es für Matrixorganisationen typisch ist. Ob zwischen oder innerhalb der Organisationen: In diesem Fall besteht die Herausforderung darin, die Zusammenarbeit sowohl für einzelne Teile der Software, den gemeinsamen Industriestandard, individuelle Anforderungen als auch für langfristige strategische Projekte zu organisieren. Somit muss die zentralisierte Softwaregestaltung des kooperativen Standards durch das IT-DL trotz Marktbeziehungen und quer zu den hierarchischen Strukturen und Abteilungsgrenzen innerhalb und zwischen den Organisationen stattfinden.

Die Softwarearchitektur des kooperativen Standards sieht drei Stufen der Standardisierung vor: Der Kern besteht aus einem harten Standard, den jedes der beteiligten EVU hat. Dann kommt der Template-Bereich, dessen modulare Funktionalitäten einzelne der EVU nutzen können, aber nicht müssen. Auf der äußersten Schale finden sich freie, individuelle Anpassungen, die nur einzelne EVU auf ihren Systemen haben. Laut Befragten sind derzeit ca. 80 % der Anpassungen und Erweiterungen der ERP-Standardsoftware harmonisiert, d.h. für alle EVU, die an der Kooperation teilnehmen, gleich. Die Softwarearchitektur ermöglicht eine Softwaregestaltung sowohl dezentral in den EVU als auch zentral im IT-DL und zudem noch – wie in den Fallstudien INTERN1 und INTERN2 auch – unabhängig vom anbietenden Unternehmen der ERP-Software SAP.

Neben Arbeitsteilung, Grundkoordination und Softwarearchitektur bestimmt der Anwendungsbereich darüber, wie die Organisationen die Softwaregestaltung einsetzen. In dem Fall ist der Anwendungsbereich der gestalteten Software die industriespezifische Datenverarbeitung: Sie reicht von der Abrechnung der Kundschaft über die regulierungsbedingte Marktkommunikation bis hin zur Energiemengenbilanzierung. Entsprechend gibt es umfangreiche Möglichkeiten der Automatisierung. Eine befragte Sachbearbeiterin (EVU2) ist vor allem mit Fehlerklärungsarbeiten, Vervollständigungsarbeiten (z.B. fehlende Daten ergänzen, Daten aus E-Mails oder Telefonaten mit der Kundschaft eingeben, einzelne Schritte im Abrechnungsprozess manuell durchführen) oder Tests von Softwareänderungen beschäftigt. Aus ihrer Sicht sind 90–95 % der Abrechnung automatisiert und in der Marktkommunikation seien die Zahlen noch höher.

8.2.1.4. KOOP2: Erweiterung und Anpassung ERP-Software durch IT-DL und EVU zur Datenverarbeitung, marktbasiert

Wie bei KOOP1 schafft die soziotechnische Konstellation die Möglichkeit, dass sich EVU mithilfe eines gemeinsamen IT-DL auf einen gemeinsamen Standard einigen – nur, dass es in diesem Fall nicht gelingt. Anders als bei KOOP1 ist die Arbeitsteilung in dem Fall weder für die Anwendung noch die Programmierung oder Gestaltung weitestgehend in den EVU oder beim IT-DL zentralisiert. Nachdem ursprünglich das IT-DL die Softwaregestaltung verantwortet hat, haben einzelne EVU wieder verstärkt die Koordination der Softwaregestaltung übernommen. Programmierung für die Erweiterung der ERP-Software findet sowohl beim IT-DL als auch teilweise innerhalb der EVU oder in den von EVU geleiteten Projekten statt (dann u.U. mit externen Programmierenden wie Freelancern). Das IT-DL übernimmt u.a. Entwicklungsaufgaben für den (halb)jährlichen Formatwechsel, der von der Regulierung vorgegeben ist, oder entwickelt kleinere Software-Lösungen. Es nimmt zudem Customizing an der ERP-Software vor. Die uneinheitliche Arbeitsteilung zeigt sich auch an der Anwendung der Software. Diese findet zum einen in den EVU und, weil einige sie ausgelagert haben, beim IT-DL statt. Ein EVU aus dem Organisationsnetzwerk hat laut einem Befragten gar keine operativen Anwendenden mehr für industriespezifische Arbeiten mit dem ERP-System. Wie bei KOOP1 muss der Arbeitsprozess für die individuelle oder Standardsoftwaregestaltung aufgrund der Arbeitsteilung Wissensgrenzen zwischen Fach- und IT-Wissen nicht nur zwischen dem IT-DL und den EVU, sondern auch innerhalb einiger EVU und innerhalb des IT-DL selbst überwinden.

Wie auch bei KOOP1 muss sich der Arbeitsprozess der Softwaregestaltung aufgrund der Grundkoordination mit der Marktbeziehung zwischen den EVU und dem IT-DL arrangieren. Wie bei KOOP1 reichen Verträge für die Kooperation nicht aus. Beziehungspflege gehört zur Koordination zwischen den Firmen dazu. Wobei der Fall zeigt, wie brüchig eine Kooperation sein kann. Die Befragten nennen einzelne Ereignisse, die Auslöser oder Manifestationen von fehlendem Vertrauen zwischen beiden waren, was dann zur Rückverlagerung von IT-Arbeit in die EVU geführt hat.

> »Warum ist man seit Jahren unzufrieden und warum ist das eskaliert? Manchmal ist es relativ einfach. Die Erwartungshaltung ist zu groß – von Anfang an. Man hat sich nie verstanden oder nie so richtig ausgetauscht. […] [W]as möchte der Kunde und was kann das Unternehmen überhaupt leisten? […] Und wir sind nie das Kernproblem angegangen. Nie. ›Komm, wir reden mal offen darüber.‹ Das ist dann sukzessive eskaliert und irgendwann hatten die Mitarbeiter alle keine Lust mehr.« (Teamleiter IT-DL)

Ein anderer Grund ist, dass ein EVU Entscheidungen über die Softwareanwendung und deren Gestaltung autonom treffen will:

> »Der Grund, warum wir zurückgeholt wurden, war, dass das Know-how wieder im Haus ist, weil die Markt-Kommunikation sehr komplex geworden ist und man nicht mehr vom Dienstleister abhängig sein wollte. […] Ja, die sind diejenigen, die die Prozesse ausführen. Aber sie sind nicht Prozesseigentümer, sollte auch kein Dienstleister sein. Aber wenn ich als Auftraggeber Entscheidungen treffe, muss ich das fachliche

> Know-how haben hinten dran … und deswegen wurde dann damals entschieden, uns wieder zurückzuholen.« (Teamleiterin Marktkommunikation EVU2)

Im Unterschied zu KOOP2 führen die EVU teilweise selbst kooperative Projekte mit anderen EVU durch und das IT-DL stellt Mitarbeitende nur zur Umsetzung einzelner Anforderungen zur Verfügung. Zusätzlich existiert innerhalb der EVU eine Matrixorganisation, d.h., die Mitarbeitenden arbeiten über Hierarchien und Abteilungen z.B. in IT-Projekten zusammen. Die Softwaregestaltung im IT-DL und jene dezentral in den EVU müssen damit basierend auf diesen Marktbeziehungen und quer zu den hierarchischen Strukturen und Abteilungsgrenzen innerhalb der EVU stattfinden.

Anders als in KOOP1 ließen die Interviews keine klare Systematik erkennen, welche EVU inwieweit einen gemeinsamen Standard einsetzen und was sie individuell noch angepasst haben. Die Organisationen erweitern das gleiche SAP-Standard-ERP-System wie in den vorhergehenden Fällen (INTERN1, INTERN2, KOOP2) und somit prägt die gleiche Softwarearchitektur die Arbeitsteilung zwischen Anwendung und Entwicklung: Das ERP-System ermöglicht es, über Schnittstellen Drittsysteme anzubinden, und erlaubt weitreichende Erweiterungs- und Anpassungsmöglichkeiten am Standard. Die Softwarearchitektur ermöglicht sowohl eine dezentrale Softwaregestaltung in den EVU als auch zentral im IT-DL unabhängig von SAP, das die ERP-Software anbietet.

Neben Arbeitsteilung, Grundkoordination und Softwarearchitektur ist der Anwendungsbereich eine Ausgangsbedingung. Er entscheidet darüber, für was die Organisationen Software gestalten und wie weitgehend sie Arbeit durch Software ersetzen können. In dem Fall ist der Anwendungsbereich der gestalteten Software wie bei KOOP1 vorwiegend die industriespezifische Datenverarbeitung: u.a. Abrechnung, Marktkommunikation und Energiemengenbilanzierung.

8.2.1.5. PAKET: Entwicklung industriespezifischer Standard-ERP-Software durch Softwarefirma zur Datenverarbeitung, marktbasiert

Als Teil der soziotechnischen Konstellation führt in diesem Fall die Arbeitsteilung dazu, dass der Arbeitsprozess der Softwaregestaltung einer Standardsoftware zentral in der Softwarefirma stattfindet und deshalb die Wissensgrenzen geringer sind. Denn die Softwarefirma macht federführend die Konzeption, die Tests und den Support für die Software in interdisziplinären Teams, die auf einzelne Funktionalitäten oder Module spezialisiert sind. Sie besitzt für die industriespezifische Softwareentwicklung tiefergehendes Wissen zur Energiewirtschaft und speziell zur Regulierung und den grundlegenden Geschäftsprozessen, d.h. all das, was alle EVU leisten müssen. Trotzdem ist die Zusammenarbeit mit ausgewählten EVU zur Gestaltung des Standards notwendig, wobei die Softwarefirma selbst über Branchenwissen verfügt, weswegen nicht so viel Wissen über die Marktbeziehung hinweg ausgetauscht werden muss. Dabei geht es vor allem um Feedback der EVU zu fertigen Konzepten und Umsetzungen. Die operativ Anwendenden sind auf einen fachlichen Bereich spezialisiert und damit auch auf den entsprechenden Teil der Software (Kund:innenservice, Energiedatenmanagement, Abrechnung etc.). Sie sitzen in den EVU, in der Softwarefirma selbst (die als Dienstleistung für EVU Business Process Outsourcing (BPO) anbietet) und bei anderen IT-DL, die für die EVU Geschäftsprozesse abwickeln.

Aufgrund der Grundkoordination innerhalb der Softwarefirma muss sich der Arbeitsprozess der Softwaregestaltung zwar mit den internen Hierarchien arrangieren. Jedoch gibt es spezialisierte, interdisziplinäre Teams für einzelne, fachlich abgegrenzte Teile der Software, in denen ein hierarchieunabhängiger Austausch möglich ist. Weil es sich zwischen Softwarefirma und EVU um eine Lieferbeziehung mit einer klar definierten Leistungserbringung handelt, sind Verträge zentral für die Beziehung. Sie legen fest, welche Funktionen die Software erfüllt. Die Softwarefirma garantiert z.B. die Umsetzung der Regulierung zum gesetzlichen Stichtag, bildet die energiewirtschaftlichen Kernprozesse ab und garantiert Zuverlässigkeit in puncto Qualität und fristgerechte Fertigstellung von Softwareupdates. Die EVU sind davon abhängig, weil sie viele ihrer Aufgaben nur mit der Software erledigen können. Für einzelne, ausgewählte EVU besteht jedoch keine reine Marktbeziehung. Diese bindet die Softwarefirma bei strategischen Entscheidungen über ein Gremium ein. Sie machen bei Arbeitsgruppen und Entwicklungsprojekten mit. Hier muss sich der Arbeitsprozess der Softwaregestaltung mit der Marktbeziehung zwischen EVU und Softwarefirma arrangieren, wenn auch nur für bestimmte Projekte, umfangreichere Konzepte oder kleinere Anforderungen.

Anders als in den anderen Fallstudien, die SAP verwendet haben, sieht die Softwarearchitektur von PAKET keine Programmierungen durch die EVU selbst vor. Der Standard an branchenspezifischen Funktionalitäten ist umfassend und soll ausreichen, um die notwendigen Geschäftsprozesse abzuwickeln. Die Anpassungsmöglichkeiten durch die EVU beschränken sich auf vordefinierte Einstellungsmöglichkeiten. Diese müssen jedoch vorgenommen werden, weshalb trotz des Standards nicht von einer Plug-and-Play-Anwendung gesprochen werden kann. Darüber hinaus stellt PAKET Schnittstellen zur Verfügung, dank derer die EVU Softwarelösungen anderer Softwarehersteller anschließen können.

Neben Arbeitsteilung, Grundkoordination und Softwarearchitektur beeinflusst der Anwendungsbereich die Softwaregestaltung, indem u.a. von ihm abhängt, welche Arbeiten Software erledigen oder steuern soll. In dem Fall ist der Anwendungsbereich zugleich der Kern der energiewirtschaftlichen ERP-Software: die industriespezifische Datenverarbeitung – wie bei KOOP1 und KOOP2. Sie ist das Zentrum der Leistungserbringung in Bereichen wie Abrechnung, Marktkommunikation oder Bilanzierung von Energiemengen. Wie auch in den vorhergehenden Fallstudien ist aufgrund des hohen Anteils an Datenverarbeitung eine hohe Automatisierung möglich. Bei hoher Automatisierung kümmern sich die Anwendenden der Software vor allem um Restfälle, d.h. bspw. einzelne Kund:innen oder Datensätze, welche die Software nicht automatisiert verarbeiten kann. In einem befragten EVU ist die Automatisierung der Datenverarbeitung seit längerer Zeit ein Thema:

> Der »automatisierte[…] Rechnungseingang: Seit 15 Jahren arbeiten wir da dran in verschiedenen Stufen.« (IT-Leiter EVU2)

8.2.1.6. KOOP3: Ko-Produktion einer IoT-Anwendung zwischen Softwarefirma und IT-DL zur Überwachung, netzwerkförmig

Wie bei PAKET hat die Arbeitsteilung zwischen Softwarefirma, IT-DL und EVU zur Folge, dass die Softwarefirma die Software zentralisiert programmiert und das IT-DL nur

mitgestaltet, indem es Anforderungen einbringt. Doch anders als bei PAKET spielt die Arbeitsteilung zwischen den Organisationen als Wissensgrenze eine geringe Rolle. Erstens ist das Wissen gleichmäßig verteilt: Mit dem IT-DL gibt es eine Organisation, die die IoT-Software des Softwareunternehmens bei den EVU implementiert, das Softwareunternehmen bei der Weiterentwicklung der Software unterstützt, selbst Erweiterungsmodule entwickelt und auch Schulungen für EVU anbietet, damit diese eigenständig Module entwickeln können. Zweitens ist das Wissen über IoT-Anwendungsfälle in der Energiewirtschaft weniger kompliziert und sowohl die Softwarefirma als auch das IT-DL verfügen über Wissen über IoT und die IoT-Standardsoftware. Weil die Wissensgrenzen geringer sind, sich die Möglichkeiten von individueller und Standardsoftwaregestaltung auf verschiedene Organisationen verteilen und sie dabei auch noch kooperieren, zählt der Fall zu den kooperativen Formen der Softwaregestaltung.

Zwar bestehen Marktbeziehungen zwischen Softwarefirma und IT-DL, doch spielen diese bei der Zusammenarbeit eine geringe Rolle. Vielmehr ist die Grundkoordination in diesem Fall das Netzwerk. Warum? Weil Verträge und hierarchische Vorgaben oder Strukturen in diesem Fall keine Hindernisse für die Softwaregestaltung darstellen. Verträge wurden von den Befragten nicht als zentral genannt, genauso wenig wie das IT-Budget. Die Kosten für Sensoren und IoT-Software sind gering. Zudem hat das IT-DL keine große Erwartung, was die Einnahmen durch IoT-Projekte und die EVU was z.B. die Kostenersparnis durch IoT-Projekte anbelangt. Es geht primär darum, den Einsatz von IoT voranzutreiben und damit um eine kooperative Softwaregestaltung bzw. Ko-Produktion einer IoT-Software.

Wissensgrenzen spielen auch wegen der Softwarearchitektur eine untergeordnete Rolle. Zum liegt das an der Modularität der Software. Die zentrale Datenplattform, von der Softwarefirma entwickelt, hat Schnittstellen, welche das IT-DL oder die EVU nutzen, um selbst Erweiterungsmodule zu programmieren. Die Softwarefirma hat ein Interesse daran, dass sich die Arbeit nicht in der Softwarefirma zentralisiert. Das zeigt sich daran, dass die Datenplattform auch ohne größere IT-Kenntnisse bedienbar sein soll u.a. weil z.B. Kommunen wenig Geld und Personal haben. Dafür existiert eine nach dem Prinzip »low code«[7] gestaltete Software, die es erlaubt, ohne große Programmierkenntnisse Sensoren zu verbauen und in den Datenfluss zu integrieren.

Der Anwendungsbereich grenzt die Möglichkeiten der Softwaregestaltung auf eine Software ein, die Daten erfasst. In diesem Fall setzen die EVU die IoT-Software zur Überwachung ein: ob von Straßen, Stromnetzen oder Abwasserkanälen. Denn unter IoT versteht der untersuchte Fall, dass Sensoren ausgebracht werden und deren Daten in einer IoT-Datenplattform zusammenfließen. Letztere soll Basis für Smart-City-Lösungen sein. Da es sich um eine Infrastruktur für Daten handelt, geht es bei der Implementierung weniger um Anwendende als um Anwendungsfälle. Die Daten fließen meist in andere Anwendungen ein. Bei einem befragten EVU fließen Daten von Sensoren über die Datenplattform in ein bestehendes System des Netzbetriebs zur Netzkontrolle ein.

7 Ansatz in der Softwareentwicklung, der versucht, auf Programmierung zu verzichten, und stattdessen auf grafische Designwerkzeuge oder stark vereinfachte Programmiersprachen setzt.

8.2.1.7. STARTUP: Primat der Softwareentwicklung in einer Organisation zur Datenverarbeitung, netzwerkförmig (rollenbasiert)

Die Arbeitsteilung ist in dem Fall so organisiert, dass die Wissensgrenzen gering sind und bleiben. Dank der soziotechnischen Konstellation muss der Arbeitsprozess für die individuelle Softwaregestaltung keine Organisations-, Team- oder Abteilungsgrenzen überwinden. Die Arbeitsteilung zwischen Anwendung und Entwicklung ist vielmehr so organisiert, dass die Programmierenden und softwaregestaltenden energiewirtschaftlichen Fachleute des Start-ups fest und regelmäßig in Kreisen zusammenarbeiten. Es gibt je einen Kreis für die zwei Teile der Software: für den Zertifikate-Handel und für die Anmeldung von E-Autos. Die Anwendenden sind nicht Teil dieser Kreis-Organisation. Sie sind aber auch nicht jene mit dem tiefergehenden Fachwissen über den Zertifikate-Handel, das die Softwaregestaltung benötigt.

Anders als in den anderen Fällen gestaltet STARTUP die Software primär nur für sich selbst. Es ist von Anfang an auf die Softwaregestaltung ausgerichtet. Es gilt der Primat der Softwareentwicklung (siehe 4.1). Die Grundkoordination ist in diesem Fall das Netzwerk – weder Hierarchie noch Markt stören bei der Softwaregestaltung. Vielmehr ist die Organisation rollenbasiert. Die Verantwortung für Aufgaben je Mitarbeitenden sind als Rollen definiert (»Programmierende«, »Product Owner:in«, »Social Media Management« etc.). Über diese Rolle sind die Mitarbeitenden Kreisen zugeordnet, wobei eine Person mehrere Rollen haben und eine Rolle mehreren Kreisen angehören kann – temporär oder langfristig. Der Netzwerkcharakter zeigt sich auch daran, dass es für den Einzelnen eine Flexibilität gibt, was die Arbeitsaufgaben anbelangt: Es gibt regelmäßige Treffen, wo alle sagen können, ob sie eine weitere Rolle übernehmen oder abgeben wollen.

> »Also, wir haben da halt auch keine Hierarchie dahinter in dem Sinne. Sondern, das ist eben ein rein rollenbasiertes System. […] [I]m Endeffekt basiert alles bei uns auf Kompetenz und Verantwortung und nicht in dem Sinne auf: ›Ja, die Person hat halt das und das Sagen‹, sondern immer nur inhaltlich auf die entsprechenden Kompetenzbereiche bezogen.« (Programmierer1)

Zuletzt zeigt sich der Netzwerkcharakter daran, dass Mitarbeitende spontan ohne Absprache mit einem Vorgesetzten via direkter Kommunikation Themen bearbeiten, z.B. durch informelle Treffen oder Online-Chats.

Dabei ermöglicht die Softwarearchitektur eine getrennte Softwaregestaltung der beiden Teile (Anmeldung der E-Autos und Quotenhandel) unabhängig voneinander in den Kreisen. Es sind nur wenige Absprachen notwendig. Die beiden Teile verfügen jeweils über Schnittstellen, um andere Softwarelösungen anzuschließen, welche die Firma nicht selbst entwickelt hat (z.B. für die Rechnungslegung).

Neben Arbeitsteilung, Grundkoordination und Softwarearchitektur ist der Anwendungsbereich eine Ausgangsbedingung für die Softwaregestaltung. Er legt fest, für welchen Zweck die gestaltete Software eingesetzt wird. Im Anwendungsbereich der Software geht es um die Anmeldung von E-Autos für den Zertifikate-Handel. Da es sich um eine reine Datenverarbeitung handelt, sind die Möglichkeiten der Softwaregestaltung weitgehend und eine Automatisierung naheliegend. Die Grenzen liegen in dem Fall mehr

in den fehlenden digitalen Schnittstellen zu Behörden und notwendigen Prüfschritten, weil z.B. beim Lesen der Fahrzeugscheine noch Fehler auftreten können oder das START-UP ausgezahlte Beträge noch einmal manuell prüfen will. Für die Prüfung der Fahrzeugscheine gibt es zwei bis drei 450-Euro-Kräfte als Anwendende. Weil STARTUP von Anfang an und auch weiterhin auf die Möglichkeiten der Softwareentwicklung setzt, machen Anwendende nur das, was (noch) nicht die Software erledigt.

8.2.2. Zusammenfassung

8.2.2.1. Überblick über zentrale Unterschiede und ihre Folgen

Die Fallstudien haben gezeigt, dass die Softwaregestaltung unter ganz unterschiedlichen soziotechnischen Konstellationen stattfindet. Zusammenfassend gibt dieser Teil einen Überblick über die Unterschiede zwischen den Fallstudien, ihre Folgen für den Arbeitsprozess der Softwaregestaltung, die Arbeit der Softwaregestaltenden und die soziotechnische Arbeitsgestaltung in den EVU.

In der untenstehenden Tabelle sind die Ausprägungen der vier Kategorien der soziotechnischen Konstellation für jede Fallstudie zusammengefasst. Wie die Tabelle zeigt, besteht die Arbeitsteilung bei INTERN1 und INTERN2 zwischen verschiedenen Abteilungen (IT- und Fachbereiche innerhalb der EVU), bei KOOP1 und KOOP2 zwischen mehreren EVU und einem IT-DL, bei PAKET zwischen mehreren EVU und einer Softwarefirma, bei KOOP3 zwischen mehreren EVU, dem IT-DL und der Softwarefirma und bei STARTUP innerhalb einer Organisation ohne Abteilungsgrenzen. Der Anwendungsbereich unterscheidet sich zwischen Datenverarbeitung (DV), Steuerung von Beschäftigten und Überwachung. Die Grundkoordination unterscheidet zwischen Markt, Hierarchie und Netzwerk. Bei KOOP1, KOOP2 und PAKET besteht zwischen EVU und IT-DL bzw. Softwarefirma eine Marktbeziehung, innerhalb der Organisationen jedoch eine Hierarchie, die bei der Softwaregestaltung eine Rolle spielt. Die Softwarearchitektur ist bei den ersten vier Fallstudien durch das SAP-ERP-System geprägt. In diesen Fallstudien wird die SAP-Software erweitert und angepasst. Die Ausprägung je Fall ist in der folgenden Tabelle aufgelistet:

Tabelle 16: Überblick soziotechnische Konstellation je Fall

Fall	Schwerpunkt Anwendungsbereich	Arbeitsteilung	Grundkoordination	Softwarearchitektur
INTERN1	Auftragssteuerung Instandhaltung (Steuerung von Beschäftigten)	integriert: ein Fachbereich und IT-Abteilung	Hierarchie	individuell: Erweiterung und Anpassung ERP, mobile App
INTERN2	Arbeitsauftragsverarbeitung Netze (DV)	integriert: etliche Fachbereiche und IT-Abteilung	Hierarchie	individuell: Anpassung ERP

Fall	Schwerpunkt Anwendungsbereich	Arbeitsteilung	Grundkoordination	Softwarearchitektur
KOOP1	Energiewirtschaftliche Geschäftsprozesse (DV)	desintegriert: EVU und IT-DL	Markt/ Hierarchie	kooperativer Standard: Erweiterung und An-passung ERP
KOOP2	Energiewirtschaftliche Geschäftsprozesse (DV)	desintegriert: EVU und IT-DL	Markt/ Hierarchie	prekärer Standard: Erweiterung und Anpassung ERP
PAKET	Energiewirtschaftliche Geschäftsprozesse (DV)	desintegriert: EVU und Softwarefirma	Markt/ Hierarchie	Standard-ERP
KOOP3	Überwachung mittels Sensoren	desintegriert: EVU, IT-DL, Softwarefirma	Netzwerk	Standard-IoT-Software, individuelle Erweiterung
STARTUP	Anmeldung und Handel CO_2-Zertifikate E-Autos (DV)	integriert: etliche Rollen und Kreise	Netzwerk	individuell: eigenständige Module Anmeldung und Handel

Die Unterschiede der soziotechnischen Konstellation haben Folgen für den 2. Teil des Analyserahmens: den Arbeitsprozess der Softwaregestaltung. Soziotechnische Konstellation und Arbeitsprozess der Softwaregestaltung haben wiederum Folgen für die Teile der Arbeitsgestaltung und die Arbeit der Softwaregestaltenden. Die wesentlichen typischen Zusammenhänge seien kurz aufgeführt und anhand von diametralen Typen zur Unterscheidung der Fallstudien verdeutlicht.

1. Die Kategorien der soziotechnischen Konstellation von Arbeitsteilung, Grundkonstellation und Architektur wirken sich auf den **Arbeitsprozess der Softwaregestaltung** aus. Sie bestimmen, ob er **dezentral in EVU oder zentralisiert** Software gestaltet.
2. Für die **Beschäftigtengruppe der Softwaregestaltenden** hängt von der soziotechnischen Konstellation ab, inwiefern sie in einer **reinen Netzwerk- oder in einer Matrixorganisation** arbeiten.
3. Auf die **soziotechnische Arbeitsgestaltung** wirkt sich die soziotechnische Konstellation dahingehend aus, ob in den Fällen ein EVU **unabhängig oder nur abhängig** von anderen Organisationen Software und damit die eigene Arbeit gestalten kann.

Die weiter unten folgenden Abschnitte zu den jeweiligen Teilen des Analyserahmens und die Zusammenfassung des Kapitels gehen ausführlicher auf die diametralen Unterscheidungstypen wie zentral – dezentral, Matrix – reines Netzwerk und unabhängig – abhängig und die Zusammenhänge ein.

8.2.2.2. Ergebnisse des Fallvergleichs je Kategorie

Für jede Kategorie der soziotechnischen Konstellation zeigen die Ergebnisse des Fallvergleichs, welche allgemeinen Aussagen sich treffen lassen.

Arbeitsteilung – wie die Wissensgrenzen verlaufen

Aus Sicht der Softwaregestaltung wird die Arbeitsteilung zwischen Anwendung und Programmierung zur Wissensgrenze, die sie überwinden muss. Sie kann innerhalb oder zwischen Firmen bestehen. Es gibt Fälle, in denen Anwendung und Programmierung in einer Organisation, aber durch Bereichsgrenzen getrennt sind (INTERN1, INTERN2). Es gibt Fälle, bei denen einzelne Organisationen interdisziplinär arbeiten und die Wissensgrenzen nur noch zwischen einzelnen Beschäftigten bestehen – im STARTUP, innerhalb von interdisziplinären Teams in der Softwarefirma von PAKET oder den IT-DL von KOOP1/KOOP2. Was auffällt, ist, dass in den Fällen weniger die Anwendenden Programmierungswissen haben, als vielmehr die Programmierenden über Fachwissen verfügen.

Die untenstehende Tabelle zeigt je Fallstudie die Arbeitsteilung zwischen Anwendung und Entwicklung (Programmierung und Gestaltung).

Tabelle 17: Arbeitsteilung zwischen Anwendung und Entwicklung

Fall	Arbeitsteilung Anwendung – Gestaltung – Programmierung
INTERN1	eine Firma (Fachbereich und IT-Abteilung getrennt), einige Programmierende extern
INTERN2	eine Firma (Fachbereiche und IT-Abteilung getrennt), einige Programmierende extern
KOOP1	verteilt auf IT-DL (Programmierung, Gestaltung, BPO) und EVU (Anwendung, Gestaltung), wenige EVU mit eigenen Programmierenden
KOOP2	innerhalb IT-DL (Programmierung, Gestaltung, BPO) und EVU (Anwendung, Gestaltung), wenige EVU mit eigenen Programmierenden
PAKET	Softwarefirma programmiert, gestaltet und wendet an (BPO), EVU wenden an und gestalten (nehmen Einstellungen am Standard vor)
KOOP3	Softwarefirma programmiert Kern und IT-DL gestaltet mit, IT-DL und EVU programmieren und gestalten Erweiterungen; Anwendung durch EVU
STARTUP	eine Firma, interdisziplinäre Kreise

Grundkoordination – Ausgangspunkt von Kommunikation und Kooperation

Von der Grundkoordination ausgehend (egal welche vorliegt) müssen die Organisationen der Fallstudien den Arbeitsprozess der Softwaregestaltung etablieren. Als Formen der Grundkoordinationen zwischen Anwendung und Programmierung zeigen sich in den Fällen Markt, Hierarchie oder Netzwerk. Wobei Letzteres besser zur Softwaregestaltung passt, weil Wissensgrenzen geringer sind und es weniger Hürden bei der Zusammenarbeit gibt. Kontrollelemente wie IT-Budget, Service Level Agreements (SLA) zwischen EVU und IT-DL, Projektverträge innerhalb eines EVU oder andere Zielvorgaben können die Folgen von Hierarchien oder Märkten auf die Softwaregestaltung verstärken

oder, wie im Fall von KOOP3, so eine geringe Rolle spielen, dass trotz Marktbeziehung der Typ der Netzwerk-Grundkoordination vorliegt.

Neben diesen Kontrollelementen kann innerhalb der EVU die IT-Abteilung der EVU die Zusammenarbeit zwischen Anwendung und Programmierung prägen. Sie spielen in den Fallstudien zwar eine Rolle. Insgesamt sind in den Fällen die IT-Abteilungen aber keine Hürde bei der Softwaregestaltung. Sie sind für den Betrieb der IT-Systeme zuständig, stellen Programmierende zur Verfügung oder koordinieren die als Externe eingebundenen Programmierenden. Vor allem bei größeren EVU koordinieren sie die Softwaregestaltung.

Tabelle 18: Grundkoordination je Fall und die Rolle der IT-Abteilung

Fall	Grundkoordination	IT-Abteilung EVU
INTERN1	Hierarchie: interne Matrixorganisation	stellt Programmierende und IT-Infrastruktur zur Verfügung
INTERN2	Hierarchie: interne Matrixorganisation	stellt Programmierende und IT-Infrastruktur zur Verfügung, verteilte IT-Teams für Softwaregestaltung in den Fachbereichen
KOOP1	Markt: zwischen EVU und IT-DL	wenn vorhanden (größere EVU): organisieren Anforderungsaufnahme, manche machen eigene IT-Projekte
	Hierarchie: Matrixorganisation innerhalb EVU und IT-DL	
KOOP2	Markt: zwischen EVU und IT-DL	wenn vorhanden (größere EVU): in einem EVU Anforderungsaufnahme organisieren; in anderem EVU initiiert sie IT-Projekte in Fachbereichen
	Hierarchie: Matrixorganisation innerhalb EVU und IT-DL	
PAKET	Markt: Softwarefirma – EVU	FB der EVU arbeiten direkt mit SF zusammen; in einem EVU entscheidet FB unabhängig von IT-Abteilung über Softwareauswahl
	Hierarchie: innerhalb Softwarefirma und EVU	
KOOP3	Netzwerk: Ko-Produktion IoT	Ansprechpersonen bei Implementierung IoT für Integration IoT-Software in bestehende IT-Landschaft
STARTUP	Netzwerk: rollenbasierte Kreise	keine

Softwarearchitektur – Grundstruktur der Arbeitsteilung in Organisationen und Organisationsnetzwerken

Die Softwarearchitektur hat sich in allen Fällen als zentral dafür erwiesen, wie sich die Arbeit in den Gestaltungsnetzwerken verteilt. Erstens ist sie zentral wegen der Aufteilung der Software in z.B. Module, mit der eine Spezialisierung der Programmierenden einhergeht und die darüber entscheidet, was ein EVU außerhalb von Softwarefirmen wie SAP, von PAKET oder der IoT-Softwarefirma von KOOP3 gestalten kann. Dabei ist die Aufteilung der Module, Softwareteile oder Schnittstellen bereits im Vorhinein gegeben. Die gegebene Architektur führt die softwaregestaltenden Organisationen fort. Zweitens schafft die Architektur Abhängigkeiten, wenn sich mehrere Teams, Abteilungen oder Organisationen auf einen Standard einigen müssen. In einigen Fallstudien ist es Teil der

Softwaregestaltung zu verhandeln, welche Anforderung in einen Standard einfließt und welche individuell umgesetzt wird.

Die Architektur prägt erstens, wo und wer Software gestalten und programmieren kann und wie sich die Programmierenden spezialisieren. Alle Fälle zeigen, dass die jeweiligen Programmierenden(teams) auf bestimmte Softwareteile spezialisiert sind. Bei KOOP3 will die Softwarefirma die technischen Voraussetzungen schaffen, damit es möglichst einfach ist, durch Schnittstellen weitere Module anzuschließen. Solche individuellen Module kann dann z.B. das IT-DL unabhängig von der IoT-Softwarefirma programmieren. Bei INTERN1 sind die Programmierenden entweder auf die mobile Lösung oder das Instandhaltungsmodul der ERP-Software spezialisiert, bei INTERN2 auf das Modul für die Auftragsverwaltung. Im Fall STARTUP entwickeln unterschiedliche Programmierende jeweils die Module für Anmeldung oder Handel von CO_2-Zertifikaten.

Bei den Organisationen, die größere ERP-Pakete (mit)gestalten, wie die IT-DL von KOOP1 und KOOP2 und die Softwarefirma von PAKET, verteilen sich die Programmierenden auf fachlich aufgeteilte Softwareteile für energiewirtschaftliche Anwendungsbereiche wie Marktkommunikation, Energiemengenbilanzierung, Abrechnung oder Instandhaltung. In diesen Fällen existieren auf diese Bereiche spezialisierte, interdisziplinäre Teams. Dort programmiert und gestaltet ein Team aus Fach- und Softwarespezialist:innen ein Modul eigenständig.

Die ersten vier Fälle haben das gleiche ERP-System von SAP (Näheres zur Entwicklungsplattform unter 4.3). Für alle EVU, die dessen individuelle Erweiterungsmöglichkeiten nutzen, bedeutet das, dass zur betrieblichen Realität nicht einfach nur die Anwendung von Software gehört, sondern auch eine Test- und Entwicklungsumgebung existiert. So hat die Architektur, die individuelle Erweiterungen zulässt, Folgen für die Anwendenden: Sie können und/oder müssen die kontinuierlichen Softwareänderungen testen.

Die Softwarearchitektur prägt zweitens die Softwaregestaltung dadurch, dass sie vorgibt, was individuell gestaltbar ist und was nicht. Gestalten Organisationen gemeinsam einen Standard, müssen sie sich im Arbeitsprozess der Softwaregestaltung darüber verständigen, was Teil des Standards wird. Dann spiegeln sich Kommunikationsstrukturen und Softwarearchitektur nicht. Ein Team programmiert den gemeinsamen Standard, und die Softwaregestaltung sorgt dafür, dass die Beteiligten über Architekturgrenzen hinweg kommunizieren (das können auch mehrere Organisationen sein). Wenn eine Organisation wie eine Softwarefirma den Standard vorgibt oder ein EVU eine Software individuell gestaltet, ist eine solche Abstimmung zwischen Organisationen nicht Teil der Softwaregestaltung.

In der Mehrzahl der EVU ist die energiewirtschaftliche Arbeitsteilung nach wie vor mehr an fachlichen Themen als an der Software-Architektur orientiert. Alle FB arbeiten mit den für sie relevanten Teilen der ERP-Software, ohne sich mit anderen abstimmen zu müssen. Es gibt aber EVU in einigen Fallstudien, die Software über Architekturgrenzen hinweg gestalten. Ob bei INTERN2 oder KOOP1: Durch fachbereichs- oder EVU-übergreifende Anforderungsrunden entscheiden verteilte fachliche Expert:innen über einen zentral durch ein Team an Programmierenden entwickelten Softwareteil. Die Kommunikationswege sind damit auf ein Netzwerk von Organisationen und Teams verteilt, die allesamt zu einem zentralen, spezialisierten Programmierendenteam führen. Geringer

ist der Kommunikationsaufwand, wenn wie bei PAKET eine Softwarefirma einen Standard gestaltet. Das jeweilige zuständige Team der Softwarefirma gestaltet und programmiert einen Teil der Software, und damit konzentriert sich dort die Kommunikation.

Die Ergebnisse fasst der Schluss des Kapitels noch einmal zusammen und stellt Bezug zur Literatur aus dem 6. Kapitel her.

Anwendungsbereich – Ausgangspunkt und Grenze für den Beitrag von Software zur Arbeit

Welche allgemeinen Aussagen lassen sich über den Anwendungsbereich fällen? Je Fallstudie ist es vom Anwendungsbereich abhängig, wie nützlich Software für diesen sein kann. Das betrifft den Anteil der Datenverarbeitung im Anwendungsbereich und ob die Softwaregestaltung einen gesamten Prozess gestaltet oder nur Teile davon. Von beidem hängt ab, was der Kern der Rationalisierungsmöglichkeit ist und wie komplex die abzubildenden Prozesse und Wissensdomänen sind.

Letztendlich lassen sich die Anwendungsbereiche der Fallstudien unterscheiden, inwieweit sie sich in Software abbilden lassen. Wenn sich der gesamte Anwendungsbereich in Software übersetzen lässt, wie dies bei der Datenverarbeitung der Fall ist, kann die Softwaregestaltung mehr leisten, als wenn nur ein Teil des Anwendungsbereichs in Software übersetzt werden kann, wie dies z.B. in der Instandhaltung der Fall ist. Die Arbeit der Monteur:innen kann Software zwar steuern. Die Arbeit an den Netzen kann die Software aber nicht erledigen. Somit unterscheiden sich die Fallstudien darin, ob Software a) Arbeitende steuert und informiert oder b) Arbeitende ersetzt (Automatisierung). Fälle von c) Steuerung von Maschinen kommen in keiner der Fallstudien vor. Vielmehr geht es in den meisten Fallstudien um die Verarbeitung von Daten, z.B. Buchhaltung, Datenaustausch mit anderen EUVs oder Rechnungsstellung. Von dem Beitrag der Software zur Arbeit, d.h. wie sehr die Möglichkeiten der Softwareentwicklung genutzt werden können, hängt der primäre Fokus der Rationalisierung ab. Wenn eine hohe Automatisierung mithilfe von Software möglich ist, hat das für die Anwendung zur Folge, dass die Anwendenden meist nur nicht-automatisierbare Restfälle bearbeiten müssen. Geht es in erster Linie um die Steuerung der Arbeitenden, bedeutet dies für die Anwendung, dass die Anwendenden Teil eines digitalen Prozesses werden und dieser ihre Arbeit steuert und/oder sie für diesen Input liefern müssen.

Tabelle 19: Anwendungsbereich: Anteil der Datenverarbeitung, Fokus Rationalisierung, spezifische Folgen für Anwendung

Fall	Bereich	Anteil Datenverarbeitung	Primärer Fokus Rationalisierung	Spezifische Folgen für Anwendung
Alle	energiewirtschaftliche (Geschäfts-)Prozesse	hoch	Automatisierung	Restfallbearbeitung
INTERN1	Instandhaltung	gering	Steuerung	Fernsteuerung, Prozessintegration
KOOP2	Kund:innenservice, IoT	gering	Information	Prozessintegration

Der Anwendungsbereich bestimmt nicht nur, wie weit die Möglichkeiten der Softwaregestaltung gehen und was der Fokus der Rationalisierung ist. Er legt zusätzlich die Prozesstiefe fest und damit, wie lang die Prozesskette ist, welche die Softwaregestaltung betreffen kann. In den Fallstudien geht mit der Prozesstiefe immer auch eine zunehmende Vielfalt der Wissensdomänen einher. Vor allem bei INTERN2 ist das ein Thema, weil mehrere Fachbereiche zusammenarbeiten müssen und ein höherer Koordinationsaufwand besteht. Dieser ist bei STARTUP deutlich geringer. KOOP3 bildet keinen Arbeitsprozess ab.

Tabelle 20: Prozesstiefe und Wissensdomänen je Fall

Fall	Prozesstiefe	Energiewirtschaftliche Wissensdomänen
INTERN1	2 Prozessteile (Instandhaltung und Auftragsverarbeitung)	verschiedene Energiesparten und Netzgebiete
INTERN2	1 Prozessteil (Softwaremodul »Work-Management«)	verschiedene Fachbereiche, ein Thema: Verwaltung von Arbeitsaufgaben für den Netzanschluss
KOOP1	viele Prozessteile	viele und verschiedene Fachbereiche auf mehrere EVU verteilt
KOOP2	viele Prozessteile	viele und verschiedene Fachbereiche auf mehrere EVU verteilt
PAKET	viele Prozessteile	viele und verschiedene Fachbereiche auf viele EVU verteilt
KOOP3	keine prozessabbildende Software	diverse Anwendungsfälle
STARTUP	2 Prozessteile (Anmeldung und Übergabe an Händler)	THG-Quote (Emissionshandel E-Mobilität)

8.3. Formen des soziotechnischen Arbeitsprozesses der Softwaregestaltung

Der soziotechnische Arbeitsprozess der Softwaregestaltung ist jener Teil des Analyserahmens, der die Umsetzung der Softwaregestaltung in den Blick nimmt. Die Analyse der Interviews ergab fünf Kategorien, die zentral für diesen Arbeitsprozess zwischen Anwendung und Programmierung sind. Mit ihrer Hilfe zeigt dieser Abschnitt, wie Organisationen oder Organisationsnetzwerke im Arbeitsprozess die Arbeitskraft transformieren, um die Möglichkeiten der Softwaregestaltung zu nutzen – ob für eine Individual- oder für eine Standardsoftware.

8.3.1. Arbeitsprozess der Softwaregestaltung: zwischen zentral und dezentral

Beim Arbeitsprozess der Softwaregestaltung lassen sich die Fallstudien danach unterscheiden, ob die Softwaregestaltung eher dezentral (innerhalb der EVU: ob als Projekte, in Abteilungen oder Teams) oder zentralisiert (in einer Softwarefirma oder einem IT-DL)

passiert. Beide Extreme hängen mit der Arbeitsteilung, der Grundkoordination und der Architektur der soziotechnischen Konstellation zusammen.

Bei **dezentraler** Softwaregestaltung ist aufgrund einer Arbeitsteilung, bei der die Softwaregestaltung Teil der anwendenden Organisation (EVU) ist, der Arbeitsprozess darauf ausgerichtet, Anforderungen aufzunehmen, diese zu sammeln und aufzubereiten und direkt zwischen Anwendenden, Gestaltenden und Programmierenden zu kommunizieren. Es können sich langfristige Beziehungen, eine gemeinsame Sprache und eine gemeinsame Wissensbasis etablieren. Entsprechend helfen die digitalen Werkzeuge dabei, direkten Input der Beteiligten zu organisieren. Der softwaretechnische Zuschnitt ist individuell.

Bei einer **zentralisierten** Softwaregestaltung ist aufgrund einer Arbeitsteilung, bei der sich die EVU auf die Softwareanwendung konzentrieren, der Arbeitsprozess darauf ausgerichtet, Anforderungen verschiedener Organisationseinheiten zu koordinieren, sie in Gremien zu verhandeln und über stärker formalisierte Abläufe aufzunehmen. Dabei stehen Anforderungen im Fokus, welche für die gesamte Branche oder zumindest für mehrere EVU relevant sind. Es ist schwieriger, kooperative Beziehungen zu etablieren, weil Spannungen bestehen: zwischen verschiedenen Marktpersonen mit unterschiedlichen Interessen oder weil Hierarchien oder Führungskräfte mit unterschiedlichen Interessen und Kompetenzbereichen beteiligt sind. Erwartungen sind abzugleichen und Konflikte zu lösen. Die digitalen Werkzeuge helfen u.a. durch die Schaffung von Transparenz bei der Koordination von Handelnden, die über mehrere Organisationseinheiten verteilt sind. Der softwaretechnische Zuschnitt ist ein Standard.

Tabelle 21: Idealtypen zentraler und dezentraler Arbeitsprozess der Softwaregestaltung

Typ	Typische Erwartungen an die Rollen	Typisch für den Ablauf	Typischer Nutzen Werkzeuge	Typische kommunikative Beziehungen	Softwaretechnischer Zuschnitt
Zentral	Koordination	Gremien für Verhandlungen, Erwartungsabgleich, Konfliktlösung	Transparenz, Abstimmungen	Spannungen ausgleichen, bürokratisch	Standard
Dezentral	Anforderungsaufnahme	Anforderungen direkt sammeln und ausarbeiten	direkter Input	kooperativ, direkt, gemeinsame Sprache	individuell

Inwiefern eine Zuordnung zu einem aus dem empirischen Material der Interviews entwickelten Idealtyp so einfach möglich ist, zeigen die Fallstudien selbst und diskutiert die Zusammenfassung am Schluss dieses Abschnitts zum Arbeitsprozess der Softwaregestaltung.

8.3.2. Darstellung der Fallstudien

Die folgenden sieben Fallstudien sind anhand der Kategorien Rollen, Ablauf, kommunikative Beziehungen, digitale Werkzeuge und softwaretechnischer Zuschnitt dargestellt. Sie vertiefen und ergänzen die im obigen Überblick aufgelisteten Unterschiede und wie die Organisationen die Möglichkeiten zwischen Individual- und Standardsoftwaregestaltung nutzen.

8.3.2.1. INTERN1: erweitertes Scrum, Gestaltungsnetzwerke, langfristige Beziehungen, dezentral

In dem Fall liegt ein dezentraler Arbeitsprozess der Softwaregestaltung für eine individuelle Software innerhalb eines EVU vor. Um einen solchen zu ermöglichen und um Anforderungen aufzunehmen, setzt das EVU Methoden wie Scrum, Resonanzgruppen und Workshops ein. Indem die kommunikativen Beziehungen offen, direkt, langfristig und tief verankert im Fachbereich bzw. nah an den Monteur:innen dran sind, erleichtern sie den Austausch quer zu den Abteilungsgrenzen und Hierarchien. Neben einem Ticketsystem wird ein Chat-Kanal verwendet, der den dezentralen Input von Anwendenden für Anforderungen ohne bürokratische Hindernisse erlaubt.

Rollen: Product Owner:innen, Anforderungsmanagende, Key User:innen

Wie für einen dezentralen Arbeitsprozess der Softwaregestaltung typisch, besteht an die Rollen vor allem die Erwartung, Anforderungen aufzunehmen, und zwar sehr nah an und direkt mit den Anwendenden im Fachbereich. Dafür gibt es in dem Fall gleich mehrere Rollen: So bearbeitet selbst der Haupt-Product-Owner (neben seiner Leitungsfunktion) Anforderungen. Daneben gibt es noch drei weitere Product Owner:innen, die für einzelne Funktionalitäten zuständig sind (wie z.B. für die mobile App der Monteur:innen), und noch fünf bis sechs spezialisierte Anforderungsmanagende aus dem Fachbereich (Hochspannung, Nieder-/Mittelspannung, Gas, Wasser, Strom). Eine befragte Anforderungsmanagerin hat viele Jour fixes mit verschiedenen Bereichen (ob IT-, Fachabteilungen oder Teams). Sie beschreibt ihre Arbeit so:

> »Ich schaue mir die Prozesse an und schaue mir an: Wo haben wir Verbesserungspotenzial, wo wir mit Digitalisierung irgendwas verbessern können? Ich nehme aber auch Anforderungen auf von Anwendern, wo die einfach sagen: Hier habe ich ein Stück Papier, das will ich künftig nicht mehr ausfüllen können. Könnt ihr das digitalisieren? Oder: Hier haben wir einen Prozess, da muss ich zehnmal telefonieren, das passt mir nicht. Und dann schaue ich mir das an. Schaue noch nach rechts und links, wie es andere machen und dass wir da irgendwie Synergieeffekte vielleicht gewinnen können. Und wir überlegen uns dann Lösungen. Ich schreib dann eine Story[8], tausche mich mit dem PO[9] […] oder mit den Ansprechpartnern dort gezielt aus – je nachdem ob backend[10] oder frontend. Und dann wird es in eine Story gegossen, priorisiert und

8 Anderes Wort für Anforderung, meist im Scrum-Setting so bezeichnet.

9 Product Owner

10 Datenzugriffsschicht einer Software, im Gegensatz zu Präsentationsschicht (Frontend) mit dem Quellcode zur Darstellung der Daten.

> umgesetzt und von mir dann getestet und kommuniziert nach draußen: Ab jetzt haben wir den und den Prozess. Das mache ich dann auch noch. […] Ich telefoniere sehr viel.« (Anforderungsmanagerin)

Als letzte Stufe der Softwaregestaltung zwischen Anwendung und Programmierung existieren Monteur:innen, welche zusätzlich zu ihrer Montagearbeit als Key User:innen agieren. Sie sind Sprecher:innen, Multiplikator:innen und technischer Support. Sie installieren z.B. mobile Apps.

Nicht nur die vielen unterschiedlichen Rollen innerhalb des Fachbereichs der Instandhaltung zeigen, dass die Organisation die Ausgestaltung der Rollen für eine dezentrale Softwaregestaltung situativ anpasst. Die situative Anpassung der Rollen zeigt sich in diesem Fall auch an der Rolle des Scrum Masters. Diese übernimmt, anders als in der Scrum-Methode vorgesehen, eine Führungskraft (ein Teamleiter). Er koordiniert die Aufgaben und kümmert sich um Schwierigkeiten im Ablauf.

Ablauf: Scrum, Resonanzgruppen, Workshops

Der Ablauf besteht primär, wie für einen dezentralen Arbeitsprozess der Softwaregestaltung typisch, im Ausarbeiten und Sammeln von Anforderungen, wobei das in diesem Fall besonders feedback-intensiv passiert. Grundlage dafür ist Scrum. Das EVU setzt die Methode vollumfänglich um, wozu ein iteratives Vorgehen gehört. Neben den typischen Rollen und Artefakten (siehe 5.2.4) ist der Ablauf darauf ausgelegt, immer wieder zwischen den Beteiligten dezentral im EVU ein gegenseitiges Verständnis zu sichern, Erwartungen abzugleichen und Wissen auszutauschen, um die Anforderungen aufzunehmen. Resonanzgruppen waren vor allem zu Beginn des Projektes und bei größeren Umsetzungsideen Teil des Ablaufs. Dort sagen z.B. Programmierende, wie sie die Anforderung verstanden haben. Nach ein paar Wochen stellen sie die Änderungen den Anwendenden vor und fragen, ob sie den Erwartungen entsprechen. Der Austausch hat sich mit der Zeit immer mehr zu den Key User:innen hin verlagert. Unter anderem in Workshops erarbeiten die Anforderungsmanagenden mit den Key User:innen Anforderungen.

Kommunikative Beziehungen: zugänglich, offen, interdisziplinär und gut vernetzt

Der Fall zeigt die für einen dezentralen Arbeitsprozess der Softwaregestaltungen typischen direkten und langfristigen Beziehungen und eine gemeinsame Sprache. Anders als in anderen Organisationen stellen für Mitarbeitende des EVU die typischen Kommunikationshemmnisse interner Gestaltungsprojekte wie Team-, Abteilungssilos oder Hierarchien keine Hindernisse dar. So existieren zusätzlich zu den Resonanzgruppen oder Workshops persönliche Beziehungen vom IT-Team zum FB und es entsteht eine gemeinsame Sprache. Die Programmierenden reden direkt mit Anwendenden, vor allem, wenn sie feststellen, dass sie über eine Story zu wenig wissen.

> »Aber ja, wir kommen gut an Informationen ran. Wir müssen nicht irgendwie ewig über irgendwelche Chefs gehen und sagen: Dürfen wir mit dem so oder so sprechen. Das ist sehr angenehm, muss ich sagen.« (Programmierer)

Der befragte Architekt beschreibt eine Offenheit im Austausch:

> »Da ist, wie gesagt, eine große Offenheit der Menschen. Insofern kann das Wissen frei fließen. […] Ich hatte den Eindruck, dass wenn man fragt, man immer Antworten kriegt. Halt so gut, wie der Gegenüber das weiß, sowohl auf Fachbereichs- als auch auf IT-Seite, als auch bei schwierigen Themen, die man vielleicht erst mal überhaupt fassen muss, damit man darüber sprechen kann.« (Architekt)

Die Anforderungsmanagerin hat ein internes Netzwerk, das ihr direkte Gespräche unabhängig von den hierarchischen Strukturen ermöglicht. Sie fragt nach, wenn sie etwas nicht versteht. Sie bekommt mit, was andere Abteilungen machen, und kann Beschäftigte zusammenbringen, bei denen sie feststellt, dass sie am gleichen Thema arbeiten. Die richtige Sprache zu sprechen, d.h. die richtige Terminologie zu verwenden, ist eine Qualifikation, die von der Anforderungsmanagerin mit der Zeit erworben wurde:

> »Ich musste IT-Deutsch lernen. Wenn ich Storys schreibe, dann muss ich dies so schreiben, dass die Softwareentwicklung versteht, obwohl ich aus der Anwender-Ecke gekommen. Ja, ich muss mich da einfach in so eine IT-Denke reinschrauben.« (Anforderungsmanagerin)

Der Fachbereich hat aus Sicht des befragten Programmierers dazugelernt, kann besser formulieren, was er will. Auch die Fehlermeldungen, die bei Tests aufgenommen werden, sind qualifizierter und unterstützen so die Fehleranalyse. Es wird nicht nur geschrieben »es tut nicht« (Teamleiter IT).

Digitale Werkzeuge: dezentraler Input dank Ticketsystem und Chat-Kanal

In dem Fall geht das EVU über die für digitale Werkzeuge typische Verwendung in dezentralen Arbeitsprozessen der Softwaregestaltung hinaus. Es existiert nicht nur ein Ticketsystem, das die auf verschiedene Abteilungen verteilten Mitarbeitenden vernetzt und ihnen ermöglicht, direkt und dezentral Anforderungen aufnehmen zu können. Es existiert zusätzlich ein Microsoft-Teams-Kanal, den Key User:innen und Anforderungsmanagende des Fachbereichs verwenden. Er hat ca. 110 Mitglieder. Dort können die Teilnehmenden Informationen austauschen, Anforderungen und Probleme besprechen. Dadurch ist eine offene, hierarchieunabhängige und direkte Kommunikation möglich. Die verwendeten Werkzeuge dienen damit primär als Infrastruktur für die Zusammenarbeit, um die Kommunikation zu ermöglichen, zu dokumentieren und zu koordinieren, und weniger zur Standardisierung oder Formalisierung.

Softwaretechnischer Zuschnitt: individuelle Anforderungen der Anwendenden

Der für einen dezentralen Arbeitsprozess der Softwaregestaltung typische individuelle softwaretechnische Zuschnitt zeigt sich nicht nur darin, dass ein EVU eine Software allein für sich gestaltet. Er zeigt sich auch darin, dass die Softwaregestaltung die individuelle Sicht der Anwendenden des EVU auf zweifache Weise berücksichtigt.

Erstens verhandelt der Arbeitsprozess die Priorisierung von Anforderungen des Managements und der Anwendenden. Das Management trägt seine Anforderungen – einige Befragte sprechen von Unternehmensanforderungen – an den Haupt-Product-Owner heran. Weil die Anforderungsmanagenden näher an den Monteur:innen dran sind,

bringen sie entsprechend mehr Anforderungen der Anwendenden ein und votieren für deren Umsetzung. Auch wenn letztendlich der Haupt-Product-Owner über die Priorität von Unternehmens- oder Anwendenden-Anforderungen entscheidet, fließen individuelle Anforderungen der Anwendenden ein.

Zweitens berücksichtigt die Softwaregestaltung die Begrifflichkeiten aus dem Arbeitsalltag der Monteur:innen. Die unterschiedlichen Netzbereiche verhandeln einen Konsens, welche Begriffe auf der Softwareoberfläche auftauchen. Innerhalb des Quellcodes der Software verwenden Programmierende Begrifflichkeiten aus dem Arbeitsfeld der Monteur:innen des EVU, um sprachliche Hürden zu verringern[11].

8.3.2.2. INTERN2: gemeinsame Anforderungsrunde mehrerer Fachbereiche, dezentral

In dem Fall liegt ein dezentraler Arbeitsprozess der Softwaregestaltung für eine individuelle Software innerhalb eines EVU vor. Die Besonderheit liegt darin, dass hier, anders als bei INTERN1, mehrere Fachbereiche an der Softwaregestaltung beteiligt sind. Repräsentant:innen aus den Fachbereichen verständigen sich zentralisiert in einer Anforderungsrunde darüber, wie sie die Software gestalten. Aufgrund der fachbereichsübergreifenden Softwaregestaltung sind die kommunikativen Beziehungen durch hierarchische Hürden gehemmt, auf kooperative Fachbereiche angewiesen und wegen der zentralen Anforderungsrunde mit bürokratischen Abläufen konfrontiert. Die beteiligten Fachbereiche verwenden unterschiedliche Ticketsysteme, die aber teilweise dezentralen Input ermöglichen. Über den softwaretechnischen Zuschnitt entscheidet die Anforderungsrunde, so dass sich trotz individueller Softwaregestaltung im EVU die individuellen Wünsche einzelner Fachbereiche nicht immer durchsetzen.

Rollen: Scrum Master, Product Owner, Anforderungsmanager

Wie für einen dezentralen Arbeitsprozess der Softwaregestaltung typisch, besteht an die Rollen vor allem die Erwartung, Anforderungen aufzunehmen, und zwar sehr nah an und direkt mit den Anwendenden in den Fachbereichen.

In diesem Fall verteilen sich die Rollen auf verschiedene Fachbereiche. So ist z.B. der befragte Product Owner zu 100 % in dem dezentralen IT-Team des Fachbereichs Netzanschluss. Seine Aufgabe ist es, die Anwendungsprozesse inkl. Software und sein IT-Team weiterzuentwickeln, Anforderungen aufzunehmen und an der Anforderungsrunde teilzunehmen. Dann gibt es noch die Anforderungsmanagerin aus der Abteilung Zentralbereich. Sie arbeitet zusammen mit der IT-Abteilung Fachkonzepte zu einzelnen Tickets aus und testet die Umsetzungen. Wie auch bei vielen anderen Fallstudien gibt es die Rolle Key User:in, die den Anwendenden am nächsten ist. Wobei in diesem Fall die Möglichkeit besteht, dass alle Anwendenden sich einbringen. Der befragte Product Owner aus der dezentralen IT im Fachbereich Netzanschluss schätzt, dass für das reguläre Anfor-

11 Das nennt man Domain-Driven Design: Die Programmierung berücksichtigt die Wissensdomäne des Anwendungsbereichs, z.B. indem Fachbegriffe aus dem Anwendungsbereich in den Quellcode einfließen. Dadurch wird die Kommunikation zwischen Programmierenden und Anwendenden einfacher, weil sie die gleichen Begriffe verwenden.

derungsmanagement 80 % der 300 Anwendenden mindestens einmal an einem Jour fixe teilnehmen.

Der Scrum Master hat die Verantwortung für die Abstimmungsrunde der Fachbereiche: dass und wie sie abläuft und optimiert werden kann. Dabei nimmt er manchmal die Rolle des Mediators ein, wie es für die Rolle Scrum Master:in üblich ist.

> »[N]atürlich gibt es auch Interessenskonflikte und da muss man natürlich schauen, dass man auf den größten gemeinsamen Nenner kommt oder im Idealfall vielleicht sogar auf eine geniale Idee, die dann alle Interessen unter einen Hut bringt. Aber der Input kommt von allen Seiten, genau. Die Moderation oder auch Mediation, falls erforderlich, die liegt dann schon bei mir.« (Scrum Master)

Dabei fällt stärker als bei INTERN1 auf, was allgemein für die Softwaregestaltung typisch ist: Die Ausgestaltung der Rollen erfolgt situativ angepasst, ändert sich im Zeitverlauf und anders als bspw. in Scrum vorgesehen, sind sie nicht immer klar spezialisiert. So koordiniert der Scrum Master zusätzlich die Beauftragungen und die Auslastung der externen Programmierenden und optimiert die Arbeit des programmierenden Teams. Letztendlich ist er sich gar nicht sicher, was er ist: IT-Koordinator, Scrum Master, agiler Coach oder Projektleiter, weil er sich nicht in einem reinen Scrum-Setting bewegt und seine Rolle nicht klar geschnitten ist. Auch die befragte Anforderungsmanagerin ist nicht nur mit Anforderungen beschäftigt, sondern nimmt auch Einstellungen an der ERP-Software vor, macht Datenbereinigungen und -auswertungen.

Im Unterschied zu INTERN1 übernehmen aufgrund der zentralen Anforderungsrunde der Product Owner oder die Anforderungsmanagerin auch koordinative Aufgaben. Sie schreiben Anforderungen nicht nur selbst, sondern sammeln sie auch und bereiten sie auf.

Ablauf: Anforderungsrunde, Refinement-Termine, IT-Teams in Fachbereichen, Tests

Der Ablauf besteht primär, wie für einen dezentralen Arbeitsprozess der Softwaregestaltung typisch, im Ausarbeiten und Sammeln von Anforderungen der Anwendenden. Wobei in diesem Fall aufgrund der zentralen Anforderungsrunde auch Verhandlungen Teil des Ablaufs sind. Aus Sicht des Scrum Masters liegt kein reines Scrum vor, sondern sie verwenden Bestandteile aus unterschiedlichen Ansätzen.

Wie dezentral die Softwaregestaltung ist, zeigt sich daran, dass der Arbeitsprozess Konzepte iterativ mit Anwendenden erarbeitet. Programmierende, Gestaltende und Anwendende arbeiten eng zusammen, was sich an kontinuierlichen Feedbackschleifen zeigt. So bringen Fachbereiche Vorkonzepte in die Anforderungsrunde ein, über die sich der Fachbereich und die Programmierenden bereits direkt ausgetauscht haben. Dadurch ist die eingebrachte Anforderung näher an der späteren Umsetzung und es gibt kein Hin und Her mehr zwischen Anwendung und Programmierung, weil die Programmierenden nicht erst in der Umsetzungs- bzw. Sprintphase mit einem Konzept konfrontiert sind, sondern vorher bereits nachfragen können. Größere Fachbereiche haben dafür ein- oder zweimal die Woche feste Refinement-Termine. Der Scrum Master sitzt meistens auch in den Refinement-Terminen, leitet sie und bekommt so mit, was dort für Themen auftauchen. Kleinere Fachbereiche können in vorgegebenen Zeitfens-

tern Termine mit den Programmierenden vereinbaren. Bei kleineren Anforderungen ist ein direkter Austausch zwischen Fachbereich und Programmierenden möglich, um nachzufragen, ob ein Konzept so in Ordnung ist. Die Schleifen zwischen Fachbereich und Programmierung gehen so lange, bis der Reifegrad passt. Dann wird die Anforderung Teil eines Sprints. Neben diesen Feedbackschleifen bei der Konzeptionierung gibt es eine zweite Feedbackmöglichkeit durch Tests. Wenn ein Fachbereich Fehler entdeckt oder mit einer Umsetzung unzufrieden ist, geht das entsprechende Ticket zurück an die Programmierung.

Die dezentrale Softwaregestaltung zeigt sich zudem durch einen Ablauf, der aktiv und direkt auf Anwendende in den Fachbereichen zugeht. So nutzt das IT-Team des Fachbereichs Netzanschluss seine dezentrale Position, um Interviews durchzuführen. Dazu werden Mitarbeitende aus jedem Team eingeladen, die einzelnen Prozesse durchgesprochen und Änderungswünsche gesammelt. Die so erarbeiteten Konzepte kommen von den Anwendenden selbst und sie nehmen deren Umsetzungen gut an, so eine befragte Person. Nach der Umsetzung holt die Programmierung noch Feedback ein, ob die Umsetzung den Erwartungen entspricht.

Dass trotz der zentralen Anforderungsrunde die dezentralen, individuellen Anforderungen der Fachbereiche eine Rolle spielen, zeigt sich daran, dass die Beteiligten gleichberechtigt sind. Zur Zeit der Befragung nahmen an der Anforderungsrunde zwei Vertreter aus der IT (inkl. Programmierende), zwei aus der IT des Zentralbereichs und vier Vertretende aus den Fachbereichen[12] teil. Für eine Sitzung nimmt jede beteiligte Person ihre Anforderungen mit, die in der Woche angefallen sind. Die Anforderungen stellen die Vertretenden der Fachbereiche kurz inhaltlich vor und agieren damit in der Rolle von Product Owner:innen. Die einzelnen Fachbereiche sind dort gleichgestellt, verfügen über ein gemeinsames Budget und sollen sich demokratisch einigen.

> »Ja, wenn man so will, sind ja die Vertreter der Fachbereiche die einzelnen Product Owner. Dafür haben wir natürlich keinen ›Ober Product Owner‹, sondern wir setzen im Moment darauf, dass sich die […] [Abstimmungsrunde] schon einigen wird. Und das hat bisher auch funktioniert, so eher demokratisch.« (IT-Manager Fachbereich)

So bereiten Konflikte keine größeren Probleme. Die Teilnehmenden haben manchmal unterschiedliche Vorstellungen und Interessen, was Anforderungen an die Software anbelangt. Dort können sie sich darüber austauschen und gemeinsame Lösungen finden.

Kommunikative Beziehungen: Kooperationsbereitschaft und direkte Kommunikation

Der Fall zeigt die für einen dezentralen Arbeitsprozess der Softwaregestaltungen typischen direkten und langfristigen Beziehungen intern in einem EVU, welche die oben genannten Rollen und Abläufe ergänzen. Allerdings existieren anders als bei INTERN1 Spannungen in den Beziehungen, weil mehrere Fachbereiche beteiligt sind. Die veränderten Erwartungen an die Beschäftigten im Zuge der Softwaregestaltung spiegeln sich

12 Netzanschluss, IT-Zentralbereich (Work-Management und Messstellenbetrieb), Zählerwesen, Abrechnung.

darin nieder, dass es für manche Mitarbeitende schwer war, Kommunikation als Teil ihrer Arbeit zu begreifen.

Das EVU konnte den Arbeitsprozess der Softwaregestaltung erst etablieren, als die fehlende Kooperationsbereitschaft sowohl zwischen den Fachbereichen als auch zwischen IT-Abteilung und Fachbereichen beseitigt war. Ein Befragter bezeichnet es als »revolutionär«, dass, als sich der Fachbereich neu organisieren wollte, dieser zwei Vertretende der IT zu einigen Terminen dazu geladen hat, aus denen dann die Abstimmungsrunde hervorging. Dass es so gut geklappt hat, lag daran, dass die Beteiligten keine vorbelasteten Beziehungen hatten:

> »[W]eil sie die Historie nicht mit sich herumgeschleppt haben, wer da mal vor zehn Jahren wem irgendwie ans Schienbein getreten ist oder irgendwie was weggenommen oder irgendwas Blödes gemacht hat.« (IT-Manager Fachbereich)

Allgemein gibt es persönliche Netzwerke zwischen den Fachbereichen und der IT-Abteilung. Als die IT noch ausgelagert war, hat das Management versucht, dass Fachbereiche mit ihr nur noch über formal festgelegte Wege kommunizieren. Jetzt ist es wieder in Ordnung, sich persönlich zu kennen und anzusprechen. Der Scrum Master beschreibt die direkte Kommunikation als effizienter, weil es kein Hin und Her zwischen den Beteiligten gibt:

> »Aber wir haben halt versucht, gerade halt möglichst mehr auf eine direkte Kommunikation umzusteigen, weil gerade in der Vergangenheit gab es dann in den Tickets häufig das Phänomen, dass dann so ein Frage-Antwort-Pingpongspiel hin und her ging. Und der Durchschnitt war da tatsächlich, dass wir zu einem Ticket so 30 Kommentare hatten.« (Scrum Master)

Die Anforderungsmanagerin aus der Zentralbereichs-IT meint, dass sie bestimmte Themen mit den Programmierenden direkt bespricht. Entweder macht sie das per E-Mail, per Telefon oder, wenn das nicht ausreicht, persönlich.

Um sich direkt auszutauschen, müssen Mitarbeitende dazu bereit sein. Einige Befragte weisen darauf hin, dass dies nicht selbstverständlich ist. So haben sich ältere Programmierende anfangs nicht gern in agilen Teams ausgetauscht. Mit Scrum verbinden sie viele Termine, Kaffeekränzchen und dass man nicht zum Arbeiten kommt. Einige Mitarbeitende wollen lieber vor sich hinarbeiten, als sich regelmäßig mit anderen abzustimmen. Im Gegenteil dazu sieht die Anforderungsmanagerin Kommunikation als selbstverständlichen Teil ihrer Arbeit an.

Digitale Werkzeuge: mehrere Softwarelösungen, um Anforderungen zu sammeln

In dem Fall zeigt sich in dem EVU die für digitale Werkzeuge typische Verwendung in dezentralen Arbeitsprozessen der Softwaregestaltung in Extremform: Es geht primär darum, Anforderungen zu sammeln, und weniger um Kontrolle oder Koordination der gesamten Softwaregestaltung. Denn es gibt unterschiedliche Ticketsysteme, Web-Formulare und Exceldateien, mit denen die Fachbereiche Anforderungen für die Anforderungsrunde sammeln. Wegen dieser disparaten Landschaft an Werkzeugen existiert kein in-

tegriertes digitales Gesamtsystem, weswegen die verwendeten digitalen Softwaretools nicht zur zentralen Kontrolle taugen. Vielmehr wenden die Beteiligten pragmatisch dezentral an, was sie kennen, können und ihnen nützlich erscheint – Hauptsache, sie können Anforderungen einfach sammeln. Allerdings gibt es zentral in der Anforderungsrunde eine Liste, damit den Teilnehmenden transparent ist, welche Anforderungen wie priorisiert sind. In Zukunft sollen aber alle das gleiche Ticketsystem verwenden.

Softwaretechnischer Zuschnitt: größter Nutzen für das Unternehmen

Der für einen dezentralen Arbeitsprozess der Softwaregestaltung typische individuelle softwaretechnische Zuschnitt zeigt sich auch in diesem Fall. Die zentrale Anforderungsrunde übt jedoch einen Konsens- und Standardisierungsdruck auf die einzelnen Fachbereiche aus. Deren individuellen Vorstellungen sind dadurch Grenzen gesetzt.

Mehr noch als bei INTERN1 muss sich die individuelle Sicht der Anwendenden des EVU gegen die Unternehmenssicht durchsetzen, weil zentral mehrere Repräsentant:innen der Fachbereiche über Anforderungen und deren Priorisierung verhandeln. Maßgeblich kann z.B. sein, wie viele Anwendende eine Anforderung betrifft oder wie teuer eine Umsetzung wird. Wie wichtig die Priorisierung ist, zeigt sich an den ca. 100 Tickets im Backlog der Anforderungsrunde, die noch umzusetzen sind.

> »Also, ich sag mal, in der Abstimmungsrunde sitzen eben die Leute drinnen, die wissen, wir müssen als Gesamtunternehmen effizient sein. Ich kann nicht jedes Wünsch-Dir-Was-Thema einsteuern. Wir sind aber auch keine Anwender. Wenn ich bei mir im Fachbereich priorisiere, da sitzt ein persönlicher Leidensdruck oft dahinter, dass er sagt: Mich stört das tagtäglich in meiner Arbeit. Betrifft aber vielleicht nur fünf Mitarbeiter von den 300. Manche Themen die betreffen alle und das in Waage zu halten, das ist, ja, eine Herausforderung.« (Product Owner IT-Team Netzanschluss)

Die dezentrale Gestaltung einer individuellen Software in einem EVU zeigt sich zudem daran, dass laut einem Befragten das genutzte System stark vom SAP-Standard abweicht. Ein Teil des Arbeitsprozesses ist es, zu verhandeln, ob ein Fachbereich eine Standardausprägung von SAP nutzt oder diesen Standard anpasst. Ein Grund für eine individuelle Anpassung kann sein, dass es mehr kosten kann, wenn das EVU beim Standard bleibt. Das ist dann der Fall, wenn dadurch das EVU mehr Personal braucht, weil die Software nicht auf die firmenindividuellen Prozesse angepasst ist.

Wie auch bei INTERN1 zeigt sich die individuelle Softwaregestaltung daran, dass der dezentrale Arbeitsprozess auf EVU-eigene Begriffe Rücksicht nimmt und einen Konsens über sie verhandelt. Die von der Softwaregestaltung betroffenen Anwendenden im Netzanschluss sind zahlreich und in unterschiedlichen Netzgebieten tätig (und damit regional weit verteilt), weswegen die Mitarbeitenden teilweise die gleichen Dinge mit anderen Namen bezeichnen.

8.3.2.3. KOOP1: zentralisiertes Anforderungsmanagement, um Standard zu verhandeln

Anders als bei INTERN1 und INTERN2 geht es in dieser Fallstudie in erster Linie darum, dass mehrere EVU die Möglichkeit der Softwaregestaltung für einen gemeinsamen

Standard nutzen wollen. Dafür gibt es einen zentralen Arbeitsprozess der Softwaregestaltung in Form eines Anforderungsmanagements, welches das IT-DL organisiert und bei dem mehrere EVU mitmachen. Jedoch ist das Besondere an dem Fall, dass die EVU verhandeln, was zum Standard gehört und was sie individuell gestalten. Dementsprechend erfolgt auch die Softwaregestaltung dezentral in und für einzelne EVU und, wie für diese Form der Softwaregestaltung typisch, die Anforderungsaufnahme direkt mit den Anwendenden. Aufgrund der Marktbeziehung zwischen IT-DL und EVU herrscht eine Spannung in den kommunikativen Beziehungen: einerseits aus partnerschaftlich-kooperativem Vorgehen für den gemeinsamen Standard und andererseits nur so lange zu kooperieren, wie es dem einzelnen EVU nützt, und z.B. Vorteile aus individuellen Alleingängen zu ziehen. Die eingesetzten digitalen Werkzeuge erledigen sowohl die typischen Zwecke einer dezentralen als auch zentralisierten Softwaregestaltung. Sie dienen dazu, die Verhandlungen zwischen den Organisationen transparent zu halten, zu koordinieren, die Umsetzung des IT-DL zu kontrollieren und dezentral Input in den EVU zu ermöglichen.

Rollen: Anforderungsmanagende, IT-Beratende, Key User:innen, Anwendungsbetreuende & Co. – zwischen und innerhalb der Organisationen

Wie für einen zentralen Arbeitsprozess der Softwaregestaltung typisch, besteht an die Rollen vor allem die Erwartung, die Anforderungserarbeitung über Organisationsgrenzen hinweg zu koordinieren. Nur so ist eine stetige Verhandlung darüber möglich, was zentral das IT-DL oder dezentral die EVU gestalten. Doch gibt es Unterschiede zwischen den EVU, inwiefern sie sich a) rein auf die kooperativen, zentralisierten Anforderungsprozesse des IT-DL ausrichten oder b) selbst Rollen haben, die eine eigenständige, dezentrale Softwaregestaltung erlauben.

Für die Koordination zwischen IT-DL und EVU gibt es sowohl innerhalb des IT-DL als auch in den EVU Rollen. Innerhalb des IT-DL gibt es mehrere Rollen, welche die zentralisierte Softwaregestaltung koordinieren. So gibt es ein neunköpfiges Team für kooperative Projekte, wovon zwei für das Anforderungsmanagement zuständig sind. Daneben gibt es extra Rollen, um die Beziehungen zu den EVU zu pflegen: die Key Account Managenden. Über sie läuft alles Kaufmännische wie z.B., wenn das Budget für eine Anforderung nicht ausreicht und EVU und IT-DL sich auf das weitere Vorgehen einigen müssen. Für die Übersetzung zwischen IT- und Energiewirtschaft existieren beim IT-DL IT-Beratende, die spezialisiert sind auf einzelne Fachgebiete wie Stammdaten, Abrechnung, Ablesung oder Instandhaltung. Sie schreiben Konzepte, unterstützen Programmierende mit fachlichem Wissen oder beraten die EVU. Jede beratende Person hat zudem als Aufgabe, bei Anforderungen zu prüfen, welche EVU sie betreffen bzw. für welche sie interessant sein könnten.

Innerhalb der EVU gibt es für das vom IT-DL koordinierte Anforderungsmanagement zuständige Mitarbeitende. Ihre Arbeit zeigt, dass die zentralisierte Standardgestaltung in diesem Fall auf dezentrale Anforderungen der Anwendenden Rücksicht nimmt und sie in den Arbeitsprozess der Softwaregestaltung einbindet. Der Anforderungsmanager von EVU2 kümmert sich um die kaufmännische Abwicklung und

sammelt Anforderungen – sowohl für den Netz- als auch für den Vertriebsbereich seines EVU.

Ebenso innerhalb der EVU sind die Key User:innen für das zentrale Anforderungsmanagement aktiv. Sie sind noch näher an den Anwendenden dran, auch selbst Anwendende der Software und sie speisen Anforderungen in das Anforderungsmanagement des IT-DL ein. Ein befragter Teamleiter (EVU2) hat in seinem Team neben Key User:innen und sechs Sachbearbeitenden noch eine zusätzliche Rolle, die an der Softwaregestaltung teilnimmt: die Fachkoordinator:innen. Sie arbeiten übergreifender als Key User:innen, stärker koordinativ, strategische und fachliche Diskussionen und Umsetzungen begleitend. In EVU2 macht ein befragter Teamleiter am Anforderungsmanagement mit. Manche EVU haben keine Key User:innen und stattdessen übernimmt das IT-DL deren Aufgaben.

Die EVU verfügen zudem über Rollen, mit denen sie dezentral und unabhängig vom zentralisierten Anforderungsmanagement im IT-DL selbst Software gestalten. So gibt es im EVU2 einen Anwendungsbetreuer, der die von ihm betreuten Softwareanwendungen unabhängig vom IT-DL gestaltet: Er leitet Projekte, macht Prozessanalysen, macht Ausschreibungen und nimmt Anforderungen auf. EVU2 ist allgemein ein Beispiel dafür, dass die Verhandlung von Individual- und Standardsoftwaregestaltung auch innerhalb der EVU stattfindet. Der befragte Anwendungsbetreuer aus EVU2 meint, dass im EVU aufgrund von Sparprogrammen vor einigen Jahren IT-Querschnittsfunktionen wie seine abgebaut wurden, weil sie kein Geld bringen würden. Nun entstehen sie wieder, weil man deren Notwendigkeit erkannt hat. Laut dem Digitalisierungsmanager von EVU2 agieren die Fachbereiche unterschiedlich: Die einen binden die IT stärker ein und arbeiten mehr mit ihr zusammen. Andere stellen selber Key User:innen ein und bauen ERP-Know-how auf, um eigenständig Einstellungen an der Software vornehmen zu können. Aus seiner Sicht gibt es einen Wellenverlauf zwischen De- und Zentralisierung.

Der Prozessmanager aus dem Vertrieb von EVU3 koordiniert dezentral die Gestaltung des CRM-Moduls (Customer-Relationship-Management[13]). Wie für dezentrale Arbeitsprozesse der Softwaregestaltung typisch, fließen die individuellen Anforderungen von Anwendenden ein:

> »Also, zum großen Teil kommen diese Anforderungen von den Anwendern. Weil letztendlich sie ja in dem System arbeiten und dort Probleme und Anforderungen, um den Prozess zu optimieren, besser einsehen können.« (Prozessmanager EVU3)

Er prüft deren Anforderungen und den Ist-Zustand der Software, spricht mit dem IT-DL und internen Mitarbeitenden und schreibt Anforderungen für das Anforderungsmanagement, die dann bei den Programmierenden landen. Zudem informiert er Anwendende, was geändert wurde, oder organisiert bei größeren Veränderungen Schulungen.

Ein letztes Beispiel für dezentrale Softwaregestaltung ist die Prozessmanagerin aus EVU1. Sie arbeitet eigenständig mit den IT-Beratenden des IT-DL zusammen, tauscht

13 Software zur Erfassung und Verwaltung von Daten der Kundschaft vor allem zu vertrieblichen Zwecken (bspw. um Marketingaktionen durchzuführen).

sich mit Programmierenden aus und spricht Softwarekonzepte mit den involvierten Anwendenden durch, damit diese die Veränderungen verstehen und sinnvoll finden. Zugleich nimmt sie selber Einstellungen an der Software vor.

Ablauf: zentral organisiertes Anforderungsmanagement und dezentrale Softwaregestaltung

Der Ablauf besteht primär daraus – wie für einen zentralisierten Arbeitsprozess der Softwaregestaltung typisch –, über Anforderungen zu verhandeln, Konflikte zu lösen und Feedback von verschiedenen EVU einzuholen. Wobei in diesem Fall trotzdem auch dezentral die EVU von den Anwendenden Anforderungen aufnehmen. So gibt es aufgrund der Mischung aus zentraler und dezentraler Softwaregestaltung Feedbacks zwischen Programmierenden, Anwendenden und Softwaregestaltenden in einem weitverzweigten Gestaltungsnetzwerk.

Die für eine zentralisierte Softwaregestaltung typischen Konflikte zwischen den EVU moderiert das IT-DL im Anforderungsmanagement auf operativer und strategischer Ebene. Auf der operativen übernehmen das die Anforderungsmanagenden des IT-DL. Dort bringen die EVU ihre jeweiligen Erfahrungen und Sichtweisen in die Diskussion über die Anforderungen ein. Dann teilt jedes EVU mit, welche Anforderungen es haben will und welche nicht – je mehr EVU eine Anforderung gemeinsam umsetzen, umso günstiger wird es. Alle zwei Wochen treffen sich die EVU dafür in einer Telefonkonferenz. Auf der strategischen Ebene des Anforderungsmanagements ist ein professioneller Mediator aktiv. Der Mediator kann durch seine Neutralität zwischen den verschiedenen Interessen des Dienstleistungsunternehmens und der EVU vermitteln. Die EVU müssen sich auf gemeinsame Ziele einigen, wenn sie einen Standard gestalten wollen. Die Treffen behandeln u.a. größere Themen oder Projekte wie z.B. eine Umstellung auf eine neue Version des ERP-Systems.

Das Besondere an dem Fall ist nun, dass nicht einfach Fachleute zentral einen Standard gestalten, sondern dezentral die EVU von den Anwendenden Anforderungen sammeln. Dafür müssen die EVU intern für sich herausfinden, wie sie die Möglichkeiten der Softwareentwicklung nutzen wollen: Welche Anforderungen nur für sie und welche für das zentrale Anforderungsmanagement relevant sind und damit in den kooperativen Standard einfließen. Die EVU müssen für sich herausfinden, wie sie sich intern organisieren, wenn sie für die Softwaregestaltung mit mehreren Organisationen zusammenarbeiten. Das EVU2 macht jede Woche Anforderungsmanagementrunden mit den Anforderungsmanagenden aus den unterschiedlichen Bereichen. Dort tauschen sich die beteiligten Personen über Anforderungen aus, die für das zentrale Anforderungsmanagement relevant sind. Daneben gibt es ca. einmal im Monat Key-User-Runden, in denen sich die Key User:innen der Fachbereiche treffen, um z.B. abzustimmen, was sie gemeinsam gestalten.

Gleichzeitig gibt es für Projekte ohne das IT-DL dezentral eigenständige Abläufe zur Softwaregestaltung. Der verantwortliche Applikationsbetreuer von EVU2, der für die Software zur Instandhaltung zuständig ist, führt als Projektleiter Workshops durch, um Anforderungen aufzunehmen. Er sieht agile Anleihen in seiner Art des IT-Projektmanagements: Die Arbeitspakete ergeben sich iterativ, es gibt vierwöchige Sprints inklusive Reviews und Retrospektive, wie es in Scrum üblich ist. Das heißt, er arbeitet

direkt Anforderungen für Programmierende aus und stimmt sich mit diesen ab. So existiert dezentral ein direkter Feedbackmechanismus zwischen Anwendung und Programmierung. EVU3 koordiniert intern für sich die Gestaltung einer Softwarelösung zur Betreuung seiner Kundschaft, die sowohl das IT-DL als auch zwei Softwarefirmen entwickeln. Das heißt, intern muss das EVU nicht nur Anforderungen aufnehmen, sondern auch die unterschiedlichen Organisationen koordinieren, welche die Programmierung übernehmen. Weil mehrere Fachbereiche des EVU betroffen sind, hat das EVU intern ein Anforderungsmanagement etabliert, damit nicht alle betroffenen Mitarbeitenden direkt ein Ticket an das IT-DL schreiben und sich die verschiedenen Teams stattdessen absprechen, was die Anforderungen anbelangt. Der befragte Prozessmanager spricht von einer »stadtwerkemodifizierten Scrum-Lösung«.

Der Grund für die dezentrale Softwaregestaltung innerhalb der EVU liegt auch darin, dass der zentralisierte Arbeitsprozess zu bürokratisch ist:

> »Aber wir machen eigentlich alles, was prozessual selber durch uns zu heben ist, selber. Weil erstens schneller, günstiger und ja, runder irgendwie. Sonst muss man immer Tickets schreiben, Ressourcen anfordern, feststellen, dass man die Ressource erst in Wochen bekommt, dann testen und irgendwann freigeben. Da wo es halt geht, machen wir es halt möglichst agil selber.« (Teamleiter EVU1)

Typisch für eine zentralisierte Softwaregestaltung und Teil der Feedbackschleife zwischen Anwendung und Programmierung sind die vom IT-DL zentral koordinierten Tests. Sie sind ein wichtiger Teil der Softwaregestaltung, weil sie Rückmeldung darüber geben, ob eine Umsetzung den Vorstellungen der EVU entspricht. Neben dem gemeinsamen Anforderungsmanagement testen die EVU und das IT-DL gemeinsam. Das IT-DL hat extra Rollen für das Testmanagement, um die Tests zu koordinieren. Dezentral hat z.B. EVU2 selber noch eine Person, die Testfallkataloge schreibt und sie an die Fachbereiche verteilt.

Kommunikative Beziehungen: partnerschaftlich und bei Bedarf direkt

In dem Fall existieren die für einen zentralen Arbeitsprozess der Softwaregestaltungen typischen bürokratischen und spannungsgeladenen marktförmigen Beziehungen, welche die Kommunikation erschweren. Doch sorgen neben Mediator, kooperativen Rollen und Abläufen kommunikative Beziehungen dafür, dass der Arbeitsprozess der Softwaregestaltung trotzdem gelingt. So gibt es langfristige Beziehungen auf Augenhöhe und direkte Kommunikation trotz formal-bürokratischer Abläufe. Trotz Marktbeziehungen zwischen EVU und IT-DL sprechen Beteiligte von einer Familie und von einem Geben und Nehmen.

Über das IT-DL Software zu gestalten, bringt einen langwierigen, formalen Prozess mit sich, an den sich die beteiligten Rollen ausrichten müssen:

> »Ticket, Angebot schreiben, Change im System, Alphatest, Kundentest, Go-Live, Stabilisierung. Ich habe ja jedes Mal eine Destabilisierung des Systems« (Key Account Manager).

Solche Hürden der Softwaregestaltung ergeben sich auch durch interne Prozesse in den EVU. Der befragte Applikationsbetreuer aus EVU2 meint, dass es dauert, bis Anforderungen durch die interne Hierarchie gehen, weil die Befehlskette für Abstimmungen zu berücksichtigen ist.

Kommunikative Beziehungen wie direkte Kontakte verringern diese Kommunikationshürden. Ein befragter Programmierer arbeitet zwar viel via Ticketsystem und ohne direkten Kontakt zu den EVU. Gleichzeitig ist er im Gegensatz zu früher wieder telefonisch erreichbar. Er ruft manchmal direkt bei einem EVU an und diese wiederum können ihn direkt auf seinem Diensthandy erreichen. Das erleichtert seiner Meinung nach die Abstimmung.

> »Mittlerweile ist mir das aber lieber, wenn ich auch direkten Kundenkontakt habe. Ich rufe dann auch manchmal direkt an. Ich habe ein Diensthandy. Die Leute sehen dann auch meine Mobilfunk-Nummer und können mich bei Nachfragen oder auch direkt erreichen. Das erleichtert meiner Meinung nach die Tätigkeit, die Abstimmung.« (Programmierer1)

Der andere befragte Programmierer2 wird bei bestimmten Themen (für die er bekannt ist) von den EVU direkt angesprochen und danach wird eine Anforderung aufgenommen. Ein Prozessmanager aus einem EVU meint:

> »Man hat einfach diesen Stille-Post-Effekt. Wenn ich eine Anforderung aufgebe, […] auf der [IT-DL]-Seite einen Berater gibt, der diese Anforderung aufnimmt und diese dann mit dem Entwickler bespricht, dann ist es halt manchmal so, je nachdem, wie weit schon der andere den anderen kennt oder halt eben auch nicht, dass manchmal nicht das Richtige dabei rauskommt. Und ich glaube schon, dass es einfacher wäre, wenn wir im Haus intern eine Entwicklungsabteilung hätten und man dann direkt mit den Entwicklern spricht, denen die Anforderung geben könnte. Dann hätte man halt eine Schnittstelle weniger.« (Prozessmanager EVU1)

In den Interviews werden auch verschiedene Elemente der Kommunikation genannt, die für die Zusammenarbeit hilfreich sind, wie Offenheit und professionelle Kommunikation:

> »Das wichtigste Konzept in unserem Bereich ist wirklich die Offenheit, da wirklich auch Leute zu integrieren, mitzunehmen, abzuholen, die immer wieder auch… Es gibt ja immer diese schöne Bringschuld. Das wird bei uns nicht funktionieren, wenn nicht jeder auch wüsste, es kann wichtig sein für die Kollegen, die oder die Veränderung zu wissen. […] Wie gesagt, bei uns läuft alles über Offenheit.« (Programmierer1)

Ein befragter Betriebsrat betont die Erwartungen an die Softwaregestaltenden, nicht nur das IT-System zu betreuen, sondern den Anwendenden zuzuhören, zu verstehen, zu übersetzen, Lösungen anzubieten, die Sprache der Anwendenden zu sprechen. Das hätte sich seit Mitte/Ende der 2010er Jahre verändert.

> »Dieses Verstehen, das einander verstehen und zuhören. Und das ist etwas, was meine Erfahrung ist: Also, bei uns hat das was mit Menschen zu tun. Wir haben die richtigen Leute eingestellt in den letzten Jahren und die sind extra…, die sind Kommunikationsprofis, auch IT-Profis« (Betriebsrat)

Neben dem direkten Austausch ist die langfristige Zusammenarbeit eigentlich charakteristisch für eine dezentrale Softwaregestaltung. Sie spielt aber auch in diesem Fall eine Rolle. Ein EVU hat mal das IT-DL gewechselt, ist aber wiedergekommen. Es hat erst beim neuen Dienstleistungsunternehmen gemerkt, was es am alten und dessen Wissen über das eigene EVU hatte. Es geht auch darum, ein gemeinsames Verständnis entwickelt zu haben:

> »Naja, je besser man mit den Personen sprechen kann und je besser man sich kennt, in der Regel ist es auch, umso besser werden dann auch die Anforderungen umgesetzt. […] Weil man einfach eine andere Sprache spricht, wenn man die gleichen Leute und wenn man weiß, wie die sozusagen ticken und die auch wissen, wie man selber so tickt. Ich glaube schon, je länger man miteinander zusammenarbeitet, dass, umso besser wird das Ergebnis. Vorausgesetzt, dass man gut miteinander zusammenarbeitet und nicht gegeneinander.« (Prozessmanager EVU1)

Das spielt auch bei individueller Softwaregestaltung dezentral für ein EVU eine Rolle. So kennt durch die lange Zusammenarbeit das IT-DL die individuellen Anpassungen im Bereich CRM für das EVU3 sehr gut. Das IT-DL war bereits bei der Einführung dabei. Zudem sind die IT-Beratenden fest den EVU zugeteilt und zugleich programmieren sie auch. Laut dem befragten Prozessmanager (EVU3) ist deswegen keine detaillierte Beschreibung bei Anforderungen notwendig. Er hat mit dem CRM-Team beim IT-DL ein kollegiales Verhältnis.

Die EVU und das IT-DL arbeiten trotz Marktbeziehungen kooperativ zusammen und sprechen von einer Art Familie. Denn trotz Konkurrenz und Kostenkalkül arbeiten sie partnerschaftlich zusammen und kennen sich untereinander. Die Kooperation ist partnerschaftlich, weil es ein Geben und Nehmen ist. Das kooperative Verhalten zeigt sich auch daran, dass die EVU die in den SLA festgelegten Möglichkeiten nicht ausreizen, Strafen einzutreiben.

> »Das ist ein sehr schönes Arbeiten, muss ich sagen, mit dem Kunden. Ich habe vielleicht auch das große Glück, dass ich auf der anderen Seite auch, sagen wir mal, Menschen habe, die sehr angenehm sind. Es ist aber auch bei anderen Kunden, wo es manchmal vielleicht etwas hitziger zugeht, wie ich so bei den Kollegen höre, ist es aber immer die Situation und der Alltag getrieben eigentlich von der Einstellung: Es muss ein Geben und Nehmen sein. Selbst bei den SLA: Es gibt natürlich SLA-Verstöße. Es sitzt bei keinem Kunden einer, der nur darauf wartet und sich freut und dann sofort die Rechnung und den Taschenrechner raus und sagt: ›Hier drei SLA-Verstöße […] Ich hätte gerne 15500 Euro für den Monat.‹ Diese Dinge werden immer erst besprochen, die Gesamtgemengelage, Ursachen, vielleicht schwierige Rahmenbedingungen mit besprochen, Herstellerqualität, Corona: Das kommt alles jetzt, auch wenn es jetzt ak-

tuell, das kommt alles zusammen und das ist ein Geben und Nehmen.« (Key Account Manager)

Es kommt vor, dass sich EVU unabhängig vom IT-DL untereinander absprechen und Änderungen vornehmen:

»… wir auch teilweise Kontakt zu den anderen Häusern aufnehmen, Ideen aufzugreifen, weiterzuverfolgen oder abzustimmen. Die Stadtwerke so und so hat schon eine Lösung dafür. Dann machen wir einfach einen Termin mit denen. Ich nehme meine Leute dazu, wie die das gebaut haben und überlegen uns noch, ob wir das auch wollen.« (Teamleiter EVU1)

Digitale Werkzeuge: mehrere Ticketsysteme und ERP-Entwicklungsumgebung

So wie der Fall zwar um eine zentralisierte Softwaregestaltung kreist, aber die dezentrale Softwaregestaltung in den EVU zulässt, so ist auch die Verwendung der digitalen Werkzeuge eine Mischung aus beiden Typen. Die digitalen Werkzeuge verwenden die EVU sowohl typisch für zentrale Arbeitsprozesse, um Abstimmungen und Transparenz gewährleisten, als auch für direkten Input durch Anwendende, was typisch für eine dezentrale Softwaregestaltung ist. Die kooperativen Beziehungen zwischen IT-DL und EVU drücken sich darin aus, dass direkte E-Mails an einzelne Mitarbeitende eine große Rolle spielen.

So ist im vom IT-DL bereitgestellten MS Sharepoint für sämtliche EVU einsehbar, welche Anforderungen im Anforderungsmanagement für den gemeinsamen Standard vorliegen und was die Umsetzung jeweils kostet. Das MS Sharepoint dient zur Abstimmung darüber, wer welche Anforderungen haben will und bezahlt. Die Ergebnisse der Abstimmungen inklusive der gesammelten Anforderungen sind dort ebenso dokumentiert.

Zusätzlich betreibt das IT-DL ein eigenes Ticketsystem für die gesamte Kommunikation und Dokumentation von Tickets. Das betrifft kleinere Anforderungen und Störungen bezüglich der Software. Die EVU können online Tickets aufgeben. Wenn die EVU für ein Ticket die betroffene Software und das fachliche Thema auswählen, dann ordnet das Ticketsystem das Ticket automatisch dem verantwortlichen Team beim IT-DL zu. Über das Ticketsystem können die EVU aber auch das IT-DL kontrollieren, z.B. wie lange eine Umsetzung gedauert hat.

Der Prozessmanager der dezentralen Softwaregestaltung im EVU3 nutzt nicht das Ticketsystem des IT-DL. Wie für dezentrale Arbeitsprozesse der Softwaregestaltung typisch, haben die digitalen Werkzeuge primär die Funktion, dass Anwendende mitgestalten können. Dafür setzt der Prozessmanager die MS-Office-365-Lösung ein, welche die Mitarbeitenden gut kennen. Dort gibt es einen MS-Teams-Kanal, der direkten Austausch und Input ermöglicht.

Soziotechnischer Zuschnitt: zentrale Synergien durch gemeinsamen Standard und dezentrale Abweichungen

Weil in dem Fall die EVU für die gesamte industriespezifische Softwarelandschaft verhandeln, was sie als Standard gestalten und was individuell, zeigt sich hier besonders,

wie der softwaretechnische Zuschnitt zustande kommt. So prägen a) institutionelle Unterschiede der Geschäftsbereiche Vertrieb und Netz die Möglichkeiten eines Standards. Es bestehen aber auch b) unabhängig davon Unterschiede zwischen den Firmen, inwiefern sie individuell gestalten. Zudem müssen die EVU c) intern die Kompetenzen haben, um überhaupt Synergien erkennen zu können.

Zu a): Allein durch die institutionelle Begebenheit, dass Prozesse im Netzbereich stärker durch die Regulierung bestimmt sind, gibt es dort in den EVU weniger individuelle Ausprägungen in der Softwaregestaltung. Im Vertriebsbereich ist ein Konsens bei Anforderungen schwieriger, weil individuelle Vorstellungen bei den einzelnen EVU vorliegen. Laut einem Anforderungsmanager (EVU2) geht auf der Vertriebsseite immer mehr zurück, auf was man sich in der Kooperation einigen kann. Wenn der Vertrieb sagt, er braucht das und das für den Markt, dann ist es schwer, den Standard zu halten, vor allem wenn dafür Geld in den EVU da ist.

> »Und das sind ja Vertriebe und das können Sie sich sicherlich vorstellen, die Vertriebe sind ja teilweise völlig unterschiedlich ausgerichtet.« (Anforderungsmanager EVU2)

Ein Beispiel dafür sind die verschiedenen Strom- und Gastarife, die man der Energiekundschaft anbietet. Auf der Netzseite ist der Gestaltungsspielraum durch die Regulierung eingeschränkt, weshalb es dafür in der Kooperation standardisierte Tarife gibt. Im Vertrieb sind die Produkte individueller, weswegen einen Standard auszuprägen schwieriger ist. Ein Befragter aus dem EVU1 nimmt eigenständig Einstellungen an der Software vor, um der Kundschaft des EVU unterschiedliche Produkte anbieten zu können.

Zu b): Neben den Unterschieden zwischen den Geschäftsbereichen gibt es auch noch firmenbedingte. Ein befragter Betriebsrat (EVU4) meint, dass früher das EVU sehr viel individuell entwickelt hat und man jetzt trotz einzelner individueller Programmierungen auf den Standard durch das IT-DL setzt.

> »Man hat das bei uns in diesem Projekt immer genannt: Hier gibt es blau, grün, gelb karierte Maiglöckchen für jeden Anwender. Also, es wurde alles versucht, in SAP zu programmieren. Selbst der kleinste Fall. […] Das hat uns immer sehr viel Geld gekostet. […] Jetzt haben wir natürlich sehr viel auf Standard gesetzt.« (Betriebsrat EVU4)

Keines der befragten EVU sagt, dass es rein auf den Standard setzt. Ein Befragter meint, dass bei der Einführung versucht wurde, am Standard zu bleiben, und die Fachabteilungen argumentieren mussten, warum eine individuelle Anpassung notwendig sei. Beispiele dafür, dass EVU vom Standard abweichen, sind eine unabhängig ausgewählte und angepasste Instandhaltungssoftware (EVU2) oder unterschiedliche Lösungen für das Energiedatenmanagement (EVU1). Bei EVU3 hat die starke Orientierung an der Kundschaft dazu geführt, dass es ein individuelles Portal für seine Kundschaft entwickelt hat:

> »Wir haben versucht, es immer unseren Kunden recht zu machen. Das führt dazu, warum der Standard sehr umgebaut worden ist zu einem sehr individuellen Produkt.« (Prozessmanager, EVU3)

EVU3 nutzt das Modul von SAP für das Customer-Relationship-Management (CRM) dafür ganz anders als die anderen EVU der Kooperation. EVU3 will es für Privat-, Geschäftskundschaft und den ÖPNV nutzen und regionale Besonderheiten berücksichtigen. Es gestaltet lieber dezentral selbst, weil die Abstimmung mit anderen EVU zu aufwendig ist. Anders als andere EVU setzt EVU3 seit längerem auf das CRM-Modul von SAP.

> »Entweder müssten alle anderen Häuser diese Entwicklungsschritte gehen, die wir gehen, oder wir müssten die Entwicklungen, die wir haben, irgendwie erst einmal beiseite stellen und sie in dieser Kooperation mit den anderen Partnern entwickeln, obwohl wir sie eigentlich schon haben (lacht). Und da ist halt die Frage, inwieweit da tatsächlich dann ein Mehrwert ist oder ob es da nicht kostengünstiger in Anführungsstrichen ist, wenn wir für uns selbst eine Plattform aufstellen.« (Prozessmanager EVU3)

Zu c): Unabhängig von solchen Unterschieden war es und ist es ein langwieriger Prozess, die Fähigkeiten zu entwickeln, um Synergien zu erkennen. Im EVU2 wurde die Rolle Prozessberatung 2020 geschaffen, um bereichsübergreifende Aufgaben zu erledigen. Sie hat nicht nur die Aufgabe, Gestaltungsprojekten durchzuführen. Sie soll auch mögliche Synergien feststellen und Entscheidungsvorlagen für übergreifende Optimierungen erarbeiten. Die Rolle soll dafür sorgen, dass Fachbereiche nicht unabhängig voneinander Anforderungen an das IT-DL stellen, sondern der Gesamtzusammenhang berücksichtigt wird. Durch den Blick auf die Organisation von den digitalen Prozessen und nicht allein von der energiewirtschaftlichen Fachlichkeit her hat das EVU eine neue Perspektive gewonnen. Erst dadurch ist aufgefallen, dass Anforderungen aus unterschiedlichen Fachbereichen zusammenhängen und z.B. Synergien möglich sind.

8.3.2.4. KOOP2: EVU und IT-DL zwischen dezentraler und zentraler Softwaregestaltung

Wie bei KOOP1 nutzen die Organisationen in diesem Fall sowohl die Möglichkeiten der individuellen als auch der Standardsoftwaregestaltung. Doch es gibt einen Unterschied: Es gibt keine zentralisierte Verhandlung darüber beim IT-DL und nur einen kleinen Teil der Softwaregestaltung erledigt das IT-DL zentral für die EVU in Form eines gemeinsamen Standards. Der Grund für die fehlende Zentralisierung liegt darin, dass IT-DL und EVU die Spannungen aufgrund der Marktbeziehungen nicht immer beseitigen konnten. Deshalb etablieren einzelne EVU wie EVU2 verstärkt selbst dezentrale Arbeitsprozesse der Softwaregestaltung. Das geht bei dem EVU so weit, dass es gemeinsame Projekte mit anderen EVU initiiert, wofür es die entsprechenden, typischen koordinierenden Rollen einer zentralisierten Softwaregestaltung hat. So muss in dem Fall von prekär-kooperativen kommunikativen Beziehungen gesprochen werden, und zwar auch für die dezentralen Arbeitsprozesse innerhalb der EVU: weil sie, nachdem das IT-DL die Koordination der Softwaregestaltung nicht mehr übernimmt, erst intern lernen müssen, abteilungsübergreifend für Softwareprojekte Beziehungen aufzubauen und zu kommunizieren. Weil es keine eingespielte, langfristige zentralisierte Abstimmung zwischen IT-DL

und EVU gibt, gibt es auch keine klare Systematik des softwaretechnischen Zuschnitts und damit, welches EVU welchem Standard folgt und was EVU individuell gestalten.

Rollen: verteilte Rollen für zentralisierte und dezentrale Softwaregestaltung (IT-koordinierender Fachbereich, Key User:innen, IT-Projektleitende & Co.)

Da es in diesem Fall sowohl zentrale als auch dezentrale Arbeitsprozesse der Softwaregestaltung gibt, existieren auch die typischen Rollen für beide. Aber anders als KOOP1 zeigt der Fall noch einmal besonders, was passiert, wenn EVU anfangen, dezentral Software zu gestalten, und sie dafür intern die entsprechenden Rollen brauchen. Die EVU brauchen dann noch mehr koordinierende Rollen als bei KOOP1, weil sie neben der Anforderungsaufnahme auch noch die Umsetzung selbst koordinieren (z.B. externe Freelancer oder das IT-DL beauftragen).

EVU2 koordiniert die Softwaregestaltung wieder verstärkt selbst. Es verändert seine internen Strukturen nur dahingehend, dass sie[es?] diese um einzelne Rollen für die Softwaregestaltung ergänzt. Diese agieren dann fachbereichsübergreifend wie auch beschränkt auf einzelne Fachbereiche und arbeiten teilweise eigenständig mit externen Softwareunternehmen zusammen. Zusätzlich gibt es extra Rollen wie eine Projekt-Coachin, um die Bedingungen für die Softwaregestaltung im EVU zu verbessern.

Zuerst zu den fachbereichsübergreifenden Arbeitsprozessen der Softwaregestaltung: Hier zeigt sich an den Rollen, dass das EVU sich auf Softwaregestaltung einstellt, dafür die althergebrachten Strukturen aber nicht radikal ändert. Das hat zur Folge, dass nicht die direkte Anforderungsaufnahme im Mittelpunkt steht, sondern die Koordination der Softwaregestaltung quer zu den althergebrachten Strukturen aus Hierarchien und Abteilungsgrenzen. So kann zwar der befragte IT-Projektmanager des EVU2 dank seiner Position in der Stabstelle Netzbereich, die hierarchisch über den Abteilungen steht, ein zentralisiertes, fachbereichsübergreifendes Softwaregestaltungsprojekt steuern. Er kann als Bindeglied zwischen kaufmännischen und technischen Fachbereichen fungieren. Dabei übernimmt er aber vor allem eine koordinierende Rolle: innerhalb des EVU und zu Externen wie IT-DL oder anderen EVU. Das Projekt, das er leitet, findet zusammen mit anderen EVU statt und es betrifft nicht nur unterschiedliche Fachbereiche, sondern auch unterschiedliche Systeme.

> »Netzbetrieb bedeutet das, das, das und wir brauchen dafür die und die Systeme und die hängen so und so zusammen. Und danach kommt dann Abrechnung, Bilanzierung, was auch immer zusammenhängt. Was ich brauche… Ja, fachbereichsübergreifende Kompetenz über Prozesse, wie hängt etwas zusammen, und ich kann dann erst die Systeme auch verorten und damit die Schnittstellen.« (IT-Projektleiter EVU2)

Unabhängig von einzelnen Projekten hat EVU2 die Aufnahme von Anforderungen und die Koordination der Softwaregestaltung in den einzelnen Fachbereichen langfristig bürokratisch gelöst, indem es die bestehenden hierarchischen Strukturen ergänzt hat. So hat es die Rollen der IT-Koordinierenden sowohl auf IT- als auch auf Fachbereichsseite für die Softwaregestaltung für jeden Fachbereich geschaffen. Die technischen IT-Koordinierenden aus der IT-Abteilung kümmern sich z.B. um die Abrechnung mit IT-DL und kontrollieren das Budget. Die fachlichen IT-Koordinierenden aus den Fachberei-

chen sind u.a. dafür zuständig, innerhalb des Fachbereichs für einen durchgängigen digitalen Prozess zu sorgen und Anforderungen aufzunehmen. Sie informieren die technischen IT-Koordinierenden u.a. über Anforderungen oder Aufträge an einen IT-DL.

Somit hat das EVU die althergebrachte Organisation auf die Softwaregestaltung eingestellt, indem es IT-Projekte durchführt und zentrale und dezentrale Koordinationsrollen in den Fachbereichen und der IT-Abteilung etabliert. Weil das EVU gemerkt hat, dass diese Strukturen noch nicht reichen, hat es zusätzlich eine Projekt-Coachin eingestellt, die die Projektarbeit durch verschiedene Maßnahmen verbessern soll. Sie führt Schulungen zur besseren Kommunikation unter Projektbeteiligten und in Workshops durch.

Im gleichen EVU2 gibt es aber auch ein Beispiel für einen dezentralen Arbeitsprozess der Softwaregestaltung unabhängig von den IT-Koordinierenden, der sich auf die Anforderungsaufnahme und -umsetzung konzentriert und weniger koordinative Aufwände betreiben muss. In dem Fachbereich für Marktkommunikation kümmert sich die Teamleiterin um die Weiterentwicklung der Marktkommunikationssoftware, die zwar an das SAP-ERP-System angeschlossen, aber nicht von SAP ist. Dafür arbeitet sie direkt mit einem Prozessmanager der IT-Abteilung, einem IT-Berater der Softwarefirma der Marktkommunikationslösung und den Key User:innen ihres Teams zusammen. Die Key User:innen ihres Teams sammeln die Anforderungen von den anderen Anwendenden ein und sind in mehrfacher Hinsicht Übersetzende:

> »Im Prinzip sind meine Leute nicht nur Key User, sie sind Prozess Manager, sie sind Problemlöser, sie sind die Schnittstelle zwischen Sachbearbeiter und der Entwicklung. Also, die müssen es übersetzen, dass es der Entwickler versteht und umgekehrt auch der Sachbearbeiter.« (Teamleiterin Mako EVU2)

Solche Rollen für eine dezentrale, von der internen Hierarchie unabhängige Softwaregestaltung gibt es auch in anderen EVU. Im EVU1 hat der befragte Sachbearbeiter, obwohl er kein Key User ist, zusätzlich die Rolle, regelmäßig bei Treffen zu Anforderungen für eine Software zur Abrechnung von Anlagen erneuerbarer Energien dabei zu sein.

Dezentrale Softwaregestaltung zu ermöglichen und zuzulassen, ist auch ein Thema bei EVU3. Es hat vor zweieinhalb Jahren entschieden, die Kompetenzen zum Thema Digitalisierung zu bündeln. Vorher haben die Fachbereiche selbstständig bei der Anpassung von SAP agiert. Jetzt wird zentral koordiniert, um eine Gesamtsicht über die Digitalisierung zu haben (auch zu Themen wie IT-Sicherheit, Datenschutz, Softwarearchitektur). Die Rolle des befragten Manager Digitalisierung ist Teil dieser Strategie. Er kümmert sich darum, eine zentrale, digitale Infrastruktur für Softwareentwicklungen zu schaffen. Diese soll es u.a. erleichtern, externe Programmierende einzubinden. Er initiiert Softwareentwicklungsprojekte, die dann auf dieser zentralen Infrastruktur stattfinden.

Weil die EVU intern mehr Know-how bei der Softwaregestaltung aufbauen und koordinieren, haben auch typische Rollen wie IT-Beratende beim IT-DL andere Aufgaben: Anders als in anderen Fällen programmiert der befragte IT-Berater des IT-DL hauptsächlich, hat kaum eine vermittelnde oder fachlich beratende Funktion und wird flexibel eingesetzt. Die EVU kaufen nur seine Leistung als Programmierer. Anders als die EVU setzt das IT-DL zwar Scrum ein. Wie in den anderen Fallstudien lebt es die Rollen aber

anders. Der Scrum Master ist gleichzeitig Projektleiter, Product Owner und macht bei Softwaretests mit.

> »Ich sage mal, die Rollen sind ein bisschen anders, nicht ganz so scharf geschnitten. Der Scrum Master ist eigentlich auch Projektleiter oder Product Owner. Das verschwimmt dann häufig ein bisschen.« (IT-Berater)

Ablauf: dezentral und zentralisiert in den EVU

Da die EVU zum Teil selbst Software entwickeln – mit mehreren Fachabteilungen und in Zusammenarbeit mit anderen EVU –, finden sich in diesen EVU typische Abläufe für zentrale Arbeitsprozesse, um Anforderungen zu verhandeln und Konflikte zu lösen. Gleichzeitig zeigen die Beispiele dezentraler Arbeitsprozesse der Softwaregestaltung in den EVU den typischen Fokus auf das Ausarbeiten und Sammeln von Anforderungen, mit den dazugehörigen dezentralen Gestaltungsnetzwerken und Feedbackschleifen zwischen Anwendung und Programmierung. Das IT-DL ist dann nur noch in die von den EVU koordinierte Softwaregestaltung eingebunden, weil es Programmierende für sie abstellt oder einzelne Anforderungen umsetzt.

Wie die Rollen bereits gezeigt haben, ist EVU2 ein Beispiel dafür, wie ein EVU mehr Verantwortung bei der Softwaregestaltung übernimmt, ohne bestehende Strukturen zu verändern, sondern diese vielmehr ergänzt. Bei einem von EVU2 koordinierten großen IT-Projekt[14] schlagen sich die internen Strukturen in der Koordination des Ablaufs nieder, der auf Gremienarbeit basiert, wie es für eine zentralisierte Softwaregestaltung typisch ist: Es gibt einen dreiköpfigen Lenkungskreis und fünf Teil-Projektleitende nach Bereichen aufgeteilt, um eine bereichsübergreifende Kooperation horizontal hinzubekommen (u.a. die Bereiche Netzbau, Netz-Service, Netz-Management und kaufmännischer Service). An täglichen Jour fixes des Projekts, in denen Aktivitätenlisten durchgegangen werden und was noch an Arbeitspaketen offen ist, nehmen immer auch die verschiedenen Teilprojekte teil. Wie in Scrum wird iterativ vorgegangen. Doch reduziert sich das regelmäßige Feedback der Anwendenden auf Tests, mit denen sie entscheiden, ob sie einer Entwicklung grünes Licht geben. Wie für eine zentralisierte Softwaregestaltung typisch, waren es vor allem Fachleute und Führungskräfte, die an den Workshops für die Anforderungsaufnahme teilgenommen und sich mit den Programmierenden ausgetauscht haben.

Ein solcher zentraler, top-down-orientierter Softwaregestaltungsprozess rührt aber nicht allein aus der Organisation des EVU her. Das hängt davon ab, ob es sich um einen zentralisierten – im Fall von EVU2 fachbereichsübergreifenden – Softwaregestaltungsprozess in einer Hierarchie und mit mehreren Abteilungen handelt oder einen dezentralen. Bei einem anderen Projekt von EVU2, bei dem es um die Betreuung der Kundschaft im Netzbereich geht, laufen die Anpassungen dezentral über die betroffenen Teams. Treffen finden direkt zwischen Anwendenden und Programmierenden stattfinden. Die Hierarchie schlägt sich dann aber in einem bürokratischen Ablauf bei

14 Es ging um die Umsetzung von Redespatch 2.0, einer Regulierung für die Verteilnetzbetriebe (VNB).

der Umsetzung nieder: Der IT-Koordinator des Fachbereichs prüft und bewertet Anforderungen, die ihm die Anwendenden geben. Er übergibt die Anforderungen an den für den Fachbereich zuständigen technischen IT-Koordinator aus der IT-Abteilung, der dann die Umsetzung organisiert und dazu auf das IT-DL oder andere Programmierende bspw. Freelancer zugreift.

Stärker noch dezentral und auf Sammeln und Ausarbeiten von Anforderungen konzentriert ist die Softwaregestaltung im Fachbereich für Marktkommunikation von EVU2. Der Fachbereich nimmt individuelle Anpassungen an der Software direkt mit dem Softwareunternehmen vor. Die Anforderungen sammelt die Teamleitung mithilfe von Key User:innen bei den Anwendenden ein und gibt sie in Form von Tickets an die Softwarefirma weiter. Alle zwei Wochen hat die Teamleitung einen Termin mit einem Berater der Softwarefirma, in dem offene Punkte und Probleme besprochen werden. Doch auch hier muss die Teamleitung existierende Hierarchien berücksichtigen. Weil Änderungen an der Schnittstelle zur ERP-Software vorzunehmen sind, muss die IT-Abteilung die entsprechenden Programmierenden beauftragen. Die Teamleitung macht regelmäßig Termine mit den jeweiligen für das SAP-Softwarepaket verantwortlichen Prozessmanagenden, um Softwareänderungen abzustimmen.

Eine andere Art, wie ein EVU selbst Abläufe schafft, um individuell Software zu gestalten, zeigt EVU3. Ein neu geschaffener, zentraler Bereich für IT-Projekte treibt für mehrere Fachbereiche die Softwaregestaltung voran. So hat das EVU über kleinere IT-Projekte Erfahrungen in der App-Entwicklung gesammelt und dezentral in den Fachbereichen ausprobiert.

> »Das war so die erste Applikation, die in eine Cloud migriert ist von On-Prem[15], und das war so ein bisschen so ein Versuchsballon für alles Mögliche. Und da haben wir sehr viele Verfahrensweisen mittlerweile auch für andere Applikationen übernommen. Aus diesen Teams raus haben sich dann noch andere Teams gebildet, die dann zum Beispiel für den Kundenservice eine App entwickelt haben, wo man dann auch irgendwo so ein bisschen Selbstvertrauen gewonnen hat und man gesagt hat: Was wir hier gemacht haben, können wir dann eigentlich für einen anderen Bereich auch machen. […] Ob wir uns jetzt eine teure Lösung einkaufen oder hier was Kleines, Schnelles selber entwickeln. Lass uns mal die eigene Entwicklung probieren. Und das haben wir an vielen Ecken und Enden mittlerweile gemacht. Und funktioniert ganz gut. Funktioniert nicht alles, muss man dazu sagen. Aber viele dieser Projekte haben wirklich Mehrwert gebracht.« (Manager Digitalisierung)

Das IT-DL hat in dieser Fallstudie deutlich weniger koordinativen Aufwand als bei KOOP1. Es nimmt vor allem Anforderungen entgegen oder stellt Programmierende für IT-Projekte zur Verfügung und leitet nur noch wenige Projekte selbst. Die Umsetzung innerhalb des IT-DL ist zum Teil nach einem angepassten Scrum organisiert. Sonst herrschen laut einem Befragten im Alltag des IT-DL verschiedene Mischformen vor – aus Scrum, Wasserfall oder anderen Methoden.

15 Kurzform für On-Premises und vor der Zeit von Cloud-Computing das gängige Modell, um Software zu betreiben: auf einem lokalen Server mit der entsprechenden Lizenz.

Kommunikative Beziehungen: prekäre Beziehungen, direkte Kommunikation und hinderliche Hierarchien

Der Fall zeigt, dass mangelhafte kommunikative Beziehungen dazu führen können, dass kein zentralisierter Arbeitsprozess der Standardsoftwaregestaltung für mehrere EVU in einem IT-DL möglich ist. Die Marktbeziehung versucht das IT-DL nun durch Beziehungspflege zu verbessern, profitiert aber bereits vom direkten Kontakt zwischen Mitarbeitenden des IT-DL und den EVU. Jene EVU, die nun die Koordination der Softwaregestaltung übernehmen, merken, dass es eine Herausforderung darstellt, kommunikative Beziehungen innerhalb ihrer Matrixorganisation herzustellen.

Das IT-DL ist noch dabei, die Beziehungspflege zu den EVU zu verbessern. Ein befragter Teamleiter des IT-DL betont die Wichtigkeit der Beziehungspflege zu den EVU. Es sei Teil seiner Arbeit. Er musste seine Führungskraft aber erst überzeugen, dass durch eine gute Beziehung die EVU z. B. eher Fehler verzeihen würden. Dazu gehört es, öfters mal anzurufen, als direkter Ansprechpartner da zu sein und auch ehrlich und offen zu sagen, was man nicht kann, um dadurch Vertrauen zu schaffen.

Doch obwohl die kooperative Softwaregestaltung der EVU mit dem IT-DL als koordinierende Organisation nicht so gut funktioniert, ist ein direkter Austausch der Beschäftigten zwischen IT-DL und EVU möglich. Für das IT-DL ist es von Vorteil, wenn Ansprechpersonen und die internen Wege in den EVU bekannt sind. Weil das IT-DL durch Outsourcing aus den EVU entstanden ist, haben einzelne Mitarbeitende in den EVU noch alte Kontakte. Der befragte IT-Koordinator eines Fachbereichs von EVU2 spricht trotz eigentlich vorhandener und vorgegebener formaler Wege über die IT-Abteilung direkt mit dem IT-DL. Er betont die Vorteile direkter Kommunikation:

> »Nichtsdestotrotz halte ich es für klug, wenn der Anforderer direkt mit dem Entwickler spricht, wenn es eben um das Customizing geht oder halt das Thema an sich, dass das wirklich auch der Entwickler im O-Ton hört, was will der Anforderer eigentlich haben. […] Bevor ich das dann hier auf drei DINA4 Seiten runter beschrieben habe, das macht dann keinen Sinn, dann spricht man lieber.« (IT-Koordinator EVU2)

Da nun EVU2 selbst anfängt, Softwaregestaltung für fachbereichs- und organisationsübergreifende Projekte zu organisieren, muss es erst noch lernen, mit den Spannungen aufgrund von Hierarchien und Abteilungsgrenzen umzugehen, und eine gemeinsame Sprache finden. Der Projektleiter aus EVU2 beschreibt ausführlich, wie er initial zwischen den verschiedenen Bereichen horizontal interdisziplinäre Kontakte geknüpft hat. Er hat Termine gemacht, damit die verschiedenen im Projekt involvierten Fachbereiche die Perspektive der anderen kennengelernt haben. Das folgende Zitat zeigt, dass bestehende Strukturen erhalten und informelle Kontakte aus dem Projekt heraus bestehen bleiben, eine gemeinsame Wissensbasis entsteht und ein Austausch unabhängig von der Projektleitung stattfinden muss:

> »Ich persönlich bin hergegangen und habe die Kolleginnen und Kollegen quasi zusammengenommen und habe versucht, jeweils die Sichtweisen der anderen darzustellen. Das heißt, die kaufmännischen Kollegen haben was mitbekommen von den technischen Kollegen, was deren ihre Hauptaufgabe ist – auch andersherum. Und,

> dass jeder ein Gefühl bekommt für die Tätigkeit der anderen, dass man einen Blick hat, dass man versteht, was wird benötigt vom anderen und was kann der überhaupt liefern. Was ist der in der Lage…? Um erst mal ein Gefühl für den Gesamtprozess zu bekommen. Und entsprechende Anpassungen wurden dann auch gemeinsam diskutiert. Das hat anfangs ein bisschen länger gedauert, weil natürlich die Sichtweise nicht da war. Aber gerade jetzt, am Ende des Projektes, zeigt sich, dass das wirklich auch nicht die schlechteste Entscheidung war, Ressourcen dafür zu verwenden, weil dann auch im wirklichen Tagesgeschäft das Know-how vorhanden ist. Da brauchen wir es. Und die Kontakte sind geknüpft. Das heißt, nicht als Projektleiter die zentrale Funktion zu sein und zentrale Kommunikation, sondern die Kolleginnen und Kollegen sollten direkt miteinander auch kommunizieren, sich austauschen, ein Gefühl für die Arbeit der anderen bekommen.« (IT-Projektleiter EVU2)

Laut dem befragten IT-Projektleiter gibt es Kommunikationshürden durch das Hierarchiedenken z.B., wenn er die IT-Koordinatoren einbeziehen muss, diese aber keine fachbereichsübergreifenden Entscheidungen fällen können. Zudem würde der IT-Abteilung der Einblick in die tägliche Arbeit der Fachbereiche fehlen. Sie habe keine Vorstellung über die Probleme der Fachbereiche und aus seiner Sicht arbeitet sie nicht lösungsorientiert.

Auch aus Sicht des IT-Koordinators aus dem Fachbereich sind Hierarchien das Problem. Die operative Zusammenarbeit unter Mitarbeitenden unterschiedlicher Bereiche funktioniert dagegen gut:

> »Also, Ingenieure untereinander, die sind sich einig und zwar, da ist es völlig wurscht, ob die aus drei unterschiedlichen Bereichen kommen. Und es ist eben auch die Erfahrung, die wir bei uns gemacht haben und auch immer noch machen: Auf der operativen Basis funktioniert alles wunderbar, ja. Die Probleme fangen in den Führungsebenen an, weil da andere Befindlichkeiten ins Spiel kommen, die wir meistens auch gar nicht kennen.« (IT-Koordinator Fachbereich EVU2)

Bei den Fachbereichen setzt, wenn es um IT-Budget oder um zu erreichende Ziele geht, bei den Führungskräften ein Silodenken ein: Wer muss bezahlen? Wer bekommt die Anerkennung? Dies ist ein weiteres Beispiel dafür, wie innerhalb der EVU das Hierarchiedenken die kommunikativen Beziehungen behindert und bei Softwaregestaltung innerhalb von EVU der direkte Austausch nicht unbedingt einfacher ist.

Digitale Werkzeuge: Ticketsysteme, MS Excel und E-Mails

In dem Fall zeigt sich die typische Verwendung von digitalen Werkzeugen für die Softwaregestaltung. Allerdings ist sie weniger eindeutig auf die Typen von zentralisierten oder dezentralen Arbeitsprozessen zurückzuführen. Auch hier liegt es daran, dass, anders als bei KOOP1, die Softwaregestaltung wieder mehr die EVU übernehmen, auch was deren Koordination anbelangt.

Zwar hat das IT-DL ein Ticketsystem. Dies nutzen die EVU aber weniger, um die von ihnen koordinierte Softwaregestaltung zu organisieren, als vielmehr dem IT-DL Anforderungen zur Umsetzung zu übergeben. Wie bei KOOP1 können die EVU mithilfe dieses

Ticketsystems die Arbeit des IT-DL kontrollieren. Dazu gehört die Kontrolle vereinbarter SLA zwischen IT-DL und EVU oder die Tickets mit einer Priorität zu versehen.

Zusätzlich dazu verwendet EVU2 ein Ticketsystem der ERP-Software von SAP für die Umsetzung von Anforderungen. Für die Erarbeitung und den Austausch über Anforderungen setzt es MS Excel und E-Mails ein. Weil es die Koordination der Umsetzung der Softwaregestaltung selbst übernimmt, nutzt das EVU zudem noch die Ticketsysteme anderer Softwarefirmen und IT-DL, mit denen es zusammenarbeitet. So nutzt es die digitalen Werkzeuge sowohl für direkten, dezentralen Input als auch für Transparenz bei zentralisierter Softwaregestaltung. Es gibt nicht das eine Werkzeug, um die Kommunikation zentral zu kontrollieren.

Softwaretechnischer Zuschnitt: Synergien über gemeinsame Projekte und Release des IT-DL

So wie auch der Arbeitsprozess der Softwaregestaltung nicht klar zentralisiert ist, zeigt sich in dem Fall auch nicht der entsprechend typische softwaretechnische Zuschnitt eines Standards. Es zeigt sich ein Flickenteppich aus Standard von SAP, durch das IT-DL gestaltetem Standard, durch EVU gestaltetem Standard und individuellen Anpassungen und Erweiterungen. So reizen die EVU in diesem Fall die Möglichkeiten eines gemeinsamen IT-DL nicht aus, zusammen Synergien zu heben.

Der Release an Softwareänderungen des IT-DL für einen gemeinsamen Standard konzentriert sich vor allem auf einzelne Regulierungen, die sämtliche EVU umsetzen müssen. Doch auch dieser gemeinsame Release bzw. der durch das IT-DL für mehrere EVU entwickelte Softwareteil war für einige Jahre ausgesetzt. Ein anderer Weg zu Synergien sind gemeinsame Projekte, bei denen die EVU unabhängig vom IT-DL gemeinsam einen Standard gestalten, wie dies in dem oben geschilderten Projekt von EVU2 passiert ist.

Anders als bei KOOP1 gibt es keine abgestimmte Systematik zwischen den EVU, was sie als gemeinsamen Standard und was sie individuell gestalten. Für die Untersuchung wäre es daher ein nicht abzuschätzender Aufwand gewesen, genau herauszufinden, welches EVU was individuell gestaltet und wo es einem Standard folgt.

8.3.2.5. PAKET: zentrale Softwaregestaltung durch eine Softwarefirma

In dem Fall gestaltet eine Softwarefirma eine industriespezifische Standard-ERP-Software, bei der die EVU nur individuelle Einstellungen vornehmen können. Entsprechend existiert ein zentralisierter Arbeitsprozess der Softwaregestaltung der Softwarefirma mit den typischen Rollen und Abläufen zur Koordination, die Verhandlungen zwischen Softwarefirma und EVU über den Standard zulassen. Doch da innerhalb der Softwarefirma interdisziplinäre Teams existieren, ist sie nicht ausschließlich auf den Input der EVU angewiesen. Zudem ist allgemeines Branchenwissen wichtiger für die Gestaltung der Branchen-Standardsoftware als Wissen über spezifische Arbeitsabläufe in einem EVU. Deswegen spielen die Anwendenden bei der Softwaregestaltung nahezu keine Rolle. Jedoch: Da die EVU dezentral Einstellungen an der Software vornehmen können, gibt es intern in den EVU noch eine dezentrale Softwaregestaltung, die sich darum kümmert – mit den entsprechenden Rollen und Abläufen. In dem Fall spielen die organisations- und abteilungsübergreifenden kommunikativen Beziehungen eine geringere Rolle als

in den anderen Fällen. Dies ist zum einen so, weil die Softwarefirma (SF) spezialisiert auf die Softwareentwicklung ist und dort interdisziplinäre Teams existieren, die sich direkt austauschen können. Zum anderen ist zwar der Austausch zwischen EVU und SF wegen der Marktbeziehung stark formalisiert und die Spannungen aufgrund der unterschiedlichen Interessen von EVU und SF fallen auf. Allerdings ist die Kommunikation zwischen EVU und SF weniger wichtig für die Softwaregestaltung der ERP-Software, weil diese hauptsächlich von der Softwarefirma ausgeht. Nur mit einzelnen, ausgewählten EVU gibt es eine engere, kooperative Zusammenarbeit für die Softwaregestaltung. In dem Fall dient das Ticketsystem der Softwarefirma entsprechend vor allem dazu, die Kommunikation und die Umsetzung von Anforderungen und Fehlern (die den EVU in der Anwendung auffallen) zu kontrollieren. Dadurch, dass die Softwarefirma für viele EVU einen Standard entwickelt, schöpft sie die Möglichkeiten an Synergien in der Softwaregestaltung stark aus. Gleichzeitig gibt es durch die Einstellungsmöglichkeiten an der Software begrenzten Spielraum für individuelle Softwaregestaltung durch die EVU.

Rollen: verteilte Fachexpert:innen zwischen allgemeinem Branchen- und firmenspezifischen Anwendungswissen

Wie für einen zentralen Arbeitsprozess der Softwaregestaltung typisch, bestehen an die Rollen vor allem die Erwartungen, die Softwaregestaltung zu koordinieren und den Standard zu gestalten. Zusätzlich existieren innerhalb der EVU noch Rollen, um individuelle Einstellungen an der ERP-Software vorzunehmen.

Die Softwarefirma koordiniert die Softwaregestaltung: ob in ihren interdisziplinären Teams, durch Projekte mit EVU oder durch Arbeitskreise, zu denen Fachleute bzw. fachliche Repräsentant:innen aus den EVU eingeladen sind. Eine Rolle ist der Lösungsarchitekt. Er koordiniert Erweiterungen der Standardsoftware. Er prüft, was in der Branche an neuen Anforderungen entsteht, und entscheidet dann, was die Softwarefirma selbst umsetzt und was sie anderen Softwarefirmen überlässt. Zum Beispiel deckt die Softwarefirma beim Thema Abrechnung von Strom aus E-Auto-Ladesäulen mit ihrer Software nicht das Auslesen der Strommenge aus der Ladesäulen ab, weil es dafür Angebote anderer Unternehmen gibt. Daneben hat die Softwarefirma IT-Beratende. Sie nehmen Anforderungen von den EVU auf und Einstellungen an der Standardsoftware vor und unterstützen darüber hinaus die EVU bei der Anwendung des Standardprodukts.

> »Also, der reguläre Weg ist, wir haben den Consultant, der nimmt dann sozusagen unsere Anforderungen auf und dann geht das [Softwarefirma]-intern weiter.« (Teamleiterin Abrechnung EVU5)

Außerhalb der Softwarefirma bringen fachliche Repräsentant:innen der Branche sich in Arbeitskreisen bei der Gestaltung des Standards ein. So sind von den vielen EVU, welche die Software nutzen, nur wenige an der Gestaltung beteiligt.

In den EVU gibt es mal mehr und mal weniger interne Beschäftigte, die sich um einen optimalen Einsatz der Software kümmern können, dafür Einstellungen an dieser vornehmen und einen hohen Grad an Automatisierung garantieren. Wenn EVU solche internen Rollen haben, handelt es sich meist um Key User:innen oder Anwendungsbetreu-

ende. Key User:innen nehmen Anforderungen auf, nehmen Einstellungen an der Software vor, geben Tickets an die Softwarefirma auf oder informieren die Anwendenden über Änderungen an der Software.

> »Ja, also, die [Anwender] bringen sich ein. Das sind ja auch die, die immer am lebenden Objekt… Die wissen ja noch mehr, was sie brauchen, damit sie diesen Prozess bearbeiten können. […] Also, der Key User sammelt das alles ein.« (Teamleiter Abrechnung EVU5)

Anwendungsbetreuende haben meist noch eine übergreifende Perspektive auf das System und nehmen Prozessoptimierungen vor.

> »Weil es natürlich auch oft so ist, dass man schauen muss, wie man eben seine eigenen Prozesse in die neue Software implementiert und dann erst mal schauen muss, wie funktioniert es eigentlich? Was kommt da hinten raus?« (Anwendungsbetreuung EVU3)

In dem Fall zeigt sich die für Rollen in der Softwaregestaltung typische situative Ausgestaltung, wie die, dass es z.B. mal nur Key User:innen gibt und mal auch noch Anwendungsbetreuende. Zudem gibt es mehr Führungskräfte als in den anderen Fallstudien, welche die Software gestalten. Es gibt Team-/Gruppenleitende, die durch ihre langjährige Erfahrung mit der Software Systemwissen aufgebaut haben und wegen ihres fachlichen Know-hows eine vermittelnde Rolle zwischen Software und Anwendung spielen. Mal arbeiten sie am Schreiben von Anforderungen mit (Teamleiter EVU5), mal nehmen sie in geringem Umfang selbst Einstellungen an der Software vor (Gruppenleiter EVU4).

Ablauf: innerhalb und zwischen Organisationen – etablierte Kommunikationswege in der Softwarefirma und Einbindung von Branchenexpert:innen

Das Sammeln und Ausarbeiten von Anforderungen erfolgt in diesem Fall maßgeblich durch die Softwarefirma. Das heißt, zentral schreibt die Softwarefirma Konzepte und holt sich die Meinungen der EVU über verschiedene Wege ein, z.B. in gemeinsamen Treffen wie Arbeitskreisen, um über den Standard zu verhandeln. Nur in Ausnahmefällen schreiben die EVU selbst die Konzepte. Weil die Softwarefirma die Hoheit hat, was in den Standard kommt, muss der Ablauf selten tiefergehende inhaltliche Konflikte zwischen den EVU lösen. Weil es um einen Branchenstandard geht, sind die Gestaltungsnetzwerke vor allem auf Branchenexpert:innen begrenzt und schließen weniger individuelle Anwendende ein. Die Feedbacks zwischen EVU und SF reduzieren sich dann meistens auf Fehler und Tests und finden, was die Softwaregestaltung betrifft, mehr innerhalb der SF statt. Dafür setzt sie verschiedene Methoden zentral in der Softwarefirma (agile wie Scrum, Wasserfallmodell, Teamarbeit, Projekte) und zentralisiert für die Softwaregestaltung (Projekte, Arbeitskreise, Anwendendengruppen, Entwicklungskooperationen) ein.

In den (Fach-)Arbeitskreisen treffen sich mehrere EVU und einige Vertreter:innen der SF mindestens zweimal im Jahr, um über Details einzelner, dringender Themen zu sprechen. 10–15 Personen im Schnitt sprechen über aktuelle Umsetzungen und Anfor-

derungen. Die Mitglieder sind von der SF ausgesuchte Mitarbeitende aus EVU, die für ihre Expertise bekannt sind. Die Softwarefirma stellt erste Entwurfsfassungen von Konzepten vor und die EVU können eigene Vorschläge einbringen. Es werden Prozesse und Eingabemasken besprochen und ggf. gezeigt. Teilweise erstellen Mitarbeitende der EVU einzelne Kapitel der Konzepte.

> »Es gibt sogenannte Facharbeitskreise und da wirken wir immer mit und es gibt quasi gewisse… Es gibt dort die Fachabteilung Abrechnung. Es gibt den technischen Netzbetrieb, es gibt Marktkommunikation. Und dort sind jeweils Kollegen von uns, die dann natürlich aufmerksam auch die Gesetze lesen und die Veränderungen und geben dann der [Softwarefirma] Hinweise. Oder: Wir hätten das gerne so und so und könnt ihr das nicht so bauen?« (Gruppenleiter EVU3)

Die Herausforderung bei zentralisierter Softwaregestaltung, nicht nah an Anwendenden dran zu sein, löst PAKET durch Prototyping, z.B. indem es laut Befragten ca. zwei von drei Anforderungen agil umsetzt, d.h. erst ein Grundgerüst programmiert und dann auf Basis von Feedback durch die EVU kontinuierlich weiterentwickelt. In anderen Fällen stellt die Softwarefirma umgesetzte Features in einer Web-Konferenz ausgewählten Anwendenden vor, um Feedback zu erhalten.

Dabei kann der interdisziplinäre Austausch innerhalb der Softwarefirma sehr intensiv und kontinuierlich erfolgen, was ein Vorteil der Zentralisierung ist:

> »Es wird meistens ein zumindest grobes fachliches Konzept vorgegeben und alles weitere, die konkrete fachliche Ausprägung und so, das wird dann mit den entsprechenden Spezialisten für die Prozesse dann geklärt und dokumentiert.« (Programmierer)

In den Teams der Softwarefirma, die jeweils für einen Teil der ERP-Software zuständig sind, gibt es tägliche Besprechungen wegen neuer Anforderungen vor allem wegen neuer Regulierung (typischerweise ohne Anwendende oder EVU). Im Team des befragten Programmierers gibt es drei, vier Jour fixes die Woche und dann zwei, drei abhängig von Ereignissen, Problemen etc., über die man reden muss.

Die Ausnahme stellt EVU5 dar, das erst vor kurzem die ERP-Software eingeführt hat. Weil das EVU einige Eigenheiten hat, die bisher die Standardsoftware nicht berücksichtigt, gibt es eine engere Zusammenarbeit – eine Art Entwicklungskooperation. Es gibt regelmäßigen Austausch zwischen dem EVU und der Softwarefirma. Die befragte Gruppenleitung aus EVU5 hat zusammen mit Key User:innen einmal in der Woche einen Termin mit der Softwarefirma und deren Programmierenden.

Neben der Teilnahme an der Konzeptionierung für den Branchenstandard leisten die EVU eine wichtige Qualitätskontrolle/-verbesserung, indem sie Tests durchführen und Fehlermeldungen zu neuen Versionen der Software aufnehmen. Wie auch bei der Konzeptionierung machen die EVU dabei in unterschiedlichem Ausmaß mit (tendenziell eher die größeren EVU).

Kommunikative Beziehungen: zwischen reiner Zulieferbeziehung, Formalisierung und partnerschaftlicher, direkter Kommunikation

Die für einen zentralen Arbeitsprozess der Softwaregestaltungen typischen spannungsgeladenen und bürokratischen Beziehungen existieren in dem Fall. Wobei einige der EVU gut mit der reinen Lieferbeziehung leben können. Andere wünschen sich einen partnerschaftlichen Umgang, um das komplizierte Produkt besser einsetzen zu können. So oder so geht es in diesem Fall zwischen EVU und Softwarefirma weniger um die Gestaltung der Software als um die Qualität eines Standardprodukts und welche Aufgaben die Softwarefirma als Zulieferunternehmen darüber hinaus hat. Nur wenige EVU arbeiten bei der Gestaltung der Software enger mit der Softwarefirma zusammen.

Wie für einen zentralisierten Arbeitsprozess der Softwaregestaltung typisch, wollen die EVU in erster Linie eine funktionierende Standardsoftware. Deshalb existieren zwischen Softwarefirma und EVU nicht immer kommunikative Beziehungen. Ein Befragter spricht von »reine[r] Dienstleistungsbeziehung« (Teamleiter EVU2). Für einen anderen Befragten ist die Beziehung erst einmal nicht wichtig, weil die Software einfach funktionieren soll. Er erwartete, dass die Softwarefirma Fehler oder Anforderungen termingerecht behebt bzw. umsetzt. Teilweise kommt es zu Eskalationen aufgrund von zu langsamer Umsetzung. Der Teamleiter aus EVU2 ist misstrauisch und hat den Verdacht, dass größere EVU schneller Rückmeldung bekommen. Er ist davon genervt, dass er immer nachfragen muss, wenn es länger dauert.

Doch ist die industriespezifische ERP-Software ein zu kompliziertes Produkt, als dass immer eine bloße Auslieferung ausreicht. Einige EVU haben deshalb die Erwartung, dass die Beziehung darüber hinausgeht. Für einige befragte EVU hat die Softwarefirma, obwohl nicht vertraglich fixiert oder Teil des Produkts, eine Bringschuld. Sie kommt nicht auf die EVU zu, um auf Verbesserungsmöglichkeiten, Einstellungsmöglichkeiten oder (neue) Funktionalitäten hinzuweisen. Ein Befragter merkt manchmal nur durch Austausch mit anderen EVU, was noch mit der Software möglich wäre. Ein EVU hat erst durch Umstieg auf ein neues Lizenzmodell gemerkt, was es noch alles für Module gibt. Aus der Sicht eines anderen Befragten wäre es partnerschaftlich, wenn mehr Austausch als dringend notwendig stattfinden würde. In Einzelfällen gibt es das und dann ist es ein Geben und Nehmen: Laut Befragten existieren bei Tickets, die schwer zu beschreiben sind und deshalb ausführlicher ausfallen, engagierte Mitarbeitende der Softwarefirma. Diese rufen an und geben Tipps und notwendige Hintergrundinformationen, was das Problem sein könnte und wie es zu lösen ist.

Das Gegenteil zur reinen Zulieferbeziehung sind die engeren Beziehungen der Softwarefirma mit EVU, mit denen sie gemeinsam Innovationsprojekte macht oder langfristig bei der Gestaltung des Standards zusammenarbeitet. Mit diesen EVU ist die Zusammenarbeit »partnerschaftlich« (Lösungsarchitekt Softwarefirma).

In diesem Fall verwenden viele EVU und ihre unzähligen Anwendenden eine Standardsoftware und es steht ihnen nur das Ticketsystem und andere formale Wege der Kommunikation mit der Softwarefirma offen. Weil die Softwarefirma eine große Kundschaft hat, ist eine stärker formalisierte Kommunikation notwendig. Dies hat Nachteile und Befragte ziehen die direkte Kommunikation vor:

> »Also, wenn man einen direkten Draht hat, dann ist es gut. Dann läuft es. Aber die Kommunikation rein über das Ticketsystem ist… Meiner Meinung nach verhindert das eine schnelle Lösung.« (Gruppenleiter Abrechnung EVU5)
> »Da ist es in der Kommunikation meistens wesentlich einfacher, wenn man direkt mit dem Kunden dann spricht und sagt: ›Hier, wie siehts aus?‹ Mit dem Bearbeiter halt wirklich, weniger dann mit den Entscheidungsebenen, sondern mit dem Benutzer wirklich der Software.« (Programmierer)

Unabhängig, ob Ticketsystem, E-Mail oder Telefon: Einige befragte Personen betonen auch bei der zentralisierten Softwaregestaltung den Vorteil, wenn es langfristige Ansprechpersonen und Netzwerke in der Softwarefirma gibt und man die gleiche Sprache spricht. Im EVU5 gibt es einen Mitarbeiter, der Programmierer in der Softwarefirma war und nun Berater ist und der beide Seiten versteht: Er kann die Anforderungen und Probleme der Anwendenden verstehen und weiß zugleich, was mit der Software möglich ist. Eine Befragte hat die Erfahrung gemacht, dass eine höfliche, sehr klar formulierte und gradlinige Kommunikation hilft, schnellere Antworten zu bekommen. Kolleg:innen, die das nicht machen, sind schon »auf die Nase gefallen« (Anwendungsbetreuerin EVU3).

Digitale Werkzeuge: Ticketsystem – digitale Vernetzung und Detaillierung

In dem Fall setzt die Softwarefirma digitale Werkzeuge für eine zentralisierte Softwaregestaltung auf typische Weise ein: um die Softwaregestaltung zu koordinieren und zu kontrollieren und weniger, damit Anwendende direkt Anforderungen aufnehmen können. Es geht darum, die Kommunikation und den Fluss an Fehlern und Anforderungen der umfangreichen Kundschaft der Softwarefirma zu kontrollieren. Zwei Befragte meinten, dass 90 % der Zusammenarbeit mit der Softwarefirma über Tickets läuft. Der Rest sind E-Mails an konkrete Ansprechpersonen. Ein anderer Interviewter meint, dass alles über Tickets läuft. EVU3 gibt ca. 400 Tickets im Jahr auf. Für die Sortierung der Tickets können die EVU eine Priorisierung hinterlegen und haben drei Typen zur Auswahl: Anforderungen, Fehler oder Unterstützung. In der Softwarefirma gibt es Ticketmanagende, deren Aufgabe es ist, ein Auge auf sämtliche Tickets zu haben, und wenn z.B. etwas eskaliert, leiten sie es an das mittlere Management oder eine Teamleitung weiter.

Softwaretechnischer Zuschnitt: zentrale Standardisierung und Priorisierung durch die Softwarefirma

Der für einen zentralen Arbeitsprozess der Softwaregestaltung typische softwaretechnische Zuschnitt für einen Standard zeigt sich in diesem Fall sehr klar. Dabei ist die Konsensfindung über den Standard in der Softwarefirma zentralisiert. Sie hat letztendlich die Entscheidungshoheit. Größere EVU haben mehr Einfluss u.a. durch die Teilnahme an Arbeitskreisen. Bei kleineren EVU geht es vor allem um eine günstige Software. Sie folgen dem Standard, weil das für sie billiger ist.

> »Bei kleineren ist es wirklich ein reines Kostending. […] Bei größeren Werken wollen auch zwar die Kosten senken, möchte aber gerne immer ganz viel individuelles Zeugs beibehalten.« (Lösungsarchitekt Softwarefirma)

Laut drei Befragten haben große EVU mehr Einfluss auf die Software. Dies liegt daran, dass ihr Kundenstamm größer ist und ein Fehler in der Software dementsprechend kostspielig und folgenschwer sein kann. Zwischen Kosten und Anpassungen muss jedes EVU für sich eine Lösung finden.

> »[ERP-Software] ist ein Standard und das Stadtwerk [Name] oder was auch immer, hat aber kein Interesse, weil der hat fünf Fehler davon und wir haben vielleicht 5000 Fehler davon. Dann haben wir natürlich ein viel höheres Interesse daran, dass es […] die Software das vollumfänglich abarbeitet.« (Gruppenleiter EVU3)

Es ist noch einfacher als bei KOOP1, im zentralisierten Softwaregestaltungsprozess regulatorisch notwendige Umsetzungen oder solche, die sämtliche EVU betreffen, in den Standard aufzunehmen. Wenn einzelne EVU sich nicht dem Standard unterordnen, dann vor allem bei wettbewerbsdifferenzierenden Prozessen, z.B. wenn EVU Anschreiben oder Rechnungen an ihre Kundschaft senden:

> »Ich habe schon das Gefühl, wir werden so ein bisschen in die Standardlösung gepresst. Wir haben immer noch sehr individualisierte Schreiben und so weiter. So viele Sachen, die sind für uns grundsätzlich unumstößlich. Wir werden jetzt keine Standardschreiben rausschicken wie 90 Prozent der [Software]-Anwender. Die müssen schon individualisierbar sein.« (Anwendungsbetreuung EVU3)

8.3.2.6. KOOP3: zentrale Softwaregestaltung durch Softwarefirma in Ko-Produktion mit IT-DL

In dem Fall nutzen die beteiligten Organisationen beide Möglichkeiten der Softwaregestaltung: Die Softwarefirma gestaltet eine Standard-IoT-Software und das IT-DL und einige EVU erweitern diese individuell. Das heißt, für den Kern der Software existiert ein zentraler Arbeitsprozess der Softwaregestaltung. Das Besondere ist aber nun im Unterschied zu den vorhergehenden Fällen, dass Hierarchien und Marktbeziehungen, auch wenn sie zwischen und innerhalb der Organisationen vorhanden sind, kein Hindernis für die Softwaregestaltung darstellen. Denn die kommunikativen Beziehungen zwischen IT-DL und SF sind sehr kooperativ, direkt und durch eine interpersonale Beziehungsfähigkeit gekennzeichnet, wodurch die Mitarbeitenden Konflikte informell lösen. Zusätzlich dazu sind nur wenige an der Softwaregestaltung beteiligt und die interdisziplinären Wissensgrenzen sind gering, wodurch sich letztendlich Anforderungen unkompliziert aufnehmen, ausarbeiten und umsetzen lassen. So kombiniert der Fall in den Rollen und Abläufen Eigenschaften einer zentralisierten und dezentralen Softwaregestaltung. Letztendlich kann von einer Ko-Produktion gesprochen werden, weil IT-DL und Softwarefirma beide ein Interesse daran haben, dass das IoT-Kernmodul der Softwarefirma weiterentwickelt wird, alle ihren Teil kooperativ beitragen und das IT-DL eigene Module an das Kernmodell andockt. Anders als bei PAKET existiert zwischen IT-DL und Softwarefirma keine reine Lieferbeziehung. Was in den Standard des IoT-Kernmoduls kommt, darüber entscheidet aber letztendlich die Softwarefirma.

Rollen: Projektmanager, Account Manager, Product Owner

Das Besondere an dem Fall ist, dass zwar die für einen zentralisierten Arbeitsprozess einer Standardsoftware typischeren koordinierenden Rollen existieren. Diese verteilen sich aber auf Softwarefirma, IT-DL und einzelne EVU. Anders als für eine zentralisierte Softwaregestaltung typisch, sind die Rollen weniger spezialisiert, weil die Implementierungsprojekte, welche die Hauptquelle von neuen Anforderungen sind, nur wenige Beteiligte haben. So mischen sich in den Rollen die Aufgaben des Koordinierens, der Aufnahme und der Ausarbeitung von Anforderungen.

Die besondere Aufgabe des IT-DL ist es in diesem Fall, in die Energiewirtschaft zu vermitteln. Er hat Rollen, die zwischen EVU und Softwarefirma koordinieren und zugleich Anforderungen sammeln und an die Softwarefirma übergeben, damit diese in die Standard-IoT-Software einfließen. Der Product Owner des IT-DL beschäftigt sich unabhängig von einzelnen IoT-Implementierungsprojekten kontinuierlich mit der Weiterentwicklung der IoT-Standard-Datenplattform. Darüber hinaus betreut er die vom IT-DL selbst entwickelten individuellen Module.

Einzelne EVU setzen sich intensiver mit dem Thema IoT auseinander und schaffen interne Positionen dafür. Ein befragtes EVU hat einen eigenen Product Owner für IoT etabliert. Er spricht sich intern mit Projektleitenden ab, welche die Implementierungsprojekte durchführen. Er ist Ansprechpartner für die eigene Kundschaft (das EVU betreut andere EVU).

In der Softwarefirma selbst gibt es Rollen, welche die Zusammenarbeit mit dem IT-DL und den EVU koordinieren, wobei sie auch Anforderungen sammeln und ausarbeiten. Wenn die Softwarefirma selbst Kontakt zu den EVU hat, dann vor allem über Projektmanagende. Das ist aber selten, weil die Zusammenarbeit mit EVU meist über das IT-DL läuft. Sie nehmen Anforderungen auf und geben sie an die Programmierenden weiter. Der Account Manager der Softwarefirma ist für den ersten Kontakt mit interessierten Unternehmen zuständig, bevor es zur technischen Umsetzung kommt. Er und das Projektmanagement haben, wie es für eine Standardlösung typisch ist, auch noch die Aufgabe, die Kundschaft zu beraten und ihre Erwartungen mit den Möglichkeiten der Software in Einklang zu bringen. Es geht darum, die Kundschaft darauf einzustellen, welche Gestaltungsmöglichkeiten es gibt und wann eine Anforderung Teil des Standards wird.

Ablauf: Anforderungstreffen, Projekte, Scrum

Auch wenn es sich in dem Fall um eine zentralisierte Softwaregestaltung handelt, Softwareanwendung, -gestaltung und Programmierung nicht in einer Organisation stattfinden und unterschiedliche Treffen Anforderungen für den Standard zum Thema haben: Sammeln und Ausarbeiten bilden jeweils gleichwertige Schwerpunkte des Ablaufs, weil aufgrund der wenigen Beteiligten die Koordination weniger kompliziert als in anderen Fällen ist. Zudem sind in diesem Fall trotz der zentralen Softwaregestaltenden Verhandlungen und Konfliktlösungen kein wesentlicher Bestandteil des Ablaufs. Er ist unbürokratisch.

Um zentral über energiewirtschaftliche Anforderungen zu sprechen, gibt es monatliche Treffen zwischen dem IT-DL und der IoT-Softwarefirma. Dort übergibt das IT-DL

Anforderungen an die Softwarefirma und beide diskutieren über sie. Die Anforderungen stammen aus IT-Projekten mit EVU und den dabei stattfindenden (wöchentlichen) IoT Jour fixes. Dort haben IT-DL und EVU Anforderungen gesammelt und über sie diskutiert, um sie dann alle ein, zwei Monate in dem Termin mit der Softwarefirma zu besprechen.

Auch die IoT-Softwarefirma macht Implementierungsprojekte, in denen sie Anforderungen sammelt. Doch betreffen solche Projekte meist Kundschaft außerhalb der Energiewirtschaft.

Intern nutzt die Softwarefirma für die Umsetzung und Ausarbeitung der Anforderungen Scrum. Diese iterative Entwicklungsmethode bietet sich an, weil auch die Anforderungen iterativ entstehen: peu à peu durch die Implementierungsprojekte oder neuen Ideen der Softwarefirma, des IT-DL oder der EVU.

Kommunikative Beziehungen: Reziprozität und persönliche Beziehungspflege

Die für einen zentralen Arbeitsprozess der Softwaregestaltungen typischen bürokratischen und spannungsgeladenen Beziehungen gibt es nicht. Das liegt daran, weil die Marktbeziehungen zwischen Softwarefirma, IT-DL und EVU und die Hierarchien innerhalb der Organisationen die Kommunikation nicht behindern. Es gibt direkte Kommunikation unabhängig von Hierarchien oder Verträgen, basierend auf Reziprozität, gemeinsamem Mehrwert bzw. gemeinsamer Ko-Produktion, längerfristigen Beziehungen und einer gemeinsamen Wissensbasis.

Das IT-DL kann durch längerfristige Zusammenarbeit und eigenes Know-how zu IoT der Softwarefirma genau sagen, was zu tun ist und wo das Problem liegt. Es hat sich zwischen beiden bereits eine gemeinsame Sprache etabliert. In diesem Zusammenhang kommt dem IT-DL das umfassende Know-how zugute, das es im Laufe der Zeit zur IoT-Software und deren Anwendung in der Energiewirtschaft erworben hat. Wenn Anforderungen direkt von den EVU kommen, ist das nicht immer der Fall. Dann ist es aufwendiger, zu einem gemeinsamen Verständnis zu kommen, damit letztendlich das umgesetzt wird, was das EVU erwartet und aus Sicht der Softwarefirma möglich ist.

Konflikte sind kein Hemmnis für die Kommunikation, z.B. wenn das befragte EVU1 andere Erwartungen an einen Umsetzungszeitraum hat. IT-DL und EVU1 klären Konflikte informell z.B. via Telefon. Es findet ein Erwartungsabgleich statt. Die unerfüllten Erwartungen und die divergierenden Vorstellungen werden angesprochen, was zur Konfliktlösung beiträgt.

> »Ja, also Konflikte gab es auch mal. Ich sage mal, es passiert glaube ich relativ automatisch mal. […] Aber ich glaube… Kein Konflikt der schriftlich erfolgt, sagen wir es mal so. Ganz normal am Telefon oder persönlich. Man sagt, dass man das Thema unprofessionell findet und eine andere Erwartungshaltung hat. So lief es bei uns auf jeden Fall bis jetzt. […] Aber wir machen es immer rein auf der persönlichen Ebene.« (Teamleiter IT EVU1)

Die Zusammenarbeit der Firmen ist geprägt durch Vertrauen und die Reziprozität der Ko-Produktion. Der direkt erlebte und vorführbare Mehrwert in Form der Software, d.h. die Daten aus den Sensoren in die bestehende IT-Landschaft der EVU zu integrieren,

ist für die Beteiligten grundlegend für eine erfolgreiche Zusammenarbeit. Sowohl beim IT-DL als auch beim befragten EVU1 ist es möglich, kleinere Piloten für IoT ohne größeren bürokratischen Aufwand durchzuführen. Stellenweise werden Projekte ganz ohne Verträge gemacht, wie EVU1 berichtet:

> »Also das sind so, ich will nicht sagen, bisschen Pilotcharakter oder, ich will nicht sagen, wie kann man das sagen: Einstieg zu einem ›proof of concept‹, zu einem neuen Produkt, der aber noch keine vertragliche Relevanz hat. Also vertraglich irgendwie zugesichert ist. Sondern beide geben einen gewissen Aufwand rein, ohne ihn vorher zu klassifizieren oder zu monetarisieren.
> I: Aber es gibt Verträge bei solchen Projekten oder ist das komplett ohne Verträge?
> B: Sehr oft ohne Vertrag.« (Teamleiter IT EVU1)

Das IT-DL hat den Vorteil gegenüber der Softwarefirma, dass es bereits länger in der Energiewirtschaft aktiv ist und es die EVU kennen. Manche EVU sind Gesellschaftende vom IT-DL. Das heißt, nicht nur die Beziehung zwischen IT-DL und Softwarefirma ist eng, sondern auch jene zwischen IT-DL und EVU. Weder Markt noch Hierarchie prägen allein die Zusammenarbeit. Verträge, partnerschaftliche Beziehungen und Projektkoordination vermengen sich.

> »Wenn du den Kunden seit 20 Jahren kennst, dann sagst du: ›Du, pass auf, ich habe das intern [im EVU] geklärt. Wir machen das jetzt, wir können das entwickeln. Aber ich brauche noch drei Wochen, bis das durch den Einkauf geht.‹ Weil solche Prozesse halt mal sehr langsam sein können. Dann sagen wir: ›Ja, okay, wir kennen uns seit 20 Jahren, wir gehen in die Entwicklung‹.« (Product Owner IT-DL)

Digitale Werkzeuge: Ticketsystem, Test-Accounts

In dem Fall zeigt sich in dem EVU die für digitale Werkzeuge typische Verwendung in zentralen Arbeitsprozessen der Softwaregestaltung in abgeschwächter Form: Denn es gibt zwar ein Ticketsystem der IoT-Softwarefirma, das vor allem dazu dient, die Anforderungen oder Fehler der diversen Kundschaft kanalisieren zu können. Ca. 100 Tickets bearbeitet die Softwarefirma im Monat (Stand 2021). Doch wurde 2021 probeweise dem IT-DL Zugriff auf das Ticketsystem gegeben. Er kann dadurch Tickets für die Softwarefirma nicht nur anlegen, sondern auch bearbeiten. Das sollen in Zukunft auch die EVU können. Das hat den Vorteil, dass das IT-DL Anforderungen sieht und direkt mit dem EVU reden kann, dass das jeweilige Ticket aufgenommen hat. So unterstützt das Ticketsystem die Ko-Produktion, bei der zentral die Softwarefirma den Standard und IT-DL und EVU dezentral individuelle Module programmieren. Sonst nutzt das IT-DL für die Übergabe von Anforderungen aus den EVU an die Softwarefirma pragmatisch E-Mails, Excel-Dateien oder andere Softwarelösungen. Wie für dezentrale Softwaregestaltung typisch, dienen sie vor allem dazu, dezentral Anforderungen direkt aufzunehmen und zu dokumentieren.

Softwaretechnischer Zuschnitt: zentral durch IoT-Softwarefirma

In dem Fall zeigt sich der für einen zentralisierten Softwaregestaltungsprozess klare Standard-Zuschnitt. Wobei zu diesem Schnittstellen gehören, so dass IT-DL oder EVU dezentral individuelle Erweiterungen anlegen können.

Grundsätzlich legt die IoT-Softwarefirma die Priorisierung fest und teilt mit, bis wann sie was umsetzt. Sie entscheidet darüber, was Teil des Standards wird. Nur bei etwas Grundlegendem oder wenn mehrere EVU des IT-DL etwas wollen (laut einem Befragten so ab drei bis vier EVU), konzeptionieren und priorisieren IT-DL und Softwarefirma gemeinsam.

8.3.2.7. STARTUP: dezentral in Kreisen – eklektische Mischung aus Holokratie und Scrum

In dem Fall liegt ein dezentraler Arbeitsprozess der Softwaregestaltung für eine individuelle Software innerhalb von STARTUP vor. Von Anfang an hat STARTUP darauf gesetzt, die Softwaregestaltung selbst in die Hand zu nehmen. Der Fall ist ein Beispiel für den Primat der Softwareentwicklung einer Organisation. Deshalb gibt es innerhalb der Organisation für den Arbeitsprozess keine Hürden durch Hierarchien oder Abteilungsgrenzen der Softwareanwendung. Es existiert vielmehr eine rollenbasierte Organisation, die stark darauf konzentriert ist, in einer Mischung aus Scrum und Holokratie Anforderungen auszuarbeiten und umzusetzen. Wie für einen dezentralen Arbeitsprozess der Softwaregestaltung typisch, sind die kommunikativen Beziehungen offen, direkt, flexibel und werden durch ein gemeinsames Ziel gestützt. Neben einem Ticketsystem nutzt STARTUP Chat-Kanäle, die den dezentralen Input von Gestaltenden, Programmierenden und Anwendenden für Anforderungen ohne bürokratische Hindernisse erlauben. Die individuell für die eigene Organisation gestaltete Software bietet STARTUP anderen Organisationen wie Automobilfirmen als Standard-White-Label-Lösung an.

Rollen: Product Owner, Gründer, Solution Architect

Wie für einen dezentralen Arbeitsprozess einer individuellen Softwaregestaltung typisch, bestehen an die Rollen vor allem die Erwartungen, Anforderungen aufzunehmen und auszuarbeiten. So ist die befragte Product Ownerin sehr wenig mit Koordination beschäftigt. Sie prüft die rechtlichen Voraussetzungen, z.B. wer bei Firmenwägen die monetäre Pauschale bekommt, stimmt sich mit Behörden ab, entwickelt Roadmaps für die Produktentwicklung und lässt Feedback des eigenen Support-Teams der mobilen Anmeldungsapp in die Software einfließen. Zudem optimiert sie die Anmeldungsprozesse und beantwortet besonders schwierige Support-Anfragen. Ganz ohne Koordinierung geht es aber nicht.

> »Ich muss alles im Blick haben und bin die Brücke zwischen Produkt-, Tech-Team und Kommunikation. Ich koordiniere und kommuniziere intern (Meetings, Tickets, Pläne, Dokumentation…).« (Product Ownerin Anmeldung E-Autos)

Einer der Gründer macht als Geschäftsführer bei der Softwaregestaltung mit. Er hat Prozesswissen zum Emissionshandel, übernimmt Aufgaben im Vertrieb, im Business Development und ist auch an der Produktentwicklung beteiligt.

»[Der] Geschäftsführer so Wirtschaftsinformatiker ist, der beide Seiten versteht. Und das ist sehr, sehr fruchtbar, weil er versteht beide Sprachen: einmal die Wirtschaftler und einmal die Techies. Und dann ist er ein sehr guter Mediator und setzt dann auch angemessene Prioritäten und Deadlines und so in Absprache auch mit uns.« (Programmierer2)

Auch in dem Bereich, der sich um die Anmelde-Software kümmert, ist er als Moderator tätig und wacht darüber, dass die Termine für die Teamtreffen stattfinden. Er agiert damit ähnlich wie ein Scrum Master. Neben diesen Kernrollen, die Anforderungen für die Programmierenden schreiben, haben in der rollenbasierten Organisation auch noch andere als Teil ihrer Rolle, bei Bedarf an Konzepten mitzuschreiben: z.B. die Programmierenden selbst, der Solution Architect oder Personen aus dem Kommunikationsbereich von STARTUP.

Der Fall zeigt, wie andere Fälle auch, dass es den Arbeitsprozess der Softwaregestaltung auszeichnet, dass Beteiligte je nach Konstellation auch mal mehrere Rollen ausfüllen – situativ angepasst und wechselnd. Wobei es hier so weit geht, dass alle gemeinsam regelmäßig in dafür vorgesehenen Treffen darüber verhandeln, welche Rollen jemand zusätzlich zu seiner Stammrolle übernimmt oder ob er gar seine Stammrolle wechselt.

Ablauf: Kreise, Scrum

Der Ablauf besteht primär, wie für einen dezentralen Arbeitsprozess der Softwaregestaltung typisch, im Ausarbeiten und Sammeln von Anforderungen. Die Organisation ist von Anfang an auf die Softwaregestaltung ausgerichtet. Es besteht ein enger und pragmatischer Austausch. Es stören keine Hierarchien oder althergebrachte, auf die Softwareanwendung ausgerichtete Organisationsstrukturen dabei, die Möglichkeiten der Softwaregestaltung auszuschöpfen – anders als bei einer Softwaregestaltung innerhalb von EVU.

Der Ablauf der Konzeptionierung zeichnet sich durch direkte Feedbacks und den direkten Austausch zwischen Anwendung, Gestaltung und Programmierung aus: Erst trifft man sich in einem Kreis, dann werden die Anforderungen in Scrum-Arbeitsweise abgearbeitet. Die offenen Anforderungen stehen im Ticketsystem Jira: im Backlog jene, die priorisiert sind, und auf dem Kanban-Board[16] jene, die aktuell die Programmierenden bearbeiten. An den Kreisen nehmen Product Owner:innen, Geschäftsführer und Programmierende teil, um Aufgaben zu planen und Anforderungen für die Softwaregestaltung zu erarbeiten.

Der Produkt-Kreis für die Anmeldesoftware findet wöchentlich statt. Die Product Ownerin lädt dazu Programmierende, bei Bedarf auch jemanden aus der Kommunikationsabteilung und, wenn auch selten, Kundschaft ein. Mit einem Kunden (einem Automobilkonzern, der die App zur E-Auto-Anmeldung nutzt) hat STARTUP eng zusammengearbeitet, um dessen Bedürfnisse bei der Entwicklung zu berücksichtigen. Der Kreis

16 Darunter ist erst einmal nur die Visualisierung der Arbeitsschritte in eine Software gemeint und nicht die Kanban-Methode. Meist ist die Darstellung ganz einfach und unterteilt die Aufgaben in solche, die noch offen sind, gerade bearbeitet werden und abgeschlossen sind.

bespricht Konzepte und nimmt Ideen aus dem Support auf, worüber auch Anforderungen der Anwendenden einfließen. Den Backlog für die Programmierenden pflegt im Bereich der Anmelde-Software »hauptsächlich« (Product Ownerin) die Product Ownerin.

Scrum lebt STARTUP im Bereich der App für die Anmeldung von E-Autos in folgender Version, was die enge und unkomplizierte Zusammenarbeit zwischen Gestaltenden und Programmierenden verdeutlicht:

> »Wir machen meistens einen Termin, in dem wir besprechen, was die Vorstellungen sind und wie das dann technisch umsetzbar ist. Das halte ich fest und wir machen erste Aufgabenstellungen daraus. Das wird dann bei den Scrum-Meetings gemeinsam nochmal durchgegangen. Für [Teil des Start-ups] nehme ich dann Ideen und Entwicklungen ab. In dem Scrum [Entwicklerteam eines Teils des Start-ups] sind 5 Personen, aber niemand in Vollzeit. Wir machen keine täglichen Stand-ups, sondern nur wöchentlich. Ansonsten machen wir Planning und Review inkl. Retro. Hinzu kommen dann bei größeren Aufgaben auch das Epic-Planning. Es gibt feste Sprints.« (Product Owner)

Auch diese Fallstudie zeigt, wie die Firmen für die Softwaregestaltung Methoden wie Scrum anpassen. STARTUP setzt kein reines Scrum um. Zum Beispiel gibt es keine täglichen Treffen, die sogenannten *Daylies*. Größere Themen teilen die Beteiligten in mehrere Aufgaben auf und deren Bearbeitung ist dann wieder in der Rollen- und Kreis-Struktur organisiert.

Was STARTUP auszeichnet und typisch für dezentrale Softwaregestaltung ist: Neben Product Ownern als übliche Anforderungsmanagende kann jede Person informell Anforderungen an die Programmierenden stellen:

> »Bei uns läuft da bei weitem nicht alles unbedingt die definierten Wege, sondern oft viel pragmatischer. Wenn dann jemand von der Kommunikation direkt zu mir kommt und sagt: ›Hey, ich hatte folgende spannende Idee und was hältst du davon?‹« (Programmierer1)

Feedback ist zum einen nach dem zweiwöchigen Sprint dadurch möglich, dass die Programmierenden »eigentlich immer« (Product Owner) Prototypen vorstellen. Zum anderen gibt es, wie in der Softwareentwicklung üblich, Tests. Die Tests laufen, typisch für diesen Fall, unbürokratisch ab – »wie es grade passt« (Product Ownerin).

Für das Software-Modul zum Handel mit Emissionszertifikaten treffen sich die Beteiligten dreimal wöchentlich. Dort ist zusätzlich noch ein Solution Architect im Kreis mit dabei. Zum Zeitpunkt der Befragung wurde der Sprint-Zyklus ausgesetzt, weil es einige dringende Themen abzuarbeiten galt, was die Flexibilität des dezentralen Ablaufs verdeutlicht.

Kommunikative Beziehungen: flexibler, direkter und offener Austausch

In dem Fall existieren die für einen dezentralen Arbeitsprozess der Softwaregestaltungen typischen direkten und langfristigen Beziehungen und eine gemeinsame Sprache. Die Befragten betonen den Vorteil direkter Kommunikation und wie offen und unkompliziert die Beziehungen die Kommunikation machen. Hemmnisse durch Hierarchien,

Abteilungs- oder Teamgrenzen oder Marktbeziehungen gibt es nicht. Als Hürde für direkte Kommunikation und schnellen Austausch ohne Termin sehen Befragte die überwiegende Home-Office-Arbeit.

Ein befragter Programmierer führt den offenen Austausch darauf zurück, dass Mitarbeitende nicht untereinander konkurrieren:

> »Das kenn ich normalerweise ganz anders bei den anderen Unternehmen. Meiner Erfahrung nach ist das immer so, dass Leute immer ein bisschen ihre Projekte oder sehr darauf geachtet haben, was halt andere sehen, was umgesetzt wird. Und so weiter. Also so ein bisschen diese Konkurrenzsituation und das nehme ich bei uns gar nicht wahr, sondern ganz im Gegenteil. Man kümmert sich untereinander.« (Programmierer1)

Da einige Mitarbeitende Teil mehrerer Kreise sind (bspw. Product Ownerin oder Geschäftsführung), tauschen sich diese nicht nur innerhalb eines Teams aus, wie das in anderen Organisationsformen der Fall wäre.

Die Kommunikation ist kontinuierlich, flexibel und immer direkt möglich – ob durch Treffen oder Chat. Die Product Ownerin sieht, was direkte Ansprache betrifft, intern keine Hürden. Laut einem befragten Programmierer werden Anforderung auch außerhalb der Kreis-Treffen aufgenommen:

> »Das ist alles nicht festgefahren, sondern man weiß genau, wenn sich jetzt jemand meldet und sagt: ›Du, ich habe folgendes Problem‹, und das kurz beschreibt, dann kann man also, wenn es notwendig ist, auch alle Planungen irgendwie umwerfen. Und auch mal, ja und sich halt um das kümmern, ganz pragmatisch, was halt gerade notwendig ist. Und da gibt es dann keine Leute, die irgendwie blöd kucken oder sagen: ›Ja puh, aber wir wollten doch dies, das oder so.‹ Sondern da ziehen halt alle am gleichen Strang, ohne viele persönliche Befindlichkeiten, die dahinterstecken, ohne irgendwelche Ego-Geschichten.« (Programmierer1)

Programmierer2, der den Handels-Software-Teil programmiert, meint, dass er auch selbst etwas in das Ticketsystem Jira einpflegt, wenn er einen Fehler an der Software feststellt. Zudem schreiben interne Anwendende und Kund:innen per E-Mail oder Chat Anforderungen oder zumindest Ideen für solche.

Der regelmäßige Austausch in den Kreisen verringert mit der Zeit die sprachlichen Hürden zwischen den Beteiligten. Nur durch die getrennten Kreise besteht noch eine Wissensgrenze, weswegen die Product Ownerin zwischen diesen vermitteln muss. Dem Kommunikations-Kreis fehlt manchmal der Überblick über Prozesse wie »Anmeldung E-Autos« oder »Handel mit Emissionszertifikaten«. Der Produkt-Kreis wiederum bekommt die Anforderungen der Kommunikation nicht automatisch mit. Diese Grenzen sind für die Softwaregestaltung allerdings sekundär, weil alle Kreise unabhängig an einem Teil der Software arbeiten können.

Digitale Werkzeuge: vor allem für direkten Input – Ticketsystem, Chats, E-Mails

In dem Fall zeigt sich in dem STARTUP die für digitale Werkzeuge typische Verwendung in dezentralen Arbeitsprozessen der Softwaregestaltung. Das Ticketsystem, in das die

Product Owner:innen den Backlog pflegen und dessen Kanban-Board die Programmierenden für die Sprints nutzen, dient vor allem dazu, Anforderungen aufzunehmen, und weniger zur Überwachung oder Standardisierung. STARTUP nutzt die ganze Bandbreite an Kommunikationskanälen, um sich direkt abzusprechen und direkt Input für Anforderungen zu sammeln und Anforderungen auszuarbeiten: E-Mails, Telefon und Microsoft Teams (Videokonferenzen, Chat-Kanäle). Durch den Chat (MS Teams) können Beschäftigte direkt eine Person aus der Kollegenschaft anschreiben und eine Chat-Gruppe aufmachen, um eine Anforderung oder ein Thema zu bearbeiten.

Softwaretechnischer Zuschnitt: individuell und eigenständig durch Start-up

Über den für einen dezentralen Arbeitsprozess der Softwaregestaltung typischen individuellen softwaretechnischen Zuschnitt entscheidet das STARTUP eigenständig. Wobei die Teilnehmenden innerhalb der Kreise gemeinsam darüber entscheiden, was die Software können soll. Die Product Ownerin priorisiert, aber diskutiert dann mit anderen, was am wichtigsten ist. Nur wenn es nicht ganz klar ist, entscheidet sie. Dabei kann die Priorisierung flexibel verändert werden, d.h., es gibt keinen eindeutigen formal-bürokratischen Prozess dafür. Anforderungen von Nutzenden der Anmelde-App fließen ein, wenn sie als sinnvoll erachtet werden und es die Software nicht zu kompliziert macht.

Individuelle Anpassungen für Firmen, welche die Software zur Anmeldung von E-Autos als White-Label-Lösung einsetzen, gibt es nicht.

8.3.3. Zusammenfassung

Das Resümee fasst die Unterschiede von zentralisierter und dezentraler Softwaregestaltung je Kategorie des Arbeitsprozesses zusammen und stellt allgemeine Aussagen zu den Kategorien auf.

8.3.3.1. Unterschiede zwischen dezentralen und zentralisierten Arbeitsprozessen der Softwaregestaltung

Die Fallstudien INTERN1, INTERN2 und STARTUP können dem Typ dezentral und KOOP1, KOOP2, KOOP3 und PAKET dem Typ zentralisiert zugeordnet werden. Doch zeigen die Falldarstellungen, dass sie nicht immer klar einem der diametralen Typen entsprechen und sich teilweise dezentrale und zentralisierte Formen mischen. Hier seien zum Abschluss des Abschnitts die Einordnungen der Fälle kurz begründet.

Ob dezentrale oder zentralisierte Softwaregestaltung: Egal welche Arbeitsteilung, Grundkoordination oder Architektur vorliegt, findet in den Fallstudien durch den entsprechenden AP die Softwaregestaltung statt. Für Fälle wie STARTUP, bei denen die Organisation auf die Softwaregestaltung ausgerichtet ist, ist aber der Koordinationsaufwand deutlich geringer. In Fällen wie KOOP1 ist der Koordinationsaufwand größer, dafür bestehen aber auch erhebliche Möglichkeiten, Synergien durch einen Standard zu heben. Bei PAKET müssen sich die EVU dem von der Softwarefirma gestalteten Standard unterordnen.

In jedem Fall übernehmen die **Rollen** Aufgaben der Koordination und der Anforderungsaufnahme oder -ausarbeitung. Die Schwerpunkte sind jedoch unterschiedlich. Bei der **zentralisierten** Softwaregestaltung der Fallstudien KOOP1, KOOP2, PAKET und

KOOP3 sind extra Rollen mit der Koordination beschäftigt: Anforderungsmanagement, Key Account Managende, IT-Projektleitung, IT-Koordination, Prozessmanagement, Lösungsarchitekt oder Anwendungsbetreuung. Das liegt schlicht an den zusätzlichen Koordinationsaufgaben aufgrund der Arbeitsteilung zwischen Anwendung und Entwicklung, die sich auf mehrere Organisationen erstreckt und wenn mehrere Organisationen eine Software gestalten wollen. Bei KOOP3 ist aufgrund der netzwerkförmigen Grundkoordination der Koordinationsaufwand geringer, weil die Softwaregestaltung keine Hierarchien und Marktbeziehungen überwinden muss. Bei **dezentralen** Softwaregestaltungen wie INTERN1, INTERN2 und STARTUP liegt der Fokus darauf, Anforderungen aufzunehmen und an die Programmierenden zu übergeben. Koordinative Aufgaben fallen zwar auch an, aber in einem geringeren Umfang. Wobei bei INTERN2 auch mehr Koordinationsaufwand besteht, den einzelne Mitarbeitende erledigen müssen, weil mehrere Fachbereiche bei der Softwaregestaltung zusammenarbeiten. Bei STARTUP ist der Koordinationsaufwand geringer: Es existiert eine Netzwerkorganisation, d.h., die Beschäftigten müssen keine Hierarchien, Abteilungs- oder Organisationsgrenzen bei der Softwaregestaltung berücksichtigen.

Die **Abläufe** sind in den **zentralisierten** Fällen aufgrund der Arbeitsteilung darauf ausgerichtet, immer wieder Erwartungen zwischen den beteiligten Organisationen abzugleichen, mit Eskalationen umzugehen und Konflikte zu lösen. Bei KOOP1, KOOP2 und PAKET finden Abstimmungen auf mehreren Ebenen zwischen den Organisationen statt. Strategische Themen verhandeln Führungskräfte bzw. Manager:innen in entsprechenden Gremien oder Terminen. Für die Abstimmung über einzelne Anforderungen gibt es separate Abläufe, wie das Anforderungsmanagement bei KOOP1 oder Arbeitskreise bei PAKET. KOOP1 setzt einen Mediator und Key Account Managende ein, um die Beziehungen zu pflegen und mit Konflikten umzugehen. In dem Fall dienen die Abläufe für Verhandlungen darüber, was zentral das IT-DL und was dezentral die EVU gestalten. Bei KOOP3 gibt es regelmäßig Treffen zwischen dem IT-DL und der Softwarefirma und es entstehen aus Projekten mit EVU neue Anforderungen an die IoT-Software. Allerdings ist aufgrund der netzwerkförmigen Grundkoordination und der kooperativen Beziehungen der Koordinationsaufwand gering. Bei den **dezentralen** Fällen konzentrieren sich die Abläufe auf die Aufnahme und Ausarbeitung von Anforderungen. INTERN1 nutzt vielfältige Methoden, um die Anwendenden direkt einzubeziehen. Bei INTERN1, INTERN2 und STARTUP arbeiten Programmierende und Gestaltende wie Product Owner sehr kontinuierlich und langfristig zusammen. Wobei bei STARTUP aufgrund der Grundkoordination die Abläufe weder Hierarchien noch Märkte überwinden müssen.

Die **kommunikativen Beziehungen** müssen bei den **zentralisierten** Fällen aufgrund der Arbeitsteilung die Spannungen ausgleichen: zwischen der notwendigen Kooperation für die Softwaregestaltung einerseits und hierarchischen und marktförmigen Beziehungen zu anderen Organisationen andererseits. Zudem sind bei den Fällen KOOP1, KOOP2 und PAKET bürokratischere Anforderungsprozesse vorhanden. Zum Beispiel müssen Führungskräfte Anforderungen absegnen oder es existiert ein formalisiertes Vorgehen. Bei KOOP3 ist das aufgrund der netzwerkförmigen Grundkoordination und den kooperativen Beziehungen nicht notwendig. Bei den **dezentralen** Fällen liegen kooperative und direkte Kommunikationsbeziehungen vor. Die unkomplizierte und offene Zusammenarbeit betonen besonders Befragte von STARTUP und INTERN1. Bei

INTERN1 und vor allem INTERN2 kommen bürokratische und formalisierte Elemente hinzu. In ersterem Fall existieren Hierarchien zwischen den Product Ownern (z.B. wer über Priorisierung entscheidet) und bei INTERN2 entscheidet die Anforderungsrunde über Anforderungen.

Die direkte Kommunikation schlägt sich in den **dezentralen** Fällen INTERN1 und STARTUP in den verwendeten **digitalen Werkzeugen** nieder. In beiden Fällen können Beteiligte der Softwaregestaltung direkt via Chat Anforderungen aufnehmen und sich absprechen, d.h. sich ohne formale Hürden austauschen. Bei den **zentralisierten** Fällen KOOP1, KOOP2 und PAKET dienen die digitalen Werkzeuge dazu (vor allem die Ticketsysteme), um für Transparenz zwischen den beteiligten Organisationen zu sorgen. Vor allem für die EVU-Kundschaft der IT-DL von KOOP1 und KOOP2 oder der Softwarefirma von PAKET ist das Ticketsystem wichtig, um vertragliche Vereinbarungen zu kontrollieren und durchzusetzen (z.B. durch in SLA festgelegte Reaktionszeiten auf Tickets). Aufgrund der netzwerkförmigen Grundkoordination und der kooperativen Beziehungen spielt beim Fall KOOP3 das Ticketsystem der Softwarefirma primär eine koordinierende Rolle. In Zukunft soll es dazu dienen, dass das IT-DL selbst Tickets bearbeiten kann, was den Charakter der Ko-Produktion in dem Fall noch verstärkt.

Bei den **dezentralen** Fällen INTERN1, INTERN2 und STARTUP ist der **softwaretechnische Zuschnitt** individuell. Bei KOOP1 existiert ein ausgefeilter Ablauf, mit dem sich die EVU darüber verständigen, was in den gemeinsamen Standard einfließt und was sie individuell selber gestalten. Bei KOOP2 existiert ein solcher Ablauf nur für bestimmte Teile der Software. In beiden Fällen gestalten einzelne EVU selbst dezentral Software unabhängig von einem gemeinsamen Standard. PAKET entwickelt **zentral** eine Standardsoftware, die allerdings individuelle Einstellungen durch die EVU zulässt. Die Möglichkeiten individueller Einstellungen nutzen einige (vor allem größere) EVU ausgiebig und haben dafür extra Rollen (z.B. Anwendungsbetreuende). Bei KOOP3 entwickelt die Softwarefirma ebenfalls eine Standardlösung, allerdings ohne große Einstellungsmöglichkeiten und dank Schnittstellen erweiterbar durch individuelle Module.

8.3.3.2. Ergebnisse des Fallvergleichs je Kategorie

Die Fallstudien zeigen, dass die Organisationen durch Rollen, Abläufe, kommunikative Beziehungen und softwarebasierte Werkzeuge die Softwaregestaltung kontrollieren und wie der softwaretechnische Zuschnitt der Software zustande kommt. Im Folgenden werden allgemeine Aussagen zu den Kategorien aufgestellt, die sich aus dem Fallvergleich ergeben.

Rollen – situativ, teils mehrere und wechselnde Rollen übernehmen

Im Konzept der soziotechnischen Netzwerkarbeit erklärt der Rollenbegriff, wie Einzelne ihr Handeln auf die Softwaregestaltung ausrichten, ohne dass ihnen genau gesagt wird, was sie in der jeweiligen Situation konkret en détail zu tun haben. Die Fallstudien zeigen, dass das Konzept der Rolle den empirischen Umständen gerecht wird, dass 1. viele der Befragten nicht mehr nur eine Rolle haben, sondern mehrere und wechselnde. Zudem 2. sind die Befragten weniger auf vordefinierte Handgriffe spezialisiert, sondern vielmehr Teil von mehreren Arbeitsprozessen wie Softwaregestaltung und -anwendung und die

Organisation und die oder der Einzelne muss die Rollen entsprechend ausgestalten und kontrollieren: situativ, verhandelt, als Teil eines Arbeitsprozesses zwischen Anwendung und Programmierung. Außerdem findet 3.Rollen-Handeln innerhalb von Hierarchien statt, auch wenn Methoden wie Scrum keine formalen Hierarchien vorsehen.

Tabelle 22: Rollen: reine Softwaregestaltungsrollen oder gemischt mit anderen und wer Schulungen zur Rolle erhalten hat

Fall	**Rolle nur für Softwaregestaltung (Anforderungen, Koordination)**	**Gemischte Rollen (Anwendung, Programmierung, Customizing, (IT) Management, Support für SA)**	**Schulung Rollen-Kompetenz (in Klammern welche Rolle Schulung absolviert hat)**
INTERN1	Product Owner:in (PO)	Key User:innen (KU), führender PO, Anforderungsmanager, Teamleiter als Scrum Master (SM)	PO Schulung (POs), Scrum Schulung und Story Writing (Anforderungsmanager)
INTERN2		KU, Ansprechperson in Fachbereichen, SM, Anforderungsmanager, PO	PO Schulung (Anforderungsmanager, PO), SM Schulung (SM)
KOOP1	Anforderungsmanager (IT-DL), IT-PL, Anwendungsbetreuer	IT-Berater, Prozessmanager, IT-Manager, Anforderungsmanager (EVU)	SM, PO, PM (Anforderungsmanager und PL IT-DL), Schulung Requirements Engineering (Anwendungsbetreuer)
KOOP2	IT-Koordinator Fachbereich, IT-PL	IT-Koordinator IT-Abteilung, KU, Manager Digitalisierung, Teamleiterin, IT-Berater, Change Manager	Scrum (IT-Berater, Change Manager), PM (Change Manager, PL), Change Manager schult intern zu Projektthemen
PAKET	EW-Fachleute in Arbeitskreisen, Prozessbetreuer	Teamleitung Entwicklung, IT-Berater, Lösungsarchitekt, KU, Anwendungsbetreuer	PM (Anwendungsbetreuung)
KOOP3	PO (EVU, IT-DL)	Projektmanager, Account Manager	unbekannt
STARTUP	PO	Solution Architect, Programmierer, Geschäftsführung	PO und Programmierer keine

Zu 1.: In allen Fallstudien gibt es Rollen, die für die Softwaregestaltung zuständig sind. Sie koordinieren oder schreiben selbst Anforderungen. Meistens haben diese Rollenträger noch weitere Verantwortungsbereiche. Wie viele Rollen jemand übernimmt und in wie vielen Arbeitsprozessen, ergibt sich im Wesentlichen aus: erstens der Überschneidung der Softwaregestaltung mit anderen Arbeitsprozessen. Manche Softwaregestaltenden sind in mehreren Projekten aktiv und/oder sind zusätzlich Anwendende und Programmierende der Software. Zweitens ist die Spezialisierung bei kleinen EVU gerin-

ger oder sie verzichten ganz auf eine gestaltende Rolle (z.B. haben einzelne EVU keine eigenen Key User:innen oder internen Support für die Standardsoftware).

Die Überblickstabelle (Tabelle 22) fasst noch einmal zusammen, dass die meisten Softwaregestaltenden mehrere Rollen innehaben. Zudem zeigt sich, dass in vielen Fällen die Rollenträger Schulungen besucht haben, worauf weiter unten näher eingegangen wird.

Zu 2.: Die Fallstudien zeigen nicht nur, dass die Angestellten oftmals mehrere Rollen übernehmen. Zudem erlauben es die Rollen, situativ und unabhängig von einer hierarchischen Position erwartungsgemäß zu agieren. Zwar zeigt die Tabelle 22 oben, dass viele der Befragten für ihre Rolle Schulungen besucht haben. In den Fallstudien sind die Schulungen und Zertifikate für Methoden bzw. Rollen (wie z.B. Product Owner:in oder Projektmanagement) aber nur der Ausgangspunkt dafür, die Rollen in einer Organisation zu etablieren.

Im Arbeitsprozess der Softwaregestaltung reihen sich die Rollen zwischen Anwendung und Programmierung wie an einer Kette aneinander. Den Ablauf der Softwaregestaltung und die Tätigkeiten der einzelnen Rollen genau zu planen, ist jedoch schwierig, weil die Kompetenzen und Wissensbestände der einzelnen Beteiligten, die verbale Ausdrucksfähigkeit und der Mitteilungswille, die soziotechnische Konstellation sowie die Abläufe der Anforderungserarbeitung unterschiedlich sind. Vielmehr findet ein Erwartungsabgleich innerhalb des Netzwerks für Softwaregestaltung statt. Wie situativ das Gestaltungsnetzwerk die Rollen prägt, zeigt sich daran, dass Softwaregestaltende mal mehr, mal weniger energiewirtschaftliches Fachwissen haben und brauchen. Somit ist der Arbeitsprozess eine eigenständige Sozialisationsinstanz, wenn auch in vielen Fällen Schulungen in den jeweiligen Rollen die Grundlage liefern.

Zu 3.: Die Fallstudien zeigen, dass sich Rollen und hierarchische Positionen unterschiedlich zueinander verhalten. Es ist je Fall anders, wie unabhängig von der formalen Hierarchie und von der Position einzelne Rollen agieren können. Vor allem in den reinen Netzwerken von KOOP3 und STARTUP agieren die Rollen unabhängig von hierarchischen Vorgaben. In mehreren Fällen geht eine hierarchische Position mit einer Rolle in der Softwaregestaltung einher. Bis auf STARTUP treffen hierarchisch den Softwaregestaltenden höhergestellte Personen in allen Fallstudien Entscheidungen über Ressourcen wie IT-Budget oder Personal.

Fazit: Für die Beschäftigten bestehen im Arbeitsprozess der Softwaregestaltung zusätzliche Erwartungen, wenn auch in unterschiedlichem Ausmaß und abhängig von der jeweiligen Situation. Zu den Erwartungen gehören jene, zwischen mehreren Rollen zu wechseln, in unterschiedlichen Arbeitsprozessen mitzuarbeiten, sich einzufügen und entsprechend anzupassen. Dazu gehört, mit ihrem jeweiligen Wissen herauszufinden, was zu tun ist. Sie sollen mit nicht immer genau geregelten Verantwortlichkeiten und Erwartungs-/Rollenkonflikten umgehen und vorhandene Spielräume für die Softwaregestaltung nutzen. Zudem besteht die Erwartung, die Rolleneinhaltung mit Kolleg:innen abzustimmen und mit zu kontrollieren. Diese Erwartungen kommen zu jenen aus 6.4.3.2 hinzu: organisationale Grenzen zu überbrücken, kooperativ zu sein, selbstorganisiert zu arbeiten, sich auf Softwareobjekte einzulassen, mit Nicht-Wissen umzugehen oder sich im organisationalen Netzwerk zu bewegen und zu lernen.

Ablauf – kombiniert mit Gestaltungsnetzwerk, Primat der Kooperation, Feedbackschleifen und Lernprozesse

IT-Projekte, Scrum und Anforderungsrunden sind in allen Fällen die wesentlichen Methoden dafür, den Ablauf der Anforderungserarbeitung und deren Übergabe an die Programmierenden zu organisieren. Sie stellen auf unterschiedliche Weise Interdisziplinarität her und werden immer situativ angepasst angewendet, z.B. Scrum ohne Review, mit mehreren Product Owner:innen oder ohne Scrum Master:in. Die Fälle zeigen Folgendes: A) Sie verbinden für die Softwaregestaltung einen mal mehr und mal weniger formalisierten Ablauf mit einem Netzwerk an Beziehungen für Anforderungen. B) Der Ablauf muss vereinbar sein mit verschiedenen Handlungsorientierung, wobei der Primat der Kooperation gilt. C) Für die Softwaregestaltung sind Feedbackmöglichkeiten wichtig, die der Ablauf herstellt. D) Der Ablauf lässt sich an die jeweilige Konstellation und die vorhandenen Rollen anpassen lässt und er in sämtlichen Fallstudien durch Lernprozesse geprägt ist, was seine Ausgestaltung anbelangt.

A) Die Fallstudien zeigen, dass es nicht um den Ablauf (wie IT-Projekte oder Scrum) allein geht, sondern auch um die dazugehörenden Gestaltungsnetzwerke. Zusätzlich zu den Projekten oder zu Scrum existieren in den Fallstudien immer horizontale Netzwerke. Dabei gibt es sowohl klare, formale Anforderungswege als auch informelle Beiträge Einzelner. Formalisiert sind z.B. die regelmäßigen Treffen (wer teilnimmt, Turnus etc.), das Niederschreiben der Konzepte, welche softwarebasierten Werkzeuge verwendet werden und einzelne Rollen, um die Kommunikation unter den Beteiligten am Arbeitsprozess der Softwaregestaltung zu organisieren.

B) Der Ablauf muss die Kombination verschiedener Handlungsorientierungen zulassen (wie schon bei der Projektarbeit unter 6.3 ausgeführt), weil er in unterschiedlichen Grundkoordinationen funktionieren muss. Bei der internen Softwaregestaltung sind es die Hierarchien und bei der firmenübergreifenden Softwaregestaltung die Marktbeziehungen, mit denen die Softwaregestaltenden umgehen können müssen. Es wird erwartet, jenseits ökonomischer Kriterien oder Weisungsbefugnissen informelle Kooperationsmöglichkeiten herzustellen, z.B. trotz formaler Vorgaben von Führungskräften, die primär auf eine kostengünstige Umsetzung abzielen, oder Verträgen, die kooperatives Arbeiten erschweren, einen persönlichen Austausch und Vertrauen herzustellen.

C) Für die Softwaregestaltung ist das Feedback zentral, weil es das gegenseitige Verständnis sichert. Ein großer Unterschied zwischen Scrum, IT-Projekten und den Anforderungsrunden ist, inwiefern Feedbacks Teil der Methode sind. Bei Scrum sind Feedbacks fester Bestandteil (siehe die Grundlagen 5.2.4). INTERN1 und INTERN2 zeigen eine feedback-intensive Kommunikation zwischen Anwendenden, Gestaltern und Programmierern, die sogar über das in Scrum Vorgesehene hinausgeht. Bei INTERN1 gibt es noch Resonanzgruppen, in denen Anwendende Feedback zu einer Umsetzung geben. Bei INTERN2 führt der befragte Product Owner zusätzlich noch Interviews mit Anwendenden des Fachbereichs (für Anforderungen und nach der Umsetzung). Unabhängig von der Methode sind allgemein Tests eine Möglichkeit für Feedbacks in allen Fallstudien: Softwarefirma oder IT-DL stellen eine Version der Software IT-Beratenden, Fachexpert:innen oder Anwendenden auf einem Testsystem zur Verfügung. Diese geben dann Rückmeldung darüber, ob Fehler auftreten oder was noch verbessert werden könnte.

D) In vielen Fällen ist der Ablauf in einem Lernprozess entstanden und die jeweilige Organisation entwickelt ihn weiter. Im Scrum gibt es die Rolle Scrum Master:in, die für einen reibungslosen Ablauf verantwortlich ist und versucht, den Prozess zu verbessern. Im STARTUP gibt es einen agilen Coach, der kontinuierlich den Ablauf optimiert. Anforderungsrunden wie in KOOP1 oder INTERN2 haben sich in einem Lernprozess herausgebildet und sind nun fest etabliert. Der zuständige Scrum Master in INTERN2 sieht sich als agiler Coach. Letztendlich sind im Kontext der Softwaregestaltung die Abläufe Gegenstand von Optimierungen, die aber nicht alle Fälle gleich intensiv betreiben.

Kommunikative Beziehungen – trotz Markt oder Hierarchie

Beim Vergleich der Fallstudien fällt auf, dass Rollen und Abläufe allein nicht ausreichen, um Software zu gestalten. Kommunikative Beziehungen sind notwendig, wenn es darum geht, A) Märkte und Hierarchien zu überwinden in der Kommunikation. Dann ist ein B) bestimmtes Maß an persönlicher Beziehung wichtig, um sich auszutauschen. Zuletzt ist C) eine gemeinsame Sprache nützlich, damit Beteiligte sich untereinander auf Anforderungen an die Programmierenden verständigen können.[17]

A) Kommunikative Beziehungen trotz Markt und Hierarchie: Bei KOOP1 und KOOP2 gibt es zusätzlich zum Vertragsverhältnis zwischen IT-DL und EVU langfristige Beziehungen. Gleichzeitig sind jene zwischen den Firmen brüchiger. Mehrere Befragte drücken eine Abwägung zwischen Alleingängen und Kooperation aus. Im Gegensatz zu KOOP2 moderiert bei KOOP1 die Konflikte zwischen den Organisationen, was gemeinsame Strategie o. ä. betrifft, ein professioneller Mediator. Im Fall PAKET existieren kooperative, langfristige Beziehungen zu EVU. Allerdings nur zu einzelnen, mit denen die SF regelmäßig in IT-Projekten oder in Arbeitskreisen zusammenarbeitet. Bei den Fallstudien INTERN1 und INTERN2 existieren persönliche Beziehungen zwischen Softwaregestaltung, Programmierenden und anwendenden Fachbereichen jenseits formaler Hierarchien und Fachbereichsgrenzen: zum einen, weil die Product Owner:innen vorher in Fachbereichen gearbeitet haben, und zum anderen, weil die Anforderungsmanagerin aus INTERN1 schon länger mit dem FB zusammenarbeitet. Bei INTERN2 sieht sich der Scrum Master als Mediator zwischen den Fachbereichen, die an der Anforderungsrunde teilnehmen. Bei KOOP3 und beim STARTUP sind Beziehungen primär kooperativ und die Zusammenarbeit nicht durch Markt und Hierarchie geprägt.

In allen Fallstudien sehen Befragte den direkten Kontakt zwischen Anwendenden, Softwaregestaltenden und Programmierenden als vorteilhaft an. Dabei erwähnen einige aus den ersten fünf Fallstudien, dass versucht wird, die direkte Kommunikation zwischen Softwareanwendung, -gestaltung und -programmierung zu unterbinden, z.B. durch Führungskräfte, Anforderungsmanagende oder vertragliche Regelungen. Die Softwarefirma von PAKET hat viele EVU als Kundschaft. Durch eine Kommunikation beschränkt auf das Ticketsystem kann sie deren vielfältige Anfragen besser verwalten.

17 Um zwischen einer gemeinsamen Sprache, einem gemeinsamen Verständnis und einer gemeinsamen Wissensbasis empirisch unterscheiden zu können, hätten noch detailliertere Fragen gestellt werden müssen. Das ist aber nicht passiert, weswegen im Weiteren keine entsprechend differenzierte Darstellung erfolgt.

Die Fallstudien lassen nicht den Schluss zu, dass der direkte Austausch innerhalb einer Organisation immer einfacher oder schwieriger wäre als zwischen verschiedenen Organisationen. Auf jeden Fall ist es einfacher im STARTUP, wo weder Führungskräfte noch Kund:innen diese erschweren, z.B. durch ein Management, welches die Kommunikation untereinander auf ein Ticketsystem beschränkt, oder eine Kundschaft, die eine funktionierende Software, aber keine aufwendigen Abstimmungen erwartet.

B) Es zeigt sich im Vergleich der Fallstudien eine enge Verbindung zwischen persönlichen Beziehungen und Kommunikation. In den Fällen sind die Beteiligten der Softwaregestaltung nicht nur unpersönliche Informationsvehikel. Sie kommunizieren ausgehend von zwischenmenschlichen Beziehungen. Diese basieren auf Langfristigkeit, Vertrauen, individuellen Kompetenzen, was Beziehungen und Kommunikation anbelangt, und weniger auf Schulungen oder Methoden.

In den Fallstudien charakterisieren die Befragten ihre Beziehungen unterschiedlich. Die verwendeten Begriffe lassen sich drei Clustern zuordnen, die Abgrenzungen zu marktförmigen und hierarchischen Beziehungen sowie die Emotionalität der Beziehungen betonen:

- Zum einen wird auf die Emotionalität und die damit zusammenhängende Kompetenz verwiesen: »Familie«, »freundschaftlich«, »miteinander sprechen«, »Konflikte persönlich klären«, »sich kümmern«, »nicht mehr verfeindet sein«, »sympathisch«, »persönliche Bindung«, »menschliche Komponente«, »sich gut kennen«.
- Es wird ein Gegensatz zum Markt betont: »keine Konkurrenz«, »Vertrauen«, »partnerschaftlich«, »transparent«, »gemeinsam«, »miteinander, »ohne Ego-Geschichten«.
- Ein nicht-hierarchisches Verhältnis wird ausgedrückt: »auf Augenhöhe«, »offen/Offenheit«, »Geben und Nehmen«.

Obwohl Befragte eine direkte Kommunikation und Beziehungen als wichtig ansehen, gibt es keine Trainings dazu. Weder wenden die Organisationen spezielle Methoden zum Aufbau von Beziehungen an, noch bieten sie Schulungen zum partnerschaftlichen Umgang miteinander an. Nur EVU2 (KOOP2) schult in der Methode »Working Out Loud«, die beim verbalen Teilen von Wissen helfen soll. Auch die Mitarbeitenden von STARTUP haben keine Schulungen im kooperativen Handeln besucht. Ein befragter Programmierer des Start-ups meint, er ist es bereits aus seinen vorhergehenden Jobs gewohnt, so zu arbeiten.

C) Bei der Sprache und dem gemeinsamen Verständnis zeigen die Fallstudien, dass nicht jede:r jede:n immer sofort verstehen muss. Aber Wissen über den Fachbereich und sprachliches Mitteilungsvermögen, langfristige Zusammenarbeit oder ein Rollen-Tausch helfen bei der Kommunikation.

Eine gemeinsame Sprache und ein gemeinsames Verständnis sind wichtig für die Softwaregestaltung. In den Fallstudien reicht es aber aus, dies mit der Zeit zu lernen. Softwaregestaltende sind jene, die »IT-Deutsch« (Anforderungsmanagerin INTERN1) lernen müssen. Dafür müssen Key User:innen, Anforderungsmanagende, IT-Projektleitende, Anwendungsbetreuende oder Product Owner:innen aber zumindest bereit sein, es zu lernen.

Dabei ist die Bedeutung von Wissen nicht eindeutig. Zum einen ist Fachwissen nicht immer entscheidend dafür, zu verstehen, was eine andere Person sagt oder schreibt: Softwaregestaltende zwischen Anwendung und Programmierung verfügen nicht immer über Fachwissen oder lernen es, wenn notwendig, erst im Austausch und über die Zeit. Der Product Owner bei INTERN2 meint, er braucht immer weniger davon. Ein Programmierender von STARTUP sagt, er braucht das fachliche Verständnis, weil es die Kommunikation vereinfacht. Es ist aber nicht unbedingt notwendig. Zum anderen ist ein bestimmtes Maß an Wissen über die Möglichkeiten von Softwaregestaltung – ob Standard oder individuell – notwendig. Genauso erforderlich ist ein rudimentäres Wissen über den Anwendungsbereich der Software. Einzelne Product Owner:innen wie von STARTUP oder KOOP3 haben interdisziplinäres Wissen in einem Ausmaß, das es ihnen erlaubt, selbst Anforderungen zu schreiben, mit denen Programmierende etwas anfangen können. Die Ausarbeitung einer Anforderung kann dann trotzdem noch Feedbackschleifen beinhalten, weil mehrere Teilnehmende ihr Wissen einbringen. 8.4 geht ausführlicher auf das Wissen der Softwaregestaltenden ein und zeigt, dass es auch stark von der Rolle im Arbeitsprozess abhängt, welches Wissen der oder die Einzelne benötigt oder lernt. Zudem ist grundsätzlich, wenn es um Softwaregestaltung geht, die Praxis des Arbeitsprozesses wichtiger. Das liegt daran, weil es ja um das Schreiben von Anforderungen für Programmierende geht und damit darum, Wissen in Software zu übersetzen, und nicht, Wissen für sich zu behalten.

Sprachliche Fertigkeiten sind neben persönlichen Beziehungen oder interdisziplinärem Wissen wichtig, damit sich zwei Personen überhaupt austauschen. Das zeigt sich z.B. bei INTERN1. Die Anforderungsmanagerin unterbindet die direkte Kommunikation zwischen Anwendenden und Programmierenden, und zwar deshalb, weil Letztere sich nicht für die Anwendenden verständlich ausdrücken können. Eine andere Möglichkeit, eine gemeinsame Sprache zu entwickeln, ist der Rollentausch. In einigen Fällen werden Beratende genannt, die vorher programmiert haben, oder Product Owner:innen, die vorher im Fachbereich waren. Sie haben den Vorteil, dass sie die fachliche und technische Welt kennen und dadurch besser über sie sprechen können. Zuletzt erweist sich in vielen Fällen die längere Zusammenarbeit über Wissensgrenzen hinweg als hilfreich (z.B. feste Ansprechpersonen zu haben), um sich immer besser zu verstehen und eine gemeinsame Basis an Wissen und Vokabular auszubilden.

Werkzeuge – Softwaregestaltung als (teil-)integrierter digitaler Prozess

In sämtlichen Fällen gehören Ticketsysteme zum digitalen Werkzeugkasten der Softwaregestaltung. Bei dezentralen Arbeitsprozessen der Softwaregestaltung geht es in erster Linie darum, Anforderungen aufzunehmen, bei zentralisierten um Transparenz und Abstimmungen. Koordinieren und Kommunizieren steht damit im Vordergrund und weniger die direkte, zentrale Kontrolle einzelner Arbeitsschritte von Mitarbeitenden. Das zeigt sich daran, dass in einigen Fällen unterschiedliche, nicht digital integrierte Softwarelösungen nebeneinander existieren, obwohl es technisch möglich wäre, nur eine (Ticket-)Software zu verwenden oder sie alle miteinander zu verbinden. Meist werden Ticketsysteme verwendet, die aus der Softwareentwicklung/-gestaltung stammen, wie Microsoft Azure DevOps, Jira oder von SAP.

Von einer technischen Kontrolle durch ein softwarebasiertes Werkzeug kann nur insofern gesprochen werden, weil Eingabefelder und Prozesswege (der Tickets bzw. Anforderungen) durch Software vorgegeben sind und der Zugriff auf die verschiedenen Softwaresysteme über Benutzer:innenrechte technisch beschränkt ist. Allerdings entscheiden unterschiedliche Mitarbeitenden und nicht nur das Management über die Priorisierungen von Tickets, Zugang zu den Systemen oder wann der Status welcher Anforderung von »Konzeptionierung« in »Programmierung« oder »Test« wechselt. Es gibt keinen festen, maschinellen Takt. Je nach Fallstudie bestimmen vielmehr Deadlines, Reaktionszeiten (festgelegt durch SLA) oder ad hoc zu erledigende Anforderungen wie Fehler oder Tests den Arbeitsrhythmus. Das spielt vor allem in den Beziehungen zwischen Softwarefirma und EVU oder IT-DL und EVU in den Fällen KOOP1, KOOP2 und PAKET eine Rolle. Durch SLA sind Strafzahlungen festgelegt, welche bei zu langsamer Reaktion auf bestimmte Tickets fällig werden. Oft ist es auch die staatliche Regulierung, die Termine für eine Umsetzung vorgibt, oder das IT-Budget setzt ein Limit, was die aufgewendete Zeit anbelangt. Das ist aber, wie gesagt, keine technisch festgelegte Zeiteinteilung.

Der durch die Transparenz sich einstellende Panoptikum-Effekt kann nicht ausgeschlossen werden. Denn die Ticketsysteme geben z.B. Auskunft darüber, welches Ticket welchen Status hat, wer es umsetzt, wie es umgesetzt wurde. Zum einen verändert allein das Wissen darum, dass alles digital dokumentiert ist, das Handeln der Mitarbeitenden. Zum anderen lassen die softwarebasierten Werkzeuge nachträgliche Auswertungen zu. Dabei entzieht sich eine direkte, persönliche Kommunikation zwischen Mitarbeitenden nicht dieser Transparenz. Denn die Organisation erwartet von Mitarbeitenden in den Fallstudien, dass sie die Konzepte als Tickets oder in anderer Form dokumentieren. Das heißt, die informelle Kommunikation wird nachträglich formalisiert. Aber auch andere digitale Werkzeuge wie E-Mail- oder Chatprogramme dokumentieren Kommunikation. Dass das Management diese unterschiedlichen Medien zur Bewertung von Mitarbeitenden direkt nutzt, erwähnt keiner der Befragten, ebenso wenig eine individualisierte Kontrolle. Es ist anzunehmen, dass die Betriebsräte nicht nur in der Softwareanwendung die individuelle Verhaltenskontrolle unterbinden, sondern auch für die Softwaregestaltung.

Ticketsysteme sind nicht die einzigen Werkzeuge, welche den Arbeitsalltag der Softwaregestaltung prägen. Für viele EVU ist es mittlerweile Realität, dass sie drei Systeme zur Verfügung haben: eines, auf dem die aktuelle Version der Software läuft (das Produktivsystem), ein Testsystem mit zu testenden Neuerungen der Software und ein Entwicklungssystem, auf dem sie programmieren. Bei SAP R/3 und damit den ersten vier Fallstudien ist die Entwicklungs- und Testumgebung integriert, d.h. die Arbeitsprozesse von Anwendung (wenn sie testen müssen), Programmierung und Gestaltung arbeiten mit einer Softwarelösung (Tests durchführen, Analyse der Software). Sie ist die Infrastruktur, die Softwaregestaltung in den EVU ermöglicht. Bei PAKET haben die EVU eine Testumgebung, bei KOOP3 nicht, wobei es in Planung ist. Da STARTUP die Software selbst entwickelt, ist die digitale Infrastruktur zum Testen und Programmieren entsprechend vorhanden.

Softwaretechnischer Zuschnitt – zwischen individuell und Standard

Der softwaretechnische Zuschnitt entscheidet darüber, ob z.B. eine Anforderung Teil eines Standards wird und welche Priorisierung sie hat. Dieser Teil des Arbeitsprozesses der Softwaregestaltung gehört zum Kernproblem der softwaretechnischen Gestaltungsmöglichkeiten (siehe 3.2). Zwischen den Fallstudien gibt es Unterschiede, wo und wer über den Zuschnitt entscheidet, wie dadurch Synergien gehoben und wie Prioritäten gesetzt werden. Es können sich verschiedene Gruppen bei der Entscheidung gegenüberstehen.

Der individuelle Zuschnitt zeigt sich daran, ob einzelne Anwendende Anforderungen einbringen können. Ob sie das tun können, hängt in den Fallstudien vor allem davon ab, wie viele Anwendende in wie vielen unterschiedlichen EVU es gibt. Zudem zeigt sich der individuelle Zuschnitt in den Fallstudien daran, welche Perspektive primär in die Software einfließt: jene des Managements, der Branche oder einzelner Anwendender. Dazu gehört, welche Priorität die zu diesen Perspektiven gehörenden Anforderungen haben. Zuletzt gibt es Fälle, in denen die Begrifflichkeiten eines individuellen Fachbereichs in die Software einfließen – ob in deren Oberfläche oder den Quellcode.

Die Gestaltung eines Standards braucht einen aufwendigeren Koordinierungsprozess, und entweder ein IT-DL, ein EVU oder eine Softwarefirma verantworten ihn. Dabei fällt auf, dass es einen institutionellen Unterschied gibt, was den Grad der Standardisierung anbelangt: Im regulierten Netzbereich ist es einfacher, sich auf einen gemeinsamen Standard zu einigen, als im wettbewerbsorientierten und auf die Wünsche der Kundschaft ausgerichteten Vertriebsbereich. Einige Fallstudien zeigen, dass es in den EVU einen bunten Flickenteppich geben kann und es schwer herauszufinden ist, wo und wer innerhalb der Organisation einen Standard oder eine individuell gestaltete Software anwendet. Einer Softwarefirma fällt es weniger schwer, zentral über einen (Branchen-)Standard zu entscheiden und diesen durchzusetzen, als wenn mehrere EVU darüber verhandeln. Bietet STARTUP seine App auch anderen Organisationen an, wird aus einer individuellen Software ein Standardprodukt.

Letztendlich zeigen die Fallstudien, dass eine Organisation entsprechend organisiert sein muss, um die Möglichkeiten der Softwaregestaltung für Synergien zu erkennen. Ob durch eine einzelne Person oder ein Gremium: Die Organisation muss fähig dazu sein, Synergien zu erkennen. Bei KOOP1 gibt es ein Anforderungsmanagement, in dem sich verschiedene EVU darüber austauschen. Bei INTERN2 kann die Anforderungsrunde mehrere Fachbereiche Synergien erkennen. Im EVU3 von KOOP1 ist es Aufgabe des Prozessmanagers, dies für die zwei Bereiche Privat- und Geschäftskundschaft zu tun. Aber auch Einzelpersonen achten auf Synergiepotenziale: Bei INTERN1 achtet die Anforderungsmanagerin darauf und bei KOOP1 sind die IT-Beratenden dazu angehalten.

8.4. Folgen für die Arbeit der Beschäftigtengruppe der Softwaregestaltenden

Die soziotechnische Konstellation und der Arbeitsprozess der Softwaregestaltung prägen die Arbeit der Softwaregestaltenden in den Organisationen der Fallstudien. Dabei ist es für die Beschäftigtengruppe der Softwaregestaltenden von Vorteil, wenn sie in einer reinen Netzwerkorganisation arbeiten und nicht in einer Matrixorganisation. Das ist

unabhängig davon, ob sie eine Individual- oder eine Standardsoftware gestalten. Warum das so ist, zeigt der Abschnitt mithilfe dreier Kategorien:

- Beschäftigungssystem: Softwaregestaltende zeichnen sich weniger durch eine hierarchische Karriere aus, sondern eher durch eine Karriere, in der es darum geht, Kompetenzen, Erfahrungen und interessante Projekte zu sammeln. Außerdem sind Softwaregestaltende in der Regel Akademiker:innen und flexibler in ihrer Beschäftigung (häufigere Wechsel von Organisationen, Projekten usw.).
- Kontrolle: Softwaregestaltende arbeiten eigenständig, gemäß ihrer Rolle entlang eines softwarezentrierten Arbeitsprozesses. Wobei die Grundkoordination der soziotechnischen Konstellation sich darauf auswirkt, wie stark Hierarchie oder Märkte die Arbeit der Softwaregestaltenden mitkontrollieren.
- Wissensverteilung: Es existiert eine Praxisgemeinschaft der Softwaregestaltenden, der es nicht darum geht, möglichst viel Wissen anzusammeln und zu konzentrieren. Sie will Wissen in Quellcode übersetzen. Nicht das Wissen der Softwaregestaltenden allein nimmt zu, sondern vor allem das in Software materialisierte und in verschiedenen digitalen Quellen hinterlegte Wissen (z.B. Quellcode, Ticketsystem, Dokumentation). So zeichnen sich Softwaregestaltende primär dadurch aus, zu wissen, wie Softwaregestaltung funktioniert. Auch wenn Softwaregestaltende interdisziplinär arbeiten: Die Fallstudien zeigen nicht, dass energiewirtschaftliches oder softwaretechnisches Wissen eine notwendige Bedingung für ihre Arbeit ist. Es ist hilfreich. Aber ob es notwendig ist und wie viel interdisziplinäres Wissen sie brauchen, ergibt sich aus der jeweiligen Praxis im Arbeitsprozess der Softwaregestaltung.

Der Abschnitt geht zum Schluss in der Zusammenfassung auf die Unterschiede zu den anderen zwei Gruppen der Anwendenden und Programmierenden ein. Der Vergleich hilft, die Besonderheiten der Arbeit der Softwaregestaltenden herauszuarbeiten. Zuerst stellt der nächste Punkt die unterschiedlichen Typen von Bedingungen vor, unter denen die Softwaregestaltenden arbeiten, um danach die Fälle darzustellen.

8.4.1. Softwaregestaltende: zwischen Matrix- und reiner Netzwerkorganisation

Für die Arbeit der Softwaregestaltenden lassen sich die Fälle grundsätzlich darin unterscheiden, ob sie aufgrund der soziotechnischen Konstellation und des Arbeitsprozesses der Softwaregestaltung in einer reinen Netzwerkorganisation arbeiten oder in einer Matrixorganisation.

Bei einer **reinen Netzwerkorganisation** findet die Softwaregestaltung im Rahmen der Grundkoordination Netzwerk statt. Die Kontrolle zeichnet sich durch eine Peer- und Objekt-Kontrolle aus (siehe 6.4.2.1) mit dem Schwerpunkt darauf, sich abzustimmen. Weil die formalen Hierarchien flach sind und für die Softwaregestaltenden eine Karriere in der Hierarchie keine Priorität hat, gibt es Kompetenzkarrieren. Die Gruppen von Softwaregestaltenden, Softwareanwendenden und Programmierenden unterscheiden sich primär soziotechnisch aufgrund ihres Verhältnisses zur Software und weniger durch rein soziale Hierarchien. Das heißt, sie sind nicht nur organisational getrennt bspw. durch verschiedene Rollen oder Teams, sondern durch unterschiedliche Zugän-

ge und Aufgaben, was die Software anbelangt. Das Wissen der Softwaregestaltung für den Arbeitsprozess der Softwaregestaltung ist durch Märkte oder Hierarchien nicht getrennt. Die Softwaregestaltenden können sich auf das Übersetzen zwischen Anwendung und Programmieren konzentrieren: Sie können sich das interdisziplinäre Wissen in der Praxis allein oder im Gestaltungsnetzwerk aneignen, weil das Wissen zugänglich ist und sie nicht nur temporär Teil des Gestaltungsnetzwerks sind. Darüber hinaus besteht eine gemeinsame Wissensbasis, z.B. weil die Programmierenden selbst über ausreichend interdisziplinäres Wissen für die Zusammenarbeit verfügen.

In einer **Matrixorganisation** findet die Softwaregestaltung im Rahmen der Grundkoordinationen Markt oder Hierarchie statt. Es besteht eine Spannung einerseits aus der Kontrolle durch Führungskräfte in einer Hierarchie oder Kundschaft und Verträge in einem Markt und andererseits einer reinen Koordination bzw. einem offenen Wissensaustausch. Weil Hierarchien oder Märkte zusätzlich zu einem Netzwerk der Softwaregestaltung existieren (z.B. bei IT-Projekten), können die Softwaregestaltenden sowohl eine Karriere in der Hierarchie als auch im Netzwerk machen. Wobei es dann eine Herausforderung für die Organisationen darstellt, beidem gerecht zu werden. Die Gruppen von Softwaregestaltenden, Softwareanwendenden und Programmierenden sind durch Hierarchien und Märkte getrennt. Ebenso verteilt sich das für die Softwaregestaltung benötigte Wissen auf Märkte und Hierarchien. Durch die Matrixorganisation ist die interdisziplinäre Zusammenarbeit nicht immer eingespielt und oftmals nur temporär. Das hat zur Folge, dass die Softwaregestaltenden eine interdisziplinäre Wissensbasis mitbringen oder diese gemeinsam mit Anwendenden und Programmierenden erarbeiten müssen.

Tabelle 23: Idealtypen Matrix- und reine Netzwerkorganisation

Typ	Kontrolle	Beschäftigungssystem	Wissensverteilung
Reines NW	Peer- und Objekt-Kontrolle, Fokus auf Koordination	keine Karriereleiter, Aufgaben und Position abhängig von Stellung zur Software	Wissen rein horizontal im Gestaltungsnetzwerk verteilt
Matrix	Mischung mit Markt und Hierarchie	Teil von Märkten oder Hierarchien	Wissen getrennt durch Hierarchien oder Märkte

Sind nun die einzelnen Fälle Beispiele dafür, dass Softwaregestaltung in einer reinen Netzwerk- oder in einer Matrixorganisation stattfindet? Antwort darauf geben die Falldarstellungen und die Zusammenfassung am Schluss dieses Abschnitts.

8.4.2. Darstellung der Fallstudien

Die Falldarstellung der Arbeit der Softwaregestaltenden gliedert sich in die Kategorien Beschäftigungssystem, Kontrolle und Wissensverteilung.

8.4.2.1. INTERN1: Softwaregestaltende zwischen Fachbereich und IT-Abteilung, Matrix

In diesem Fall, in dem ein EVU dezentral eine individuelle Software gestaltet, finden die Softwaregestaltenden in der Matrixorganisation förderliche Bedingungen für ihre Arbeit vor. Die Softwaregestaltenden sind zwar in die Hierarchie der Softwareanwendung eingebunden und das Management gibt den Rahmen vor. Allerdings lässt es die Softwaregestaltenden weitestgehend eigenständig arbeiten. Ob sie eigene Karrierewege verfolgen können und immer zu ihrer Expertise passende Aufgaben bekommen, ist noch offen. Wie für eine Matrixorganisation typisch, arbeiten Softwaregestaltende temporär in Projekten und die Karrierewege entsprechen einem EVU, das primär auf die Softwareanwendung ausgerichtet ist. Dafür konnte das EVU trotz Matrixorganisation dafür sorgen, dass die Softwaregestaltenden sich auf die Übersetzungsarbeit von fachlichen Bedarfen und Anforderungen konzentrieren und sich in der Praxis interdisziplinäres Wissen aneignen, wobei sie die Abteilungsgrenzen dabei nicht behindern. Dafür ist hilfreich, dass die Softwaregestaltenden und Programmierenden selbst über energiewirtschaftliches Wissen verfügen und längerfristig zusammenarbeiten.

Das **Beschäftigungssystem** richtet sich insofern an die Softwaregestaltenden aus, als sie nicht in einer Abteilung feststecken. Sie sind nur temporär zu Softwaregestaltungsprojekten wie der mobilen App für Monteur:innen zugeteilt oder haben parallel noch andere Projekte. Der befragte Teamleiter aus der IT gesteht, dass sie überlegen, wie die Softwaregestaltenden zwischen verschiedenen Projekten wechseln können, damit den intrinsisch motivierten Leuten nicht langweilig wird. Dazu würde gehören, die Karrierewege anzupassen. Eine Karriere in der Hierarchie interessiert die befragte Anforderungsmanagerin nicht. Sie will keine Führungskraft werden, sondern ist inhaltlich motiviert.

> »Wir arbeiten ja, wenn wir was Größeres haben, in einer Projektstruktur und dann kann man da für das Projekt halt Projektleiter werden, Projektverantwortliche. Sowas könnte ich mir eher vorstellen, weil es eine inhaltliche Arbeit ist. […] Disziplinarisch, das nächste, was ich machen könnte, wäre Teamleiter. Hätte ich gar keine Lust drauf.« (Anforderungsmanagerin)

In der Matrixorganisation arbeiten die Softwaregestaltenden eigenständig. Bei der **Kontrolle** legt die Hierarchie bzw. das Management weder konkrete einzelne Arbeitsschritte fest noch die Arbeitsgeschwindigkeit. Das zeigt sich daran, dass sie die digitalen Werkzeuge vor allem zur Koordination und Kommunikation verwenden. Das Management entscheidet über die Ressourcen wie IT-Budget und Stellenvergabe und welche Softwaregestaltungsprojekte das EVU macht. Die Führungskräfte kontrollieren nur die Ergebnisse oder wenn etwas im Prozessablauf auffällt:

> »Also, ich glaube, solange ich liefere, interessiert es keinen, wann ich was mach. Also, ich fühle mich nicht kontrolliert, nee.« (Anforderungsmanagerin)

Das führt dazu, dass die Softwaregestaltung als eigenständige Arbeit erlebt wird und einer persönlichen Identifikation mit der Arbeit Raum lässt:

> »Also das, was ich mache, ist eigentlich Handwerk und das ist das Schöne daran. Ich habe ein Ergebnis.« (Anforderungsmanagerin)

Nur weil die Softwaregestaltung Teil der Hierarchie eines EVU ist, konzentriert sich das **Wissen** über sie nicht beim Management. Vielmehr verteilt sich das Wissen auf mehrere Abteilungen und auf jene, die (temporär) an ihr mitwirken. Dabei kann in dem Fall trotz der Matrixorganisation das EVU einen Arbeitsprozess der Softwaregestaltung etablieren, der es den Softwaregestaltenden erlaubt, sich auf die Übersetzung zwischen Anwendung und Entwicklung zu konzentrieren, d.h. darauf, Anforderungen zu sammeln und zu schreiben. Dabei kooperieren die Anwendenden, auch wenn die Softwaregestaltenden ihnen nicht disziplinarisch vorgesetzt sind, und die Softwaregestaltenden sammeln interdisziplinäres Wissen an:

> »Also, ich kannte die agile Arbeitsweise und die ganzen Methoden nicht. Das habe ich alles gelernt. Ich musste sehr viel SAP lernen. […] Also, ich könnt immer noch nichts entwickeln. Ich bin kein Entwickler. Aber wenn ich einen Fehler sehe, weiß ich, in welche Ecke ich den wahrscheinlich schieben muss. Wen ich brauche, welchen Entwickler: Frontend, Backend und welche Richtung dort.« (Anforderungsmanagerin)

Dazu gehört, dass die anderen Beteiligten auch über interdisziplinäres Wissen verfügen: Die vier Product Owner:innen waren vorher im Fachbereich und die Programmierenden arbeiten bereits länger für den Fachbereich.

8.4.2.2. INTERN2: Softwaregestaltende in mehreren Fachbereichen, Matrix

Wie bei INTERN1 sind die Arbeitsbedingungen für die Softwaregestaltenden im EVU in der Matrixorganisation insgesamt förderlich. Zwar gibt es einerseits in dem Fall bereits teilweise eigene Karrierewege für Softwaregestaltende. Andererseits nennen Befragte das Management explizit als hinderlich für die eigene Arbeit. Denn es verfolgt keine abteilungsübergreifende Perspektive, was für die Softwaregestaltenden hilfreich wäre. Dafür können die Softwaregestaltenden ungehindert von Abteilungsgrenzen im horizontal verteilten Gestaltungsnetzwerk zwischen den Fachbereichen das notwendige Wissen sammeln und in Anforderungen übersetzen.

Was das **Beschäftigungssystem** anbelangt, fällt in dieser Fallstudie auf, dass es im Gegensatz zu INTERN1 teilweise neue Karrierewege gibt und gleichzeitig alte Muster fortbestehen. Denn das EVU bietet einzelnen Softwaregestaltenden bereits die Möglichkeit, eine für sie reizvolle Karriere zu verfolgen. Derzeit diskutiert das EVU, wie Scrum Master:innen oder Product Owner:innen aufsteigen können. In der Zentralbereichs-IT gibt es bereits die Möglichkeit, eine Karriere als Fachexpert:in ohne personelle Verantwortung zu machen und auf diesem Wege in eine höhere Vergütungsgruppe zu kommen. Andererseits hat der befragte Product Owner aus der Fachbereichs-IT seine Rolle schon länger inne und ist in sie hineingewachsen, ohne dass sich dies in der formalen Organisation niederschlägt (z.B. in einer neuen Stellenbezeichnung). Er hat nicht die althergebrachte Karriere gemacht vom Teammitglied zum Teamleiter. Der Scrum Master strebt selbst als nächsten Karriereschritt die Leitung eines internen Teams inklusive

Personalverantwortung an. Für ihn ist eine Karriere im Netzwerk, d.h. sich fachlich weiterzuentwickeln, nur zweite Wahl.

Obwohl wie bei INTERN1 die Softwaregestaltung Teil einer Hierarchie ist, sind die Softwaregestaltenden keiner direkten **Kontrolle** der Führungskräfte ausgesetzt und arbeiten selbstständig. Wie auch bei INTERN1 steht die Ergebniskontrolle im Vordergrund – ob beim Product Owner, Scrum Master oder der Anforderungsmanagerin. Inhaltlich spielt die Führungskraft des Scrum Masters keine Rolle. Er setzt aber Ziele fest.

»Und sorgt eigentlich dafür, dass es mir gut geht, sage ich mal so.« (Scrum Master)

Teilweise verhindern aber bestimmte Führungskräfte z.B. aus dem Fachbereich einen offenen Austausch. Übergreifende Entscheidungen innerhalb eines Fachbereichs oder zwischen den Fachbereichen fallen schwer. Sie müssen sich z.B. darüber einigen, wie die Anforderungsrunde abläuft oder wie sich das IT-Budget verteilt.

Was das **Wissen** anbelangt, ist für die softwaretechnische Interdisziplinarität die Matrixorganisation ein Nachteil. Denn in diesem Fall verteilt sich das Wissen für Softwareanwendung, -gestaltung und -programmierung auf mehrere Fachbereiche und die Softwaregestaltenden müssen in einer Organisation agieren, die primär auf die Softwareanwendung ausgerichtet ist, und entsprechend haben u.a. die Führungskräfte kein Softwaregestaltungswissen. Doch wie auch bei INTERN1 können die Softwaregestaltenden durch den kontinuierlichen Austausch im Gestaltungsnetzwerk Wissen austauschen – fast wie in einem reinen Netzwerk. Weder Programmierende noch Anwendende halten ihr Wissen zurück. So lernen die Beteiligten voneinander und alle bauen mit der Zeit interdisziplinäres Wissen auf. Dabei sagen Scrum Master, Product Owner und Anforderungsmanagerin gleichermaßen, dass für ihre Arbeit das Wissen, wie Software gestaltet wird, wichtiger ist als Fach- oder ERP-Wissen. Die Anforderungsmanagerin meint allerdings, dass man ohne energiewirtschaftliches Wissen langsamer ist und vieles nicht sofort versteht. Das Wissen lernt sie jetzt mit der Zeit in der Praxis und durch einen Kollegen, der mehr Erfahrung hat.

Die Softwaregestaltenden kompensieren untereinander die Nachteile einer Organisation, die primär auf die Softwareanwendung ausgerichtet ist, indem sie sich EVU-weit vernetzen. Der Product Owner aus dem Fachbereich tauscht sich in einem Expert:innenkreis mit Mitarbeitenden anderer Netzbetriebe aus, die eine ähnliche Tätigkeit wie er haben. Im IT-Bereich gibt es einen firmenweiten, organisierten Austausch für Scrum Master:innen und sogenannte Gilden für Software-Architekt:innen.

8.4.2.3. KOOP1: IT-DL als Heimat der Softwaregestaltenden, Matrix

Im Vergleich zu INTERN1 und INTERN2 ist es für die Arbeit der Softwaregestaltenden von Vorteil, dass das IT-DL auf die Softwaregestaltung spezialisiert ist. Entsprechend bietet das IT-DL die gewünschten Karrierewege, kann für die Auslastung der Softwaregestaltenden sorgen und eine stetige softwaretechnische Interdisziplinarität garantieren. In dem Fall zeigt sich, dass für die Softwaregestaltenden ein Arbeitsmarkt existiert, den sie für eine fachliche Karriere unabhängig von internen Hierarchien nützen können. Der Nachteil ist in diesem Fall die Marktbeziehung zwischen IT-DL und EVU. Das führt dazu, dass die Kundschaft (EVU) des IT-DL die Arbeit der Softwaregestaltenden mitkon-

trolliert. Zudem führt es zu einer Abhängigkeit, was Wissen anbelangt: Einerseits fehlt den meisten EVU das Wissen, um selbst die Möglichkeiten der Softwaregestaltung einschätzen zu können. Andererseits haben jene EVU, die selbst intern Softwaregestaltende beschäftigen und mit dem IT-DL langfristig zusammenarbeiten, eine Ansprechperson, mit der sie eine gemeinsame Wissensbasis teilen.

Was das **Beschäftigungssystem** anbelangt, zeigt sich in dem Fall statt einer fachlichen Karriere innerhalb einer Hierarchie, dass für Softwaregestaltende eine fachliche Karriere im Markt möglich ist. Das IT-DL ist Teil dieses Marktes für IT-Fachkräfte. Es gibt Mitarbeitende, für die ist das IT-DL nur eine Durchgangsstation in ihrem beruflichen Werdegang. Vor allem die Spezialisierung auf die ERP-Software von SAP bietet die Möglichkeit, sich für eine Vielzahl anderer Firmen zu qualifizieren. Für das IT-DL bedeutet das laut einem Befragten eine stetige Personalfluktuation, was aus seiner Sicht neue Impulse bringt und dafür sorgt, dass das IT-DL nicht überaltert.

Für die Softwaregestaltenden ist auch eine Karriere innerhalb des IT-DL möglich. Zudem ist für sie der Vorteil einer zentralisierten Softwaregestaltung in einem IT-DL, dass sie für mehrere EVU tätig werden können. Dadurch ist für sie in diesem Fall einfacher möglich, was sie sich auch schon bei INTERN1 und INTERN2 gewünscht haben: interessante Aufgaben und Projekte zu bearbeiten. Vor allem für junge Mitarbeitende sind innovative Projekte ein stärkerer Anreiz als der Aufstieg zur Teamleitung. Der Anforderungsmanager des IT-DL meinte, dass ihn die Aufgabe motiviert, nicht die Aussicht auf eine Karriere. Auch in den EVU haben die Softwaregestaltenden solche Wünsche. In EVU2 gibt es bereits zwei Karrierepfade: eine Führungskraft-Karriere und Fachkarriere. Im EVU3 spricht der Prozessmanager davon, dass er eine fachliche Karriere und keine disziplinarische machen will.

Inwiefern ist die Matrixorganisation für die **Kontrolle** der Softwaregestaltenden wichtig oder stört sie sogar bei ihrer Arbeit? Es überrascht, dass auch in diesem Fall, wie schon bei INTERN1 und INTERN2, trotz Markt und Hierarchien die Führungskräfte sowohl bei den EVU als auch dem IT-DL nur wenn notwendig direkt kontrollieren. Auch wenn die Befragten aus den EVU alle unterschiedliche Positionen haben und daher unterschiedliche Führungskräfte, berichten alle davon (ob Anwendungsbetreuer, Anforderungsmanager oder Prozessmanager), dass sie wenig bis nur in Ausnahmefällen Kontakt mit ihrer Führungskraft haben. Sie können weitestgehend selbstständig arbeiten. Die Selbständigkeit zeigt sich auch an der Arbeitsbelastung und den Zielen. Die Arbeitsbelastung ist individuell unterschiedlich und hängt von Vorgaben des Managements ab. Ein anderer Befragter meint, er kommt durch Selbstorganisation gut hin und hat keinen Zeitdruck.

> »Es ist so bei uns, dass wir unsere Arbeitsbelastung gut selbst managen können.« (Applikationsbetreuer EVU2)

Von welchen anderen Umständen die Arbeitsbelastung abhängt, ist unklar. Beide befragten Prozessmanager (EVU3, EVU1) sprechen von einer hohen Arbeitsbelastung. Beim IT-DL müssen zwar die Softwaregestaltenden bei mehreren Projekten oder Tickets für mehrere EVU mitarbeiten. Jedoch hat diese keine direkten Folgen für die Arbeitsbelastung des einzelnen Beschäftigten. Das liegt z.B. daran, dass das Projektmanagement-

Team des IT-DL Projektarbeit selbstorganisiert verteilt. Jede:r muss sich selbst fragen, was er oder sie noch schafft.

Auch vorgegebene Ziele führen nicht automatisch dazu, dass Führungskräfte das Handeln der Beschäftigten einschränken. Der IT-Projektleiter des EVU sieht sich durch das IT-Budget wenig eingeschränkt. Auch in einem anderen EVU gibt es »relativ viel« (Prozessmanager EVU3) Budget und es wird zur Verfügung gestellt und spielt keine große Rolle. Der Applikationsbetreuer (EVU2) schließt die Zielvereinbarung jährlich mit seiner Führungskraft ab (woran auch sein Gehalt hängt). Auch der Prozessmanager (EVU1) hat individuelle Ziele, die aber seiner Meinung nach nicht so wichtig sind für sein Gehalt. Beim IT-DL erarbeiten Führungskräfte ergänzend zum Arbeitsvertrag mit den einzelnen Angestellten Ziele im Dialog. Das Unternehmen selbst hat Ziele, was Umsatz, Projekte und Qualität anbelangt. Diese spielen aber laut den Befragten bei der Arbeitsbelastung keine Rolle. Der Anforderungsmanager des IT-DL setzt sich die intrinsischen Ziele selbstbestimmt, wie z.B. defizitäre Projekte zu vermeiden oder besser zu werden. Die Ziele der Firma sind für ihn sekundär. Dazu kommt, dass der Bonus, den das IT-DL ausschüttet, für alle Mitarbeitenden gleich ist.

Was die **Wissensverteilung** anbelangt, können sich die Softwaregestaltenden trotz Hierarchien innerhalb der EVU austauschen und trotz Marktbeziehungen existieren längerfristige Beziehungen zwischen EVU und IT-DL, wodurch eine gemeinsame Wissensbasis nicht nur temporär besteht. Durch das vom IT-DL kooperativ organisierte Anforderungsmanagement bestehen Gestaltungsnetzwerke in die EVU hinein und damit eine gemeinsame Wissensbasis bei den Beteiligten. Allerdings ist die Beziehung nicht gesichert, denn das IT-DL steht im Wettbewerb mit anderen. Wenn die Wissensbasis fehlt, fällt das schnell auf, so z.B., wenn Mitarbeitende des IT-DL noch nicht so lange für die Energiewirtschaft tätig sind:

> »[E]s gibt da teilweise Kollegen, die da neu sind bei dem Dienstleister und die machen da irgendwas, was nicht richtig ist, weil die aber einfach die Gegebenheiten, diese stadtwerkespezifischen Gegebenheiten nicht kennen« (Anforderungsmanager EVU2).

Manche EVU sind abhängig vom IT-DL, weil sie selbst kein Know-how in der Softwaregestaltung haben. Manche bauen erst wieder Kompetenzen auf oder sammeln durch eigene Softwaregestaltung interdisziplinäres Wissen. Es gibt in den EVU Mitarbeitende wie Prozessmanager, die sowohl fachliches als auch softwaretechnisches Wissen mitbringen. Die befragte Prozessmanagerin des EVU1 schätzt bei ihr das Verhältnis von IT- zu Fachwissen mit 30:70 ein. Der Vorteil ist in diesem Fall, dass innerhalb des IT-DL die Interdisziplinarität tägliche Praxis ist. Die befragten Programmierenden sehen sich weniger als Teil ihrer Programmierenden-Teams als vielmehr von interdisziplinären Teams, die sich für Projekte oder einzelne Anforderungen zu spezifischen Fachthemen zusammenfinden. Es gibt IT-Beratende und Programmierende, die nur für ein EVU arbeiten und sich dadurch besser mit dessen System auskennen. In einem Fall ist der Ansprechpartner des IT-DL Programmierer und Berater zugleich, der bei der Einführung der Lösung und an ihrer individuellen Anpassung beteiligt war. Er verfügt damit über ein umfassendes fachliches und softwaretechnisches Wissen.

Schulungen dienen als Einstieg. Die einzelnen Mitarbeitenden bauen ihr Wissen über die Jahre auf und verfügen über individuelle Wissensbestände und Spezialisierung – je nachdem, an welchen Projekten sie teilgenommen haben und in welchem Fachbereich sie tätig sind.

8.4.2.4. KOOP2: Softwaregestaltende in fremdem Umfeld der EVU, Matrix

Wie im Fall von KOOP1 hat hier das IT-DL den Vorteil, sich auf die Softwaregestaltung spezialisieren zu können. Entsprechend finden die Softwaregestaltenden dort andere Karrierewege, eine von der Softwareanwendung unabhängige Kontrolle und eine Spezialisierung auf interdisziplinäres Wissen vor. Jedoch haben einzelne EVU in diesem Fall die Softwaregestaltung wieder verstärkt selbst übernommen. Diese EVU können weder passende Karrierewege noch eine stetige Auslastung oder immer interessante Projekte für die Softwaregestaltenden gewährleisten. Stärker als in anderen Fällen zeigt sich das Problem von einer Wissensverteilung in Hierarchien und Marktbeziehungen. Die Gestaltungsnetzwerke der Softwaregestaltenden sind prekär und so fehlt manchmal die gemeinsame Wissensbasis. Dafür sind die Softwaregestaltenden in den EVU unabhängig vom IT-DL, was das notwendige Wissen zur Softwaregestaltung anbelangt.

Das Besondere an dem Fall ist, dass aufgrund der ursprünglichen Spezialisierung des IT-DL auf die Softwaregestaltung dieser sich hinsichtlich **Beschäftigungssystem** auf diese ausgerichtet hat. Ähnlich wie in der Fallstudie KOOP1 nimmt nach Aussage des befragten Bereichsleiters im IT-DL die Bedeutung von Karriereleitern ab, und die Mitarbeitenden wollen weniger Führungskraft werden als vielmehr interessante Projekte machen. Das Unternehmen versucht seit mehreren Jahren neben der Führungslaufbahn neue Karrieremodelle zu entwickeln. Die Idee einer Fachexpert:innen- und Projektmanagementlaufbahn hat das IT-DL aufgegeben. Es gibt nun Kompetenzlevel mit entsprechenden Gehaltsbändern, bei denen Mitarbeitende durch Erfüllen bestimmter Anforderungen fachlich und nicht hierarchisch aufsteigen können. Was den Wechsel zwischen Organisationen anbelangt: Wie auch bei KOOP1 wechseln IT-Fachkräfte wie IT-Beratende und Programmierende eher zu Wettbewerbern und nicht innerhalb des organisationalen Netzwerks aus EVU und IT-DL. Die fachliche Karriere im Markt – inklusive der Neuverhandlung des Gehalts – ist damit wie schon bei KOOP1 eine Alternative zu jener innerhalb einer Organisation.

Ein EVU2 der Fallstudie hat zwar wieder die Softwaregestaltung übernommen. Aber es zeigt sich, dass es dabei zu Konflikten mit den bestehenden Beschäftigungssystemen kommt. Denn dort dominiert die Karriere in der Hierarchie. Die Befragten – IT-Koordinator und Projekt Coach – haben das EVU2 einige Monate nach dem Interview verlassen. Das lag nicht daran, dass sie keine hierarchischen Karrieren machen können. Vielmehr gründet es in der fehlenden Perspektive, dass das EVU sich zu einer agileren Organisation mit flachen Hierarchien und interessanten Projekten wandeln würde.

Was die **Kontrolle** der Arbeit der Softwaregestaltenden anbelangt, gibt es Unterschiede zwischen IT-DL und EVU. Das Beispiel eines IT-Beratenden beim IT-DL zeigt exemplarisch, wie sich selbstständiges Arbeiten mit Markt und Hierarchien mischt. Der befragte Teamleiter als Führungskraft des IT-Beratenden kümmert sich um Budget, Verträge, neue Projekte, Entscheidungen zu fachübergreifenden Themen und Personalführung. Er fordert eigenständiges Arbeiten ein: stärkere Lernbereitschaft

und Eigeninitiative. Beides sieht er als festen Bestandteil der Arbeit an. Mitarbeitende sind gefragt, die bereit sind, sich auch über eigene Themengebiete hinaus jenseits von Schulungen selbstständig in neue Themen einzuarbeiten. Sie sollen Unternehmer im Unternehmer werden:

> »Habe es aber geschafft, meine Philosophie ein bisschen durchzusetzen. Also, das, was ich bei [große Beratungsfirma] erfahren habe, gelernt habe: dass der Berater der Unternehmen ein Unternehmer im Unternehmen ist und für sich selbst verantwortlich ist und man ihm durchaus genügend Verantwortung und Kompetenz geben kann, auch beim Kunden vor Ort entsprechend seine Aufgaben zu übernehmen. Das hat in den letzten Jahren eigentlich ganz gut gefruchtet.« (Teamleiter IT-DL)

Der IT-Berater gibt an, wenig Kontakt zu seiner Führungskraft zu haben, und er arbeitet die Aufgaben ihrer Priorisierung gemäß in seinem eigenen Tempo der Reihe nach ab (ohne feste Vorgaben für die Anzahl der umgesetzten Anforderungen o. ä.). Jedoch spielen neben der Führungskraft für die Mitarbeitenden des IT-DL Markt-Kennzahlen eine Rolle. Umsatzziele fließen in die Faktura-Ziele der Beratenden ein, d.h., sie müssen eine bestimmte Anzahl an Tagen bei einem EVU sein und damit Umsatz für das IT-DL generieren.

Innerhalb der EVU nehmen einzelne Befragte ihre Führungskräfte als hinderlich oder gar überflüssig für die Softwaregestaltung wahr. Der befragte IT-Koordinator stellt seine Führungskraft in Frage, da er alles eigenständig erledigt. Andere Führungskräfte entscheiden über das Vorgehen bei der Softwareanpassung. Was an dem Fall auffällt: In einer Matrixorganisation hängen die Softwaregestaltenden davon ab, dass Führungskräfte sie unterstützen. Führungskräfte koordinieren Projekte, sprechen bei Problemen mit dem IT-DL. Sie fungieren als Netzwerkende, die Beteiligte eines Projekts kennen und helfen, sich innerhalb der Firma durchzusetzen. Sie werden bei Eskalationen aktiv. Wie auch schon in anderen Fallstudien haben Softwaregestaltende Freiheiten bei der Zielvereinbarung. Der IT-Koordinator aus EVU2 verhandelt mit seiner Führungskraft, was die Ziele sind:

> »Ich sag auch meinem Chef ganz klar: ›Ja, kann ich noch machen‹. Oder: ›Nee, das passt jetzt nicht, weil, ich möchte noch die anderen Themen auch sauber zu Ende bringen.‹ Und das ist auch etwas, was ich halt lernen habe müssen. Und ich glaube, das muss jeder irgendwann lernen, seine eigenen Grenzen und halt eben auch zu sagen: Was ist mir wichtig? Ist mir wichtig, nur Quantität rauszuballern? Und mir ist Qualität ganz wichtig.« (IT-Koordinator Fachbereich EVU2)

Die Matrixorganisation zeigt sich in der **Wissensverteilung** vor allem innerhalb der EVU. Dort besteht eine starke Trennung zur Softwareanwendung und -programmierung. Erst mit der Zeit, oftmals temporär und abhängig von Einzelpersonen oder einzelnen Fachbereichen, entstehen a) Gestaltungsnetzwerke zwischen Softwareanwendung, -gestaltung und -programmierung und b) entwickeln Führungskräfte zumindest ein grundlegendes Verständnis für die Softwareentwicklung.

Bei einer auf die Softwareanwendung ausgerichteten Organisation wie EVU2 ist erst einmal keine gemeinsame, interdisziplinäre Wissensbasis für die Softwaregestaltung vorhanden. Das zeigt sich an Führungskräften, die kein Wissen über die Softwaregestaltung haben, aber haben sollten, weil sie in Matrixorganisationen über den Wissensaustausch entscheiden können. Das zeigt sich an einem Softwaregestaltungsprojekt in EVU2, bei dem es an einer gemeinsamen, interdisziplinären Wissensbasis fehlt, z.B. bei der Zusammenarbeit zwischen IT-Abteilung und Fachbereichen. Es sind neue, für die Softwaregestaltung geschaffene Rollen wie die IT-Koordinierenden der Fachbereiche, die sich teamübergreifende Kompetenzen aneignen können. Sie sind für die Softwaregestaltung in mehreren Teams zuständig und können so interdisziplinäre Zusammenhänge verstehen lernen. Im Team Marktkommunikation des EVU2 ist die Zusammenarbeit mit dem zuliefernden Softwareunternehmen eingespielt und die Teamleitung und einzelne ihrer Mitarbeitenden haben das entsprechende Wissen zur Softwaregestaltung. Hier zeigt sich, dass es in einer Matrixorganisation von Vorteil ist, wenn die Führungskraft selbst mitgestaltet oder weiß, wie Softwaregestaltung funktioniert.

EVU3 hat eine andere Variante, um in einer Matrixorganisation für Softwaregestaltende eine Wissensbasis zu schaffen, damit sie mit Anwendenden und Programmierenden zusammenarbeiten können: Das EVU hat zentral ein Team etabliert, das selbst Wissen durch Softwaregestaltungsprojekte in unterschiedlichen Fachbereichen aufbaut. Zugleich sorgt es dafür, dass die Matrixorganisation interdisziplinäres Gestaltungswissen in und zwischen den Fachbereichen aufbaut.

Eine Basis interdisziplinären Wissens ist fest in der Hierarchie des IT-DL verankert. Der befragte Teamleiter war selber einmal IT-Berater und hat fachlich und softwaretechnisch tiefergehendes Wissen. Der IT-Berater kann programmieren und kennt sich in einzelnen Prozessen der Energiewirtschaft aus. Insgesamt hat das IT-DL sowohl softwaretechnisches als auch fachliches Wissen. Es kennt die ERP-Firma SAP und deren Software sowie die EVU sehr gut. Zudem hat es aufgrund der BPO-Dienstleistungen selbst Anwendende der Software im Haus.

8.4.2.5. PAKET: Softwaregestaltende einer Standardsoftware, Matrix

In diesem Fall, in dem eine Softwarefirma zentral eine Standardsoftware gestaltet, finden die Softwaregestaltenden nur bedingt förderliche Bedingungen für ihre Arbeit vor. Die Softwaregestaltenden innerhalb der Softwarefirma sind getrennt von den vielen Anwendenden in den EVU. Die Softwaregestaltenden in den EVU, welche Einstellungen am Standard vornehmen, sind vom Standard abhängig, den die Softwarefirma ausliefert. Aber ob innerhalb der Softwarefirma oder der EVU: Die Softwaregestaltenden sind größtenteils fest einem fachlich spezialisierten Bereich wie Energiedatenmanagement oder Marktkommunikation zugeordnet und arbeiten entsprechend kontinuierlich interdisziplinär. Anders als für SAP gibt es in diesem Fall keinen eigenen Arbeitsmarkt für IT-Beratende der industriespezifischen ERP-Software und damit weniger Karrieremöglichkeiten für die Softwaregestaltenden. Innerhalb der EVU sind Softwaregestaltende höhergestellt als die Anwendenden, aber Führungskräften untergeordnet. In dem Fall fällt stärker auf als bei KOOP1 und KOOP2, dass aufgrund der Marktbeziehung die Wissensverteilung zwischen EVU und IT-DL zu einer Abhängigkeit führt. Die EVU müssen sich eigenständig über Neuerungen an der Software nach Updates und über die vorhandenen

Einstellungsmöglichkeiten am Standard informieren. Der Wissensaustausch zwischen EVU und Softwarefirma ist weniger problematisch, wenn beide aus einer Kooperation in puncto Softwaregestaltung Vorteil ziehen. So geschieht dies z.B., wenn ausgewählte EVU für die Gestaltung des Standards Anforderungen liefern, bewerten oder testen und damit den Standard bereits kennen, bevor ihn die Softwarefirma ausliefert.

Das **Beschäftigungssystem** der Softwaregestaltenden ist davon geprägt, dass sie in hierarchischen Organisationen angestellt sind. In den EVU gibt es für sie keine eigenständigen Kompetenzkarrieren. Sie machen Karrieren in der Hierarchie, und je nach EVU oder Team kann die Softwaregestaltung Teil einer Führungsposition oder ganz ausgelagert sein, z.B. an IT-Beratende. Was auffällt, ist, dass in einigen EVU die Team- und Gruppenleitung Software gestaltet, d.h., sie nimmt Einstellungen an der Software vor oder gibt Fehler-Tickets auf. Selbstständiges Arbeiten und IT-Affinität sind von Vorteil, wenn man Karriere machen will.

> »Aber im Wesentlichen ist mein Anreiz sozusagen: Ich interessiere mich für Prozesse. Ich interessiere mich für Prozessoptimierung und habe halt sozusagen auch eine hohe IT-Affinität. Und das ist sozusagen ein bisschen auch mein eigener Antrieb für diese Position, in der ich jetzt bin.« (Gruppenleiter Abrechnung EVU5)

Softwaregestaltende der Softwarefirma wie IT-Beratende oder Lösungsarchitekten können für unterschiedliche EVU arbeiten, aber auch Softwaregestaltungsprojekte innerhalb der Softwarefirma durchführen. Aber anders als bei den ersten vier Fallstudien, die SAP verwenden, sind hier die IT-Beratenden ausschließlich in der Softwarefirma angestellt. Sie haben als Arbeitsmarkt für eine mögliche weitere fachliche Karriere nur die EVU, welche das Standardpaket anwenden, und keine große Anzahl an kooperierenden Firmen wie bei SAP.

Bei der **Kontrolle** der Softwaregestaltenden ist wie auch in den anderen Fallstudien die Erfolgskontrolle wesentlich. Die Führungskräfte erwarten selbstständiges Arbeiten und eigenständiges Lernen. Zugleich sind Softwaregestaltende entweder in Hierarchien eingebunden oder in Marktbeziehungen mit den entsprechenden Folgen. Letzteres bedeutet, dass z.B. SLA deren Arbeit mitkontrollieren. Innerhalb der Hierarchien setzen Führungskräfte den Rahmen für die Arbeit. Das gilt für Key User:innen oder Anwendungsbetreuende innerhalb der EVU, Lösungsarchitekten oder IT-Beratende in der Softwarefirma.

Eigenständiges Arbeiten und reine Erfolgskontrolle durch Führungskräfte kann zu einer extremen Belastung werden. Das ist vor allem dann der Fall, wenn eine Person sowohl das Tagesgeschäft der Softwareanwendung als auch der Softwaregestaltung übernehmen muss, zusätzlich einem Termindruck ausgesetzt ist und keinen Einfluss auf die Softwarefirma und deren Softwaregestaltung hat. So kann sich ein befragter Gruppenleiter mit der Softwaregestaltung nur nebenher beschäftigen und es gibt keine Person in seiner Kollegenschaft, die ihm die Softwaregestaltung abnimmt und sich mit der Softwarefirma austauscht. Er muss sowohl Anwendungsarbeiten erledigen als auch intern die Rolle als Anwendungsbetreuer übernehmen. Weil über einen längeren Zeitraum eine Kombination von Termindruck durch die Regulierung und stetige Updates der Software hinzukam, ist er fast ein Jahr aufgrund von Burnout ausgefallen.

Was die **Wissensverteilung** anbelangt, fällt bei diesem Fall besonders auf, dass die Softwaregestaltenden der EVU, die Einstellungen an der Standardsoftware vornehmen, nicht Teil des Praxisnetzwerks sind, das die Standardsoftware gestaltet. Mit der Folge, dass sie erst an dem fertigen Softwareprodukt über Neuerungen etwas lernen können. Das Learning by Doing und damit der Aufbau von Erfahrungswissen geschieht erst in der Anwendung der Standardauslieferung:

> »Wir müssen es aber dann im […] täglichen Geschäft praktisch beginnen, die Prozesse im Kern zu verstehen, und deswegen ist da Learning by Doing ganz häufig so, dass man dann die Fälle im System sieht und autodidaktisch versucht, sich die Sachen so zurechtzulegen, wie sie denn sein sollen und dann darüber den nötigen Erkenntnisgewinn zu bekommen und Worst Case hat man es gerade verstanden und dann kommt das nächste Update und es ist schon wieder hinfällig. Also, da ist man eigentlich im permanenten Sprint hinterherzukommen.« (Leiter EVU2)

Wobei in manchen EVU die Softwaregestaltenden privilegierten Zugang zu dem Wissen haben und Marktbeziehungen und Hierarchien keine Hürden sind. Dann sind aber die Softwareanwendenden noch nicht über Änderungen informiert, was zeigt, wie Hierarchien den Wissensaustausch im Team bestimmen:

> »Also, die Anwendungsbetreuer sind viel intensiver im Austausch mit dem Softwarehersteller und dadurch ›persönlich betreut‹ in Anführungszeichen. Die Mitarbeitenden, die operativ tätig sind und die Prozesse vollziehen, sind diejenigen, die es sich selbst beibringen aktuell. Und das ist eigentlich eine Reihenfolge, die ist eigentlich falsch.« (Betriebsrat EVU6)

Weil die Softwareanwendenden nicht Teil eines Gestaltungsnetzwerkes sind, hängt es von ihrer Eigenmotivation ab, sich mit der Software auseinanderzusetzen. Doch fällt es der Führungskraft eines EVU schwer, die Mitarbeitenden dazu zu bringen, eigenständig zu lernen.

> »Es ist schon so, dass die Kollegen ambitionierter sein müssten, was ihr Arbeitsumfeld angeht. Also, sie müssten schon nach links und rechts gucken, wie man das so schön sagt oder versuchen eigenständig sich Dokumentationen anzulesen. Es gibt von der [Softwarefirma] unzählige Dokumente, die man sich durchlesen kann, und dann versteht man natürlich das Programm auch anders. Und da ist natürlich der Kollege, der sagt: Nö, damit möchte ich mich nicht beschäftigen. Es muss mir jemand erklären. Und wenn mir das niemand erklärt, dann…« (Gruppenleiter Anwendungsbetreuung EVU3)

Aus Sicht des befragten Gruppenleiters wären noch mehr IT-affine Leute hilfreich, die Anwendungsbetreuung machen und Einstellungen an der Software vornehmen können.

Die Softwaregestaltenden innerhalb der Softwarefirma arbeiten kontinuierlich interdisziplinär. Wobei das Wissen über die Branche wichtiger ist als jenes über die Anwendung in einzelnen EVU. Dabei sind Führungskräfte anders als in anderen Fallstudien eng eingebunden. Es zeigt sich der Vorteil, wenn Führungskräfte auch Wissen über

Softwaregestaltung haben. Der Teamleiter eines befragten Programmierers hat tieferes fachliches Wissen über die Energiewirtschaft:

> »[D]er fachliche Teamleiter, […] der sich um die Prozesse kümmert, das Fachliche im Kopf hat und uns dann genauer sagen kann: Okay, an der Stelle musst du das und das machen. Oder wenn man halt mal Fragen hat, wenn man mal den Prozess nicht so 100 % kennt: Hier, wie ist das in den und den Spezialfällen?« (Programmierer)

Der befragte Programmierer selbst meint, bei ihm ist das Verhältnis 40 % Wissen über die Energiewirtschaft und 60 % über die Programmierung. Das Problem an der Marktbeziehung ist für ihn, dass Programmierende schwerer an die Bedarfe der Anwendenden in den EVU herankommen, weil sie sehr weit weg von ihnen sind.

8.4.2.6. KOOP3: Softwaregestaltung als neues Betätigungsfeld, reines Netzwerk

In dem Fall behindert die Verteilung der Softwaregestaltung auf unterschiedliche Organisationen nicht die Arbeit der Softwaregestaltenden, was typisch für eine reine Netzwerkorganisation ist. Für die Beschäftigten stehen weniger Karrieren in den Hierarchien im Mittelpunkt. Sie können ihrer intrinsischen Motivation nachgehen und das Thema IoT für ihre jeweilige Organisation voranbringen. Die Befragten arbeiten selbstständig, weil sie IoT auch als ihr eigenes Projekt begreifen und nicht nur als eines ihrer Vorgesetzten oder ihrer Kundschaft. Dabei ist das interdisziplinäre Wissen über die Möglichkeiten der Softwaregestaltung und die Bedarfe der EVU anders als bei PAKET gleichmäßig verteilt, für die Softwaregestaltenden gut zugänglich und es besteht eine gemeinsame Wissensbasis.

Die Softwaregestaltenden arbeiten in einem **Beschäftigungssystem**, das es ihnen erlaubt, kontinuierlich an dem Thema IoT zu arbeiten. Sie arbeiten zwar flexibel an Projekten mit, jedoch immer zum gleichen Thema IoT und der gleichen IoT-Standardsoftware der Softwarefirma. Es gibt sowohl in den EVU als auch dem IT-DL neue Aufgaben und gar neue Teams für IoT. Die Beteiligten begreifen es als ihr eigenes Projekt. Die Softwaregestaltung ist in diesem Fall ein klar abgegrenztes Betätigungsfeld. Aus den Interviews geht nicht hervor, dass mit der Mitarbeit an der Softwaregestaltung – ob im EVU, beim IT-DL oder in der Softwarefirma – eine hierarchische Karriere verbunden wäre. Weniger ist der Aufstieg in der Hierarchie ein Anreiz als beim Thema IoT dazuzulernen, interessante Projekte zu machen und die Software weiterzuentwickeln.

Selbstständiges Arbeiten und damit eine indirekte **Kontrolle** sind vorherrschend. Der Fokus der Beteiligten liegt auf der Koordination untereinander, was typisch für reine Netzwerke ist. Die Beteiligten vor allem des IT-DL und der Softwarefirma sind intrinsisch motiviert und sie nutzen formelle Wege vor allem dazu, um die Arbeit zu koordinieren. Die Zusammenarbeit ist persönlich, partnerschaftlich und auf Augenhöhe. Die Führungskräfte fungieren bei den befragten EVU als Bindeglied in den Hierarchien, wie es in Matrixorganisationen für die Softwaregestaltung hilfreich ist.

Trotz Organisationsgrenzen und Marktbeziehungen ist das **Wissen** gleichmäßig im organisationalen Netzwerk verteilt. Die Softwaregestaltenden verfügen nach längerer Mitarbeit im organisationalen Netzwerk über eine gemeinsame Wissensbasis, um mitgestalten zu können. Das Besondere an dem Fall ist, dass sowohl die Softwarefirma als

auch das IT-DL und EVU1 Wissen über die Implementierung (deren Durchführung und über Funktionalitäten und Anwendungsfelder der Software) und Erweiterung (zusätzliche Module programmieren) der Software haben. Das Wissen, was in den Quellcode der IoT-Software einfließt, kommt aus Implementierungsprojekten unterschiedlicher Anwendungsfälle des IT-DL mit EVU. Wie bei den anderen Fallstudien liefert die Softwarefirma neben der Software Wissen über Änderungen aus: Die Softwarefirma stellt bei Updates Frequently Asked Questions (FAQ) als Dokumentation von Funktionalitäten und Neuerungen zur Verfügung. Das IT-DL schickt zu Updates einen Newsletter an die EVU. Das heißt, Softwarefirma und IT-DL erwarten, dass die Empfangenden, für die diese Informationen von Interesse sind, sie sich selbst aneignen. Somit spielt die Marktbeziehung doch noch insofern eine Rolle, als hier kein direkter Austausch erfolgt. Es bleibt den anwendenden Einzelnen nur, sich eigenständig einzulesen.

8.4.2.7. STARTUP: Softwaregestaltende im Kern der Organisation, reines Netzwerk

Der Fall zeigt typische Eigenschaften davon, wenn Softwaregestaltende in einem reinen Netzwerk arbeiten: Was das Beschäftigungssystem anbelangt, können Mitarbeitende in der rollenbasierten Organisation ihren Interessen nachgehen, indem sie Rollen wechseln, und es geht nicht darum, in einer Hierarchie aufzusteigen. Die Kontrolle passiert im Wesentlichen horizontal durch die Kollegenschaft und ermöglicht Eigenmotivation, individuelle Steuerung der Arbeitsbelastung und Selbstständigkeit. Durch den kontinuierlichen, iterativen Austausch im Netzwerk für die Softwaregestaltung ist eine gemeinsame Wissensbasis und ein unkomplizierter Zugang zum Wissen gesichert. Weil das Wissen nicht hierarchisch oder über einen Markt verteilt ist, bestehen keine Hindernisse für den Austausch.

Was das **Beschäftigungssystem** anbelangt, können die Beschäftigten nicht in einer formalen Hierarchie aufsteigen, sondern nur Verantwortung für zusätzliche oder andere Rollen übernehmen. Um sich darüber auszutauschen, wer welche Verantwortungsbereiche hat, gibt es regelmäßig Treffen. So geht es im STARTUP um das Arbeiten an einem gemeinsamen Ziel und den möglichen zu teilenden Mehrwert durch eine wachsende Organisation:

> »Je mehr erfolgreiche Ideen wir haben, die wir umsetzen können, desto mehr Kuchen ist halt da, der aufzuteilen ist.« (Programmierer1)

Der Primat der Softwareentwicklung zeigt sich in dem Fall daran, dass die Softwaregestaltenden Teil des stabilen Kerns der Organisation sind. In den ersten vier Fallstudien sind die Softwareanwendenden fest in den EVU angestellt und langfristige Mitglieder ihrer Teams. Sie sind dort auch schon länger. Es sind die Softwaregestaltenden, die sich flexibel in einer Matrixorganisation bewegen. Hier ist es umgekehrt, weil in dem Fall die Softwareentwicklung der Ausgangspunkt der Organisation ist. Die Anwendenden sind flexible 450-Euro-Kräfte, die weniger verdienen, Teilzeit arbeiten und nicht studiert haben. Von den anderen Beschäftigten haben alle bis auf jemanden aus dem Kommunikationsbereich einen akademischen Abschluss.

Bei der **Kontrolle** spielen Führungskräfte keine Rolle. Die Arbeit ist ein persönliches Projekt in stetiger Auseinandersetzung mit der iterativ gestalteten Software, der Kun-

den- und Kollegenschaft. Die Product Ownerin lernt viel und meint, es macht ihr Spaß. Sie schreibt Arbeitszeiten auf, um nicht zu viel zu arbeiten. Kontrolle gäbe es gar nicht. Druck entsteht durch sich verantwortlich fühlen, durch die Kundschaft und die Erwartungen an die Software.

> »Druck entsteht eher dadurch, dass ich mich für vieles verantwortlich fühle und (gerade in einem Start-up) oft das Gefühl entsteht, dass es um die Existenz geht.« (Product Ownerin)

Man erwartet Verantwortungsbereitschaft und Eigenverantwortung voneinander. Die Peer-Kontrolle ist Teil der rollenbasierten Organisation.

In dem Fall behindern weder Hierarchien noch Marktbeziehungen den Wissensaustausch der Softwaregestaltenden. Das notwendige **Wissen** zur Softwaregestaltung befindet sich innerhalb von STARTUP und ist gemeinsame Basis der Arbeit. Den Kern der Organisation bilden interdisziplinäre Kreise, in denen sich regelmäßig Fach- und IT-Wissen treffen und die Beteiligten Möglichkeiten der Softwaregestaltung mit jenen der energiewirtschaftlichen Bedarfe abgleichen. Die Softwaregestaltenden können als Kern der Organisation immer auf Programmierende oder Anwendende zugreifen. Die befragte Product Ownerin meint, sie lernt durch die Teilnahme an den Kreisen etwas über interne Abläufe und den Nutzen für die Kundschaft. Alle, die z.B. wie die Anwendenden nicht an Kreisen teilnehmen, nehmen nicht am Wissensaustausch teil. Das Wissen konzentriert sich damit bei den Mitarbeitenden, die dort mitmachen, und sie können interdisziplinäres Wissen akkumulieren unabhängig von Hierarchien und Märkten. Es ist also nicht die Hierarchie, die die Wissensverteilung bestimmt. Wenn, dann ist die Wissensverteilung die Ursache für informelle Hierarchien.

8.4.3. Zusammenfassung

8.4.3.1. Unterschiede: zwischen reiner Netzwerk- und Matrixorganisation

Die Falldarstellungen zeigen, dass sie nicht immer klar einem der diametralen Typen von Matrixorganisation und reiner Netzwerkorganisation entsprechen und sich teilweise einzelne Elemente der beiden Typen vermischen. Trotzdem vereinen sie entweder mehr Eigenschaften von dem einen oder dem anderen Typ. So gehören KOOP3 und STARTUP zum Typ reiner Netzwerkorganisationen, die auf die Softwaregestaltung ausgerichtet sind. Dort müssen die Softwaregestaltenden in ihrer Arbeit keine Kompromisse mit bestehenden Strukturen eingehen. Reine Netzwerkorganisationen stellen einen Vorteil für die wissensintensive Softwaregestaltung im Vergleich zu jenen vom Typ Matrixorganisation dar (INTERN1, INTERN2, KOOP1, KOOP2 und PAKET). Hier sei zum Abschluss des Abschnitts kurz dargelegt, warum welcher Fall welchem Typ zugeordnet werden kann.

Bei STARTUP und KOOP3 zeigt sich die Netzwerkorganisation an der Kontrolle der Softwaregestaltung, bei der die Peer- und Objekt-Kontrolle (im Sinne Rennstams, siehe 6.4.2.1) zentral sind, Softwaregestaltende unabhängig von Führungskräften agieren und der Fokus auf der Koordination der Zusammenarbeit liegt. Das Beschäftigungssystem zeichnet sich durch Kompetenzkarrieren statt durch solche in Hierarchien aus. Es

geht um Arbeit als persönliches Projekt und darum, etwas zu lernen. Statt rein sozial in Hierarchien eingeteilt und Führungskräften untergeordnet zu sein, unterscheiden sich Programmierende, Softwaregestaltende und Anwendende in ihrer Position soziotechnisch durch ihren jeweiligen Bezug zur Software. Die Softwaregestaltenden haben keine Schwierigkeiten, auf das für ihre Arbeit notwendige Wissen zuzugreifen, weil eine langfristige Praxisgemeinschaft der Softwaregestaltung mit interdisziplinärer Wissensbasis besteht und die diversen Methoden der Softwaregestaltung den Kommunikations- und Wissensfluss sicherstellen. Entweder sind die Softwaregestaltenden durch ihre Rollen und ihre Teilnahme in Kreisen Teil des interdisziplinär arbeitenden Kerns der Organisation wie bei STARTUP. Oder Markt und Hierarchien spielen im organisationsübergreifenden Netzwerk eine so geringe Rolle, dass sie für ihre Arbeit keine Hürden beim Wissensaustausch darstellen wie bei KOOP3.

Bei den anderen Fallstudien liegen Matrixorganisationen vor. Ob in IT-Projekten, Arbeitskreisen oder Scrum: Immer sind die Softwaregestaltenden Teil von Märkten und/oder Hierarchien, deren Kontrolle sie unterliegen. Es besteht eine Spannung darin, die Mitarbeitenden zu kontrollieren (ob durch Führungskräfte in Hierarchien oder Kundschaft in Marktbeziehungen) und ihnen die notwendige Freiheit für den kommunikativen Austausch und das Schreiben von Anforderungen zu geben, damit sie dabei ihre Subjektivität einbringen können, z.B. in Form intrinsischer Motivation. Die Beschäftigungssysteme für Softwaregestaltende zeichnen sich dadurch aus, dass sie Teil von Märkten oder Hierarchien sind und dadurch entsprechende Karrierechancen haben: in einer internen Hierarchie aufzusteigen oder in andere Organisationen zu wechseln. Vor allem Ersteres hat ein begrenztes Potenzial, um sich rein fachlich weiterzuentwickeln und dafür die (finanzielle) Anerkennung zu bekommen. In den Matrix-Fallstudien ist das Wissen durch Hierarchien oder Märkte getrennt. Doch stellen die Beteiligten Interdisziplinarität nicht nur temporär her. Es ist noch mehr als sonst bei der Softwaregestaltung von Vorteil, wenn die oder der Einzelne viel weiß. Dann kann er nämlich unabhängig von Hierarchien und Marktbeziehungen agieren und eine fehlende interdisziplinäre Wissensbasis bei Anwendenden und Programmierenden ausgleichen. Dabei ist es hilfreich, wenn Führungskräfte der EVU sich nicht nur als Anwendende begreifen und entsprechend auch Wissen über Software und Softwaregestaltung haben. Sie nehmen in einer Matrixorganisation zentrale Positionen ein und können die Softwaregestaltung unterstützen, z.B. indem sie eine abteilungsübergreifende Zusammenarbeit fördern.

8.4.3.2. Ergebnisse: Fallvergleich je Kategorie und Vergleich mit Anwendenden und Programmierenden

Die Falldarstellungen haben gezeigt, dass Softwaregestaltende zwar immer in organisationalen Netzwerken arbeiten, die reine Netzwerkorganisation jedoch Vorteile gegenüber der Matrixorganisation hat. Durch den Vergleich mit den Beschäftigtengruppen der Anwendenden und Programmierenden arbeitet dieser Abschnitt noch einmal heraus, was das Beschäftigungssystem, die Kontrolle und die Wissensverteilung der Softwaregestaltenden ausmacht.

Beschäftigungssystem – Kompetenzkarriere, flexible Beschäftigte und Akademiker:innen-Dominanz

Insgesamt zeigen die Fallstudien, dass sich neben einer allgemeinen Akademisierung ein Karrieresystem unabhängig von Hierarchien und der energiewirtschaftlich-fachlichen Arbeitsteilung wie z.B. Netz, Vertrieb, Handel etabliert.

Es fällt auf, dass Programmierende und Softwaregestaltende allesamt studiert haben und es bei den Anwendenden mehr die Key User:innen, Prozessmanagende oder Anwendungsbetreuende sind, die einen akademischen Abschluss haben – also eben jene, die sich mit der Softwaregestaltung befassen. Eine befragte Sachbearbeiterin braucht ein abgeschlossenes Studium, um mehr Projektarbeit machen zu dürfen. Die Bedeutung von IT-Wissen nimmt zu und hochautomatisierte und digital-integrierte Prozesse machen ein tiefergehendes und breiteres fachliches Wissen für die (Fehler-)Fallbearbeitung notwendig.

Wie die Fallstudien zeigen, haben Softwaregestaltende im Unterschied zu Programmierenden und -anwendenden ganz eigene Karrieremöglichkeiten – und zwar unabhängig davon, ob sie innerhalb eines EVU angestellt sind oder für einen IT-DL arbeiten. Die meisten befragten Softwaregestaltenden wollen keine Führungskraft werden oder in der Hierarchie aufsteigen, sondern eine Karriere als Fachexpert:innen und damit z.B. interessante Projekte machen. Vor allem in den jüngeren Organisationen, wo es um IoT oder Emissionshandel geht, ist die Arbeit ein persönliches Projekt der Beschäftigten, mit dem sie sich identifizieren. Beim IT-DL von KOOP2 gibt es bereits Kompetenzkarrieren, bei denen man durch Klettern auf definierten Kompetenzprofilen die entsprechenden Gehaltsbänder nach oben steigt. In den bestehenden Hierarchien in den EVU sind in einigen Fällen Kompetenzen in der Softwaregestaltung hilfreich für eine Führungskräftekarriere oder der Aufstieg damit verbunden, Softwaregestaltung zu koordinieren.

Eine Gemeinsamkeit zwischen Programmierenden und Gestaltenden ist, dass sie sich auf die Vermarktung ihrer Arbeitskraft konzentrieren[18], vor allem wenn sie für ein IT-DL arbeiten. Das tun sie zum einen für das IT-DL selbst, weil es Geld damit verdient, seine Fachkräfte zu verkaufen bzw. auszuleihen. Sie vermarkten ihre Arbeitskraft auch für die eigene weitere Karriere, um in anderen Organisationen mehr zu verdienen oder interessantere Projekte zu machen. Im Vergleich zu den Anwendenden haben Programmierende und Gestaltende eine bessere Bezahlung und bessere Marktchancen. Beide Gruppen zielen eher auf eine fachliche Karriere ab. Wenn sie in einer Matrixorganisation arbeiten, teilt sie die Softwaregestaltenden flexibel zu IT-Projekten zu, um sie stetig auszulasten.

In den Fallstudien gibt es für Anwendende im Vergleich zu den anderen Gruppen weniger Karrieremöglichkeiten. Vertiefte Softwarekenntnisse zu erwerben, kann für sie hilfreich sein, um in der Hierarchie aufzusteigen. So bedeutet der Aufstieg als

18 Sieht man die sozialen Netzwerke von Xing und Linkedin als Wege, sich selbst zu vermarkten, unterstützt der Feldzugang (siehe Kapitel zu Forschungsdesign und Methode, siehe 3.2.1) die These, dass Softwaregestaltende und Programmierende sich stärker am externen Arbeitsmarkt orientieren: Auf den Webseiten von Xing und Linkedin sind vor allem Programmierende, IT-Beratende, ERP-Fachleute, Product Owner:innen, Projektmanagende, Anforderungsmanagende oder Scrum Master:innen zu finden.

Key User:in einen Aufstieg über Softwaregestaltung. Einige der befragten Team- oder Gruppenleitenden waren vorher Sachbearbeitende und damit reine Anwendende.

Kontrolle – eigenständiges Arbeiten in softwarezentrierten Prozessen

Beim Vergleich der Fallstudien überrascht, dass Führungskräfte in der Softwaregestaltung eine ähnliche Rolle spielen wie in der Anwendung und Programmierung. Unabhängig vom Arbeitsprozess halten sich die Führungskräfte aus der operativen Arbeit weitestgehend heraus. Bei allen ist der Kern der Kontrolle ein softwareintegrierter Arbeitsprozess, kombiniert mit Führungskräften, die nur in Ausnahmesituationen intervenieren und, wenn überhaupt, die Ergebnisse kontrollieren. Man könnte auch sagen: Die Beschäftigten arbeiten weitestgehend selbstständig an und mit der Software, die sie in einen digitalen Prozess integriert. Wobei die Grundkoordination (der soziotechnischen Konstellation) zwischen Anwendung und Programmierung Folgen für die Kontrolle hat.

Kontrolleigenschaften unabhängig von Gruppenzugehörigkeit: selbstständiges Arbeiten, softwarezentrierter Arbeitsprozess, Auswirkungen der Grundkoordination

Zuerst zu jenen Dingen, welche die Kontrolle der Arbeit unabhängig davon prägen, ob sie Teil von Softwareanwendung, -gestaltung oder -programmierung sind: Erstens arbeiten die Befragten eigenständig. Zweitens arbeiten alle softwarezentriert. Drittens prägt bei allen die soziotechnische Konstellation die Arbeit.

Erstens geben in allen Fällen und für alle Gruppen die Befragten an, dass sie eigenständig arbeiten und die Führungskräfte nur in Ausnahmefällen eingreifen und die Arbeitsergebnisse kontrollieren. Sie sind disziplinarische Führungskräfte, geben aber nur selten oder gar keine Arbeitsanweisungen. Je nach Fall legen sie das Budget fest, stellen Mitarbeitende ein, initiieren Projekte oder klären Konflikte innerhalb (z.B. mit anderen Führungskräften oder Teams) oder zu anderen Organisationen (Softwarefirma, IT-DL, EVU). In manchen Fällen wirken sie bei der Softwaregestaltung mit (EVU2 von KOOP2, einige EVU bei PAKET). Einige Zitate sollen das nochmal verdeutlichen:

Anwendende:

> »Wenn wir uns eine Woche lang nicht mehr zurückgemeldet haben, dann kommen schon Fragen: Warst du überhaupt beim Arbeiten oder hast du überhaupt die Arbeit gemacht? Wir arbeiten viel bei Kunden und wenn die etwas von uns wollen und da ist der Monteur nicht gekommen, der meldet sich natürlich auch dann, wie es denn aussieht, wenn er seinen Baustrom kriegt oder so.« (INTERN1 Monteur)
>
> »Ich war sehr selbstorganisiert, tatsächlich auch schon in der Abrechnung. Und meine Chefin hat mir da eigentlich auch immer den Weg auch so gelassen, weil ich hatte ja meine Fälle zugewiesen. Um die habe ich mich gekümmert. […] Natürlich hat jede Führungskraft drauf geschaut, wenn irgendwas länger liegt, wie es aussieht, warum es länger liegt. […] Ich habe meine Arbeit immer schon, ich sag es einfach mal, gut gemacht, so dass ich nicht kontrolliert werden musste.« (KOOP2 EVU2 Teamleiterin Mako)
>
> »Ja, ich bin recht frei in meiner Arbeit, tatsächlich. Ich darf sehr viel, Gott sei Dank. Ich werde nicht so viel kontrolliert.« (KOOP2 EVU2 Sachbearbeiterin)

Programmierende:

> »Und war haben Reviews, Code Reviews: Da kontrollieren wir uns gegenseitig auf Augenhöhe. Und wir haben verschiedene Analyse-Tools, die uns selber helfen, uns selbst zu kontrollieren. Also, im Grunde ist eigentlich die Eigenmotivation sehr, sehr hoch und für einen selbst, aber auch aus dem Entwicklerteam heraus. Und da gibt es eine gewisse Kontrolle. Wobei das nicht als Kontrolle empfunden wird, sondern eher als Unterstützung, würde ich sagen. Es gibt aber keinen, der auf meine Arbeit draufguckt. Könnte auch glaube ich keiner.« (INTERN1 Programmierer)

Gestaltende:

> »Es ist so bei uns, dass wir unsere Arbeitsbelastung gut selbst managen können.« (KOOP1 Applikationsbetreuer EVU2)
> »Aber zum großen Teil sind wir da relativ alleine unterwegs oder alleinständig[19] … eigenständig, so wollte ich sagen« (KOOP1 Anforderungsmanager IT-DL)

Zweitens sind die Befragten alle Teil eines softwarezentrierten Arbeitsprozesses mit unterschiedlichen Schwerpunkten, welche Software sie für ihre tägliche Arbeit hauptsächlich verwenden (ERP-System, Ticketsystem, E-Mails, MS Excel etc.). Die Anwendenden werden von der Software gesteuert (Monteur:innen) oder arbeiten ausschließlich mit ihr (Sachbearbeitende, Kund:innenservice). Genauso die Programmierenden, die ihre Programmierung über die Entwicklungsumgebung abarbeiten. Für die Softwaregestaltenden sind Ticketsysteme zentral. So spielt bei allen Software als ein Wissensobjekt (nach Rennstam 2012, ausführlich unter 6.4.2.1) und damit eine Form von Objekt-Kontrolle durch Technik eine Rolle. In allen Fällen werden die Tätigkeiten digital geführt – ob durch Ticketsystem, Entwicklungsumgebung oder ERP-Software.

Drittens prägt die Grundkoordination die Kontrolle: Von ihr hängt der Schwerpunkt der Kontrolle durch die Software und durch eine Gruppe (Management, Kundschaft, Kollegenschaft) ab.

Bei KOOP3 und STARTUP ist die Software vor allem ein Bezugspunkt zur Koordination und die softwarebasierten Werkzeuge fungieren mehr als Koordinationsinfrastruktur denn zur Kontrolle. Die Softwaregestaltenden können sie von überall aus nutzen. In den genannten Fällen kontrollieren aufgrund der digitalen Transparenz durch die verwendeten Werkzeuge und Methoden die Kolleg:innen mit. Bei KOOP1, KOOP2 und PAKET erfüllen die Ticketsysteme eine Funktion für die Kundschaft. Sie machen für diese transparent, was die Softwarefirma bzw. das IT-DL macht, und sind die Basis für die Überprüfung von SLA (z.B. die vereinbarten Reaktionszeiten für Tickets zu kontrollieren). Bei den internen Fällen ist nicht ganz klar, inwiefern das Management das Ticketsystem nutzt. Keine befragte Person hat das erwähnt oder von individueller Verhaltenskontrolle gesprochen. Trotzdem besteht intern für alle anderen Beteiligten, die an den

19 Auch wenn die Interpretation einzelner Phrasen und Wörter nicht Teil der Forschungsmethode ist, so zeigt dieser Versprecher doch auf, wie der Arbeitsalltag für viele aussieht: Sie arbeiten viel allein und bekommen wenig bis gar keine Anweisungen durch die Führungskraft im Arbeitsalltag.

Tickets arbeiten, eine Transparenz über den Arbeitsfortschritt. Keine befragte Person berichtet davon, dass Arbeitsschritte und das Arbeitstempo durch die Software vorgegeben sind. Vielmehr liefert die Software für jeden (Management, Kundschaft, Kollegenschaft) eine Rückmeldung, wie viele Fälle noch offen sind, welchen Bearbeitungsstand ein Ticket hat oder wo das nächste Strom- oder Gasnetz zu warten ist.

Je nach Grundkoordination ist der Schwerpunkt unterschiedlich, welche Gruppen kontrollieren: Die Fallstudien unterscheiden sich, ob mehr das Management (Hierarchie), die Kundschaft (Markt) und das Gegenüber (Netzwerk) kontrolliert. Bei den internen Softwaregestaltungen (INTERN1, INTERN2) entscheidet das Management über die Ressourcen (IT-Budget, Stellenvergabe, Entscheidung über Projekte). Zusätzlich zum Management gibt es bei den firmenübergreifenden Gestaltungsprozessen (KOOP1, KOOP2, PAKET, KOOP3) noch die Kundschaft bzw. die EVU, die in Form von SLA und Preisen kontrollierend wirken. Bei KOOP1, KOOP2 gibt es festgelegte Reaktionszeiten für bestimmte Tickets, welche die EVU den IT-DL stellen. Bei STARTUP und KOOP3 ist es schwer zu entscheiden, ob die Kolleg:innen oder die Kund:innen wichtiger sind. Denn sowohl in der Kollegenschaft besteht ein stetiger Austausch als auch mit der Kundschaft, die die App zur Anmeldung von E-Autos bzw. die IoT-Software nutzt. So oder so ist in beiden Fällen die hierarchische Kontrolle schwach. In keinem der Fälle gibt es eine direkte Kontrolle durch die Führungskraft in Form von konkreten Arbeitspaketen und wie diese abzuarbeiten sind.

In der Tabelle sind die Schwerpunkte der Kontrolle durch Gruppe und Ticketsystem je Fall aufgelistet, die mit der Grundkoordination zusammenhängen:

Tabelle 24: Vergleich Schwerpunkt der Kontrolle in Abhängigkeit zur primären Grundkoordination je Fall

Fall	Primäre Grundkoordination	Schwerpunkt Kontrolle durch Gruppe	Besondere Verwendung Ticketsystem
INTERN1	Hierarchie	Management (IT-Budget, Ziele)	Management-Kontrolle
INTERN2	Hierarchie	Management (IT-Budget, Ziele)	Management-Kontrolle
KOOP1	Markt	Verträge (SLA), Preise	Kontrolle durch Kundschaft bei IT-DL
KOOP2	Markt	Verträge (SLA), Preise	Kontrolle durch Kundschaft bei IT-DL
PAKET	Markt	Verträge (SLA), Preise	Kontrolle durch Kundschaft bei SF
KOOP3	Netzwerk	Ko-Produktion	Koordination
STARTUP	Netzwerk	Kollegenschaft/Peers und Kundschaft	Koordination

Kontrolleigenschaften je Gruppe: Unterschiede in der qualitativen und quantitativen Intensivierung

Im Vergleich zur Gruppe der Anwendenden und Programmierenden sind die **Gestaltenden** zum einen Teil eines anderen Arbeitsprozesses und allein deshalb schon anders kontrolliert. Wie bereits in den vorhergehenden Kapiteln ausgeführt, spielt für sie in ih-

rer Arbeit Kommunikation und Kooperation eine größere Rolle. Ihre Arbeit ist weniger durch die Software vorgegeben (z.B. bearbeiten sie nicht nur Restfälle). Für sie ist Software mehr ein Hilfsmittel. Sie sind interdisziplinär zwischen Anwendung und Entwicklung tätig. Mehrere Befragte geben explizit an, dass sie ihre Arbeit gern machen. Ihre Arbeit ist unterschiedlich stark formalisiert (z.B. wie genau der Ablauf des Anforderungsmanagements vorgegeben ist). Sie sind einer *qualitativen Intensivierung* ausgesetzt. Erstens sind sie das, weil sie in vielen Fällen IT-, energiewirtschaftliches und Methodenwissen kombinieren (Key User:innen, IT-Beratende, Anwendungsbetreuende) oder zumindest mit den jeweiligen Expert:innen kommunizieren können müssen (Anforderungsmanagende, IT-Projektmanagende). Zweitens arbeiten sie öfters gleichzeitig in mehreren Projekten und müssen mehrere Prozessabschnitte der Datenverarbeitung überblicken und die gestaltete Software ist nicht ihr alleiniger Arbeitsgegenstand. Die *quantitative Intensivierung* ist bei den Gestaltenden schwer zu beurteilen. Mehrere sprechen davon, dass es phasenweise mal mehr und mal weniger zu tun gibt und sie selbst Grenzen setzen müssen (durch Priorisierung oder »Nein« sagen können). Die meisten sprechen von einer geregelten Arbeitszeit. Zudem sind die Kontexte sehr unterschiedlich: ob sie für mehrere Kund:innen und in mehreren Projekten tätig sind oder nicht.

Die Arbeit der **Programmierenden** wird durch die Tests anderer und/oder durch Code-Reviews von Kolleg:innen geprüft. Es ist durch die Ticketsysteme transparent, was sie abgearbeitet haben, und falls es regelmäßige Treffen wie Daily bei Scrum gibt, müssen sie den Teammitgliedern Rede und Antwort stehen. Ihre Spezialisierung und die meist schwierige Abschätzung des Arbeitsaufwandes geht mit einer fehlenden detaillierten Kontrolle von außen einher. Eine *qualitative Intensivierung* existiert bei den befragten Programmierenden, weil sie fachliches mit IT-Wissen kombinieren und in verschiedenen Projekten, Rollen oder für mehrere Kund:innen tätig sind. Sie spitzt sich bei Programmierenden wie im Fall eines Befragten von KOOP2 zu, der für mehrere, unterschiedliche energiewirtschaftliche Fachbereiche entwickeln muss und dadurch entsprechend mehr Wissen braucht. Zudem müssen aufgrund individueller Anpassungen für einzelne EVU bei KOOP1 und KOOP2 die Programmierenden den Überblick über diese individuellen Umsetzungen behalten. Einige Programmierende beschreiben es als positiv, wenn sie sich auch mal in neue Technologien einlernen müssen. Wie bei den Gestaltenden ist es bei den Programmierenden schwierig, pauschal von einer *quantitativen Intensivierung* zu sprechen. Sie arbeiten in den untersuchten Fällen meist Programmieraufgaben ab, für die Prioritäten vorgegeben sind. Das lässt Spielraum, um eigene Grenzen zu setzen:

> »Ich könnte mich gerne überlasten, aber ich tu's nicht.« (Programmierer KOOP1)

Eine Intensivierung ist aber nicht ausgeschlossen, wenn z.B. eine fixe Deadline existiert: wenn zu einem Stichtag mehrere umfangreiche Umsetzungen, z.B. aufgrund einer neuen Regulierung, abgeschlossen werden müssen. Zwar arbeiten die Programmierenden in einigen Fällen für mehrere EVU gleichzeitig (bei KOOP1, KOOP2, KOOP3 und PAKET). Doch hat dies nicht zwangsweise eine quantitative Intensivierung zur Folge. Eine Programmierung kann für viele EVU gleichzeitig gelten (bspw. bei einer Standardsoftware), d.h., eine Fehlerkorrektur kann gleich mehrere Anfragen von EVU zufriedenstellen.

Die **Anwendenden** sind hierarchisch und von der Bezahlung Programmierenden und Gestaltenden untergeordnet. Die Anwendenden sind meist passiv und reagierend: Sie müssen sich auf Änderungen einstellen oder erfahren erst nach Fertigstellung, was sich geändert hat. Ihre Arbeit ist das Objekt der Gestaltung und nur wenige gestalten mit. Vor allem wenn die Softwaregestaltung intern stattfindet, können Anwendende eigene Wünsche äußern und der Betriebsrat die Mitgestaltung einfordern. Ihre Arbeit ist in den meisten Fällen das, was die Software nicht erledigen kann bzw. noch nicht in dieser abgebildet ist. Das bedeutet auch, dass sie fehlerhafte oder nicht ausgereifte Software kompensieren müssen. Sie fungieren als Puffer für mangelhafte Softwareentwicklung. In einem Fall ist die Cloud langsam (so ein Befragter von KOOP2), im anderen sind in den Softwareupdates Fehler, welche die Arbeit der Anwendenden belasten (PAKET), oder die Prozessintegration leistet der Anwendende manuell (INTERN2). Sie müssen mit den Reaktionszeiten leben, die sich durch die Zusammenarbeit mit einem IT-DL oder Softwareunternehmen ergeben. Diese Puffer-Position kann über Fragen der Softwaregestaltung hinausgehen und die Anwendenden können zu allgemeinen, operativen Puffern werden. Ein befragter Gruppenleiter, der auch operative Software anwendet, hatte Burnout. Diesen hatte er vor allem, weil zu wenig Personal vorhanden war, und wegen der ständigen regulatorischen Änderungen und des Termindrucks im Bereich der Energiemengenbilanzierung, in der er tätig ist. Bei der Energiemengenbilanzierung müssen zu einem Stichtag nicht nur Daten, sondern vor allem Geld fließen. Anderen Anwendenden wie der befragten Sachbearbeiterin war die Restfallbearbeitung zu viel Routinearbeit, weswegen sie die Stelle gewechselt hat. Zudem sind die Anwendenden direkt im Kontakt mit der Kundschaft. Der befragte Monteur meint, er bekommt die Aggressionen der Kundschaft ab, wenn er bspw. wegen einer Baustelle den Strom bei jemandem abstellen muss. Für einen Teil der Anwendenden findet eine *qualitative Intensivierung* statt, weil sie vermehrt in Prozesszusammenhängen denken müssen (Sachbearbeitende), mehrere Fachgebiete (Key User:innen) oder mehrere Netzgebiete und Sparten kennen müssen (Monteur:innen). Arbeiten Anwendende im BPO-Bereich, arbeiten sie für eine größere Kundschaft und es ist mit einer *quantitativen Intensivierung* zu rechnen. Wobei in einigen EVU diese auch unabhängig davon aufgrund von Personalmangel oder phasenweise durch IT-Projekte entsteht.

Wissensverteilung – Praxisgemeinschaft und in Software materialisiertes Wissen

Es lassen sich grundsätzlich drei Wissensgruppen unterscheiden: Programmierung, Anwendung und Gestaltung. Dabei fällt in den Fallstudien auf, dass es für die Softwaregestaltenden weniger darum geht, viel Wissen anzusammeln. Sie sind vielmehr dazu da, um Wissen in Software(quellcode) zu überführen. Primär geht es darum, an das notwendige interdisziplinäre Wissen heranzukommen und lernbereit zu sein. Wissen selbst zu haben ist zweitrangig, wenngleich nützlich und in den meisten Fällen vorhanden. Das Wissen von Softwaregestaltenden kann umfassend sein: Wissen über die Möglichkeiten der Softwaregestaltung und individuelle Bedarfe der EVU oder allgemeine der Branche zu haben. Methoden wie Scrum oder IT-Projektmanagement zu beherrschen. Manche setzen Softwaregestaltung selbst um, wenn sie programmieren oder Einstellungen an einer Standardsoftware vornehmen. Der Extremfall sind IT-Beratende, die sich energie-

wirtschaftlich auskennen, Anforderungen schreiben, selbst programmieren und Einstellungen an einer Standardsoftware vornehmen. Doch auch sie müssen sich mit Anwendenden oder anderen Beteiligten wie IT-DL oder Softwarefirma absprechen. Die Praxis ist deshalb so entscheidend, weil die Software sich stetig verändert, ebenso wie die Organisationen und die Branche. Es muss immer wieder neu verstanden werden, was energiewirtschaftlich notwendig und was softwaretechnisch möglich ist. Drei Thesen stehen im Mittelpunkt der Zusammenfassung zur Kategorie Wissensverteilung:

1. Die Softwaregestaltung ist eine Praxisgemeinschaft, in der die Softwaregestaltenden arbeiten.
2. Für das Verständnis der Wissensverteilung ist das in Software materialisierte Wissen entscheidend.
3. Das Management verfügt nur selten über Wissen zur Softwaregestaltung.

Erstens stellt sich der Arbeitsprozess der Softwaregestaltung in den Fallstudien als eine Praxisgemeinschaft dar. Sie verfügt über das Wissen zu Methoden der Softwaregestaltung, und wenn sie schon selbst kein tiefergehendes Wissen über energiewirtschaftliche Regulierung, Standardsoftwarelösungen oder Programmierung hat, schafft sie zumindest die Möglichkeit, sich darüber auszutauschen – ob längerfristig oder temporär in Projekten. Dass die Praxis wichtiger ist, als Wissen zu besitzen, zeigt sich an drei Aspekten: an a) dem kontinuierlichen Austausch zur Softwaregestaltung, der stattfindet und zu Anforderungen führt (von dem bestimmte Gruppen ausgeschlossen sind), dass b) unterschiedliche Praxis- und damit Lernbiografien entstehen und c) EVU feststellen, dass sie nur dann Software gestalten können, wenn sie Teil dieser Praxis sind.

Zu a) In der Softwaregestaltung arbeiten Beschäftigte zwischen Anwendung und Programmierung zusammen. Auch einzelne Anwendende wie Key User:innen sind Teil davon. Die Gestaltenden haben Kontakte, kennen die Organisation und einzelne Umsetzungen und Methoden wie Scrum. Es besteht ein kontinuierlicher Austausch zwischen einem Kern an Mitarbeitenden in den Fällen INTERN1, INTERN2, KOOP3, STARTUP, genauso wie bei PAKET intern in der Softwarefirma. Bei KOOP1 und KOOP2 sind es längere Beziehungen zwischen den Firmen und viele Treffen, welche eine gemeinsame Praxis herstellen. Aber auch innerhalb des IT-DL und teilweise in den EVU arbeiten die verschiedenen Gruppen kontinuierlich zusammen. Bestimmte Berufsgruppen wie IT-Beratende erfahren einen kontinuierlichen interdisziplinären Wissenserwerb. Sie machen nicht nur in verschiedenen Projekten mit, sondern sind sowohl fachlich nah an den Anwendungsbereichen als auch softwaretechnisch nah an der Software dran. Die Programmierenden verfügen in allen Fällen über energiewirtschaftliches Wissen, um die industriespezifischen Anforderungen umzusetzen.

Zu b) Dabei bestimmt die Praxis, was die oder der Einzelne weiß. Es entstehen durch individuelle, interdisziplinäre Praxisbiografien individuelle Lernbiografien, weil je nach Softwaregestaltung die Softwaregestaltenden an unterschiedlichen Softwarelösungen, an unterschiedlichen Anforderungen, Projekten, energiewirtschaftlichen Themengebieten und mit unterschiedlichen EVU zusammenarbeiten. Wer mitgemacht hat bei der Gestaltung, weiß mehr darüber. Die klassische Lernbiografie von Ausbildung, dann Sachbearbeitertätigkeit, Wechsel in ein anderes Team oder Aufstieg zur Teamleitung gibt es

immer noch. Diese wird jedoch nun durch neue Möglichkeiten ergänzt. Vor allem die befragten Gestaltenden und Programmierenden haben mehr interdisziplinäres Wissen und Einkommen im Laufe der Zeit gewonnen, ohne dadurch hierarchisch aufgestiegen zu sein.

Zu c) Dass die Praxisgemeinschaft den Zugang zu Wissen sichert, zeigt sich daran, dass das Wissen, das über Anforderungen in der Software eingeflossen ist oder dokumentiert wurde, allein nicht ausreicht. Es macht einen Unterschied, ob man Teil der Praxis war bzw. ist oder nicht. Das zeigen vor allem Marktbeziehungen, die zu einer Abhängigkeit führen und Hürden für eine gemeinsame Praxis darstellen. Bei KOOP2 verlagert ein EVU die Anwendung der Software zurück in das Unternehmen, um intern Wissen zu haben und die Softwaregestaltung selbst machen zu können. Zudem bauen einige EVU intern IT-Projektmanagement-Kompetenz auf. Innerhalb des IT-DL von KOOP1 arbeiten die Programmierenden eng mit energiewirtschaftlichen Fachleuten zusammen, wodurch ein stetiger Wissensaustausch besteht. Ein Klient des IT-DL von KOOP1 hat das Wissen, was das IT-DL über die Jahre über das EVU gesammelt hat, unterschätzt. Er hat den Wechsel zu einem anderen IT-DL bereut. Bei PAKET sind kleinere EVU abhängig von externem Support der Softwarefirma (z.B. durch IT-Beratende), weil sie intern keine ERP-Fachleute haben. Bei KOOP3 schickt das IT-DL Newsletter über Softwareupdates, die sich die EVU durchlesen können. Die reine Anwendung, die nicht an der Softwaregestaltung teilnimmt, ist aus der Wissensgemeinschaft Softwaregestaltung ausgeschlossen. Das ist in allen Fallstudien so.

Zweitens prägt die Wissensverteilung die Arbeit der Softwaregestaltenden dahingehend, dass der zentrale Ort des Wissens die Software selbst ist: ob Quellcode der gestalteten Software oder die verwendeten Softwarewerkzeuge wie Ticketsysteme. Immer mehr Wissen, ob über energiewirtschaftliche Regulierung, Geschäftsprozesse, Arbeitsabläufe, Softwareänderung etc., wandert in Software. Dabei müssen die Softwaregestaltenden alles, was sie gestaltet haben, nicht mehr aktiv wissen – auch nicht, wie es die Programmierenden umgesetzt haben. Das zeigen alle Fallstudien: Peu à peu wächst durch Anforderungen die Software. In ERP-Systemen wie jenem aus der Fallstudie PAKET steckt alles, was zum operativen Betrieb der energiewirtschaftlichen Datenverarbeitung notwendig ist. Dabei findet die Softwaregestaltung in einem technischen Umfeld aus lesbaren Objekten statt. Softwaregestaltende dokumentieren in den softwarebasierten Werkzeugen wie Ticketsystem, Dokumentationssoftware[20], MS Sharepoint oder E-Mails (inkl. Newsletter über Updates), Programmierende im Quelltext der Software selbst.

Was alle Befragten egal welcher Gruppe sagen, ist, dass Learning by Doing für sie wichtig ist – ob aus Dokumenten oder mit der Software selbst. Auch wenn viel Wissen lesbar oder durch Ausprobieren und Mitmachen erlernbar ist, sieht ein Befragter Software als primäres Wissensquelle und -lager kritisch: Ein Betriebsrat findet, dass sein EVU zu sehr auf Learning by Doing im Softwaregebrauch setzt. Es sollten wieder mehr Schulungen stattfinden. Ein befragter Programmierer hält eine Art Führerschein für Anwendende für hilfreich, weil sie in der alltäglichen Anwendung nicht alles lernen (bspw. die Zusammenhänge und Einstellungsmöglichkeiten in der Software). Bei PAKET sagen

20 Eine weit verbreitete Softwarelösung dafür, die auch STARTUP einsetzt, ist Confluence der Firma Atlassian.

mehrere befragte Personen, dass die Anwendenden eine funktionierende Software wollen, sich nicht eigenständig mit den möglichen Einstellungen befassen möchten und in ihrem Arbeitsalltag nicht genug dazu lernen.

Drittens verfügt das Management wie Team-, Abteilungs-, Bereichs- oder IT-Leitung nur in wenigen Fällen über Softwaregestaltungswissen. Größtenteils nimmt es nicht direkt an der Softwaregestaltung teil und interveniert teilweise nur bei der Auswahl der Anforderungen. Drei befragte Team- bzw. Gruppenleitende wirken bei der Softwaregestaltung mit. Wobei es dabei immer um eine Standardsoftware geht, die nur durch Einstellungen von den EVU verändert werden kann. Softwaregestaltungskompetenzen haben nur wenige FK (wie z.B. Schulungen in Scrum oder IT-Projektmanagement). Das Management setzt den Rahmen z.B. durch das IT-Budget oder kann in Matrixorganisationen ein wichtiger Sponsor für Softwaregestaltungsprojekte sein, indem es z.B. die Zusammenarbeit mit anderen Abteilungen durch die Führungskräfte dort vorantreibt. KOOP1, KOOP2 und PAKET nutzen Marktmechanismen und kaufen die Software(gestaltung) ein. Der Nachteil ist, dass das Wissen dann nicht direkt im Zugriff des EVU-Managements ist. Wie auch allgemein für Anwendung und Entwicklung hat das alles zur Folge, dass das Management bzw. die Führungskräfte keine direkten, konkreten Anweisungen mehr geben können, weil ihnen das Wissen dazu fehlt. In dem Fall, in dem Arbeitsanweisungen gemacht werden, erstellen dies die Key User:innen (PAKET, EVU5). In anderen Anwendungsbereichen werden keine verwendet (Sachbearbeitende KOOP2 EVU1, Teamleitung EVU2 KOOP2, Anwendende KOOP1 EVU3, Anwendende EVU1 KOOP1). Die Sachbearbeitenden von EVU2, KOOP2 haben nur ein Schema, wie sie vorgehen sollen, weil die Fälle zu individuell sind.

Allgemein wird das Wissen immer umfangreicher und eine Organisation allein kann nicht mehr darüber verfügen. Zum Beispiel sind bei INTERN1 und INTERN2 einige programmierende Externe. Bei KOOP1, KOOP2, KOOP3 verteilt sich das Wissen auf mehrere Organisationen. Die fachliche Spezialisierung der Mitarbeitenden nimmt zu und damit das Wissen, was ihre Führungskräfte nicht haben: u.a. durch Automatisierung, komplexere Regulierung, gestiegene Bedeutung von IT, höhere Ansprüche der Kundschaft, Produktvielfalt.

8.5. Folgen für die soziotechnische Arbeitsgestaltung der Softwareanwendung in den EVU

Ein Teil der Arbeitsgestaltung der Softwareanwendung in den EVU hängt an der Softwaregestaltung. Der Abschnitt nimmt diesen Teil in den Blick, der sich aus dem Verhältnis der Arbeitsprozesse von Softwaregestaltung und Softwareanwendung ergibt. Er lässt sich mithilfe der Kategorien Einfluss und Konflikt analysieren. Anders als im letzten Punkt 8.4. geht es dabei nicht um einzelne Gruppen von Beschäftigten. Es geht um das Verhältnis zweier Arbeitsprozesse zueinander, über das nicht nur die operativ Softwaregestaltenden oder -anwendenden entscheiden, sondern in den Fällen, wo Hierarchien vorliegen, das Management strategische Entscheidungen fällt, Ziele vorgibt, Ressourcen zur Verfügung stellt und eigene Anforderungen einbringt.

Der **Einfluss** beider Arbeitsprozesse aufeinander betrifft vier Aspekte:

- Kontrollverhältnis: Wird der Arbeitsprozess der Softwareanwendung jenem der Softwaregestaltung untergeordnet? Werden Softwareanwendung und -gestaltung z.B. getrennt in zwei verschiedenen Organisationen kontrolliert?
- Reorganisation: Wird der Arbeitsprozess der Softwareanwendung zum Zwecke der Softwaregestaltung reorganisiert?
- Ziele: Wie verhalten sich die jeweiligen Ziele von Softwareanwendung und -gestaltung zueinander und wer kann sie festlegen (z.B. individuell etwas für einen Anwendungsbereich zu entwickeln oder eine skalierbare Standardsoftware; die Anwendungsorganisation zu reorganisieren oder nicht)?
- Partizipation: Wer gestaltet wie mit? Wer aus der Softwareanwendung hat Zugang zur Softwaregestaltung? Welche Rollen nehmen Expert:innen ein und wie verhalten sich Anwendenden- und Unternehmens- bzw. Managementanforderungen zueinander?

Für zwei der genannten Aspekte des Einflusses der Softwaregestaltung auf die Softwareanwendung diskutiert der Abschnitt folgende Thesen:

1. Für die Reorganisation deutet sich in einigen EVU an, dass sie sich auf eine bestimmte Form der Organisation zubewegen: eine auf die Softwaregestaltung ausgerichtete softwaretechnische Prozessorganisation. Eine solche Organisation bedeutet nichts anderes, als dass die Softwaregestaltung die Software und die Organisationen eines End-to-end-Prozesses übergreifend und ohne die Behinderung durch Fachbereichsgrenzen ändern kann.
2. Hinsichtlich der Partizipation stellt sich die Frage, inwieweit die Anwendenden an der Gestaltung der Software beteiligt sind und ob sich die Fälle Partizipationstypen zuordnen lassen.

Die **Konflikte** zwischen Gestaltung und Anwendung sind entweder innerhalb (wie jene zwischen Management- und Anwendendenanforderungen) oder außerhalb der EVU (wie z.B. zwischen individuellen Alleingängen einzelner EVU und kooperativem Standard mehrerer EVU). Dabei zeigen die Fallstudien, dass Betriebsräte innerhalb der EVU bei der soziotechnischen Arbeitsgestaltung insofern mitbestimmen, indem sie intervenieren, aber nicht aktiv gestaltend sind. Sie spielen bei Fragen der Partizipation insofern eine Rolle, als sie bei Rahmenbedingen mitentscheiden und weniger bei der inhaltlichen Gestaltung der Software (z.B. wenn ein Betriebsrat einfordert, dass die Softwaregestaltung Anforderungen der Anwendenden berücksichtigt).

Dieser Abschnitt geht in den folgenden Punkten auf die Unterschiede zwischen den Fallstudien ein, stellt die Fälle mithilfe der Kategorien Einfluss und Konflikt einzeln dar und fasst die Ergebnisse zusammen.

8.5.1. Soziotechnische Arbeitsgestaltung: zwischen Abhängigkeit und Unabhängigkeit

Die Fälle lassen sich dahingehend unterscheiden, ob die EVU unabhängig oder abhängig von IT-DL oder Softwarefirmen die Arbeit der Softwareanwendung gestalten können.

Welcher Typ in einem Fall vorliegt, ergibt sich aus der soziotechnischen Konstellation und dem Arbeitsprozess der Softwaregestaltung.

Bei der **unabhängigen** soziotechnischen Arbeitsgestaltung ist aufgrund der Arbeitsteilung die Softwaregestaltung Teil der anwendenden Organisation (EVU) und der **Einfluss** auf die soziotechnische Arbeitsgestaltung durch die Softwaregestaltung umfasst nicht nur die Software, sondern auch den anzuwendenden Arbeitsprozess. Die **Konflikte** bei der Arbeitsgestaltung sind rein intern. Je nach Grundkoordination bestehen dann diese internen Konflikte in Hierarchien, Märkten oder Netzwerken.

Bei einer soziotechnischen Arbeitsgestaltung vom Idealtyp **abhängig** reduziert sich der **Einfluss** der Softwaregestaltung auf die angewendete Software und ist nur über Externe möglich (IT-DL, Softwarefirmen). Das liegt an der Arbeitsteilung, bei der sich die EVU ausschließlich auf die Softwareanwendung konzentrieren. Die **Konflikte** in der Arbeitsgestaltung durch Softwaregestaltung existieren entsprechend vor allem mit den externen Softwaregestaltenden. Je nach Grundkoordination bestehen dann diese Konflikte in Hierarchien, Märkten oder in Netzwerken.

Tabelle 25: Idealtypen unabhängige und abhängige soziotechnische Arbeitsgestaltung

Typ	Einfluss der Softwaregestaltung	Konflikte
unabhängig	intern: Software und Organisation	intern
abhängig	extern: nur auf Software	extern

Anhand von vier Unterkategorien lässt sich der Einfluss zwischen Softwareanwendung und -gestaltung näher beschreiben: Kontrollverhältnis, Reorganisation, Ziele und Partizipation.

Was das Verhältnis der Arbeitsprozesse der Softwareanwendung und Softwaregestaltung zueinander betrifft, ist bei einer **unabhängigen** Arbeitsgestaltung das **Kontrollverhältnis** beider so, dass die Softwaregestaltung die Softwareanwendung kontrollieren kann, was sich z.B. darin niederschlägt, dass Softwareanwendende oder deren Führungskräfte Veränderungen aufgrund der Softwaregestaltung nicht verhindern können. Zudem ist eine wechselseitige **Reorganisation** von Softwaregestaltung, Software und Softwareanwendung möglich. Bei den **Zielen** kann das EVU sowohl über jene der Softwareanwendung als auch der -gestaltung entscheiden. Ebenso kann bei der **Partizipation** das EVU entscheiden, wer an der Softwaregestaltung teilnimmt, und es ist ein direkter Einbezug der Anwendenden möglich.

Was das Verhältnis der Arbeitsprozesse der Softwareanwendung und Softwaregestaltung zueinander betrifft, ist bei einer **abhängigen** Arbeitsgestaltung das **Kontrollverhältnis** beider so, dass sie sich nicht gegenseitig kontrollieren. Die Softwaregestaltung kann bei der -anwendung keine Veränderungen durchsetzen. Auch eine **Reorganisation** von Softwareanwendung und -gestaltung ist nur getrennt möglich. Die **Ziele** beider Arbeitsprozesse sind getrennt: Jene des Arbeitsprozesses Softwaregestaltung sind auf den Softwareproduzenten (IT-DL oder Softwarefirma) ausgerichtet und die vom Arbeitspro-

zess der Softwareanwendung auf das EVU. Die **Partizipation** konzentriert sich auf Fachexpert:innen und nicht auf Anwendende bzw. entscheidet nicht das EVU darüber, wer mitgestaltet.

Tabelle 26: Idealtypen unabhängige und abhängige soziotechnische Arbeitsgestaltung – Unterkategorien

Typ	Kontrollverhältnis Softwareanwendung (SA) – Softwaregestaltung (SG)	Reorganisation SA – SG	Ziele SA – SG	Partizipation SA an SG
unabhängig	Kontrolle SA durch SG möglich	Software, SG und SA im Wechselspiel möglich	beides auf EVU-Nutzen ausgerichtet	direkter Einbezug Anwendende möglich
abhängig	Kontrolle SA – SG: nur getrennt möglich	nur getrennt möglich	Ziel SG entscheidet Softwarefirma oder IT-DL	(Branchen-)Fachleute, anwendende Organisation entscheidet nicht

Sind nun die einzelnen Fälle Beispiele für eine unabhängige oder abhängige soziotechnische Arbeitsgestaltung der Softwareanwendung? Antwort darauf geben die Falldarstellungen und die Zusammenfassung am Schluss dieses Abschnitts.

8.5.2. Darstellung der Fallstudien

Das Verhältnis der Arbeitsprozesse von Anwendung und Gestaltung stellt dieser Abschnitt für die sieben Fallstudien jeweils anhand der Kategorien Einfluss (inkl. ihrer vier Unterkategorien) und Konflikt dar.

8.5.2.1. INTERN1: iterativer soziotechnischer Wandel eines Fachbereichs, unabhängig

Es hat Folgen für die soziotechnische Arbeitsgestaltung des Anwendungsbereichs der Software, dass das EVU in der Fallstudie die Möglichkeiten der individuellen Softwaregestaltung dezentral für den Fachbereich Instandhaltung nutzt: Dadurch kann das EVU unabhängig Arbeit via Software gestalten, d.h. den Arbeitsprozess der Softwareanwendung verändern. Das zeigt sich erstens am Einfluss. Denn das EVU hat sowohl Einfluss auf die Gestaltung der Software als auch auf die anwendende Organisation. Zudem kann das EVU eigenständig über die Ziele sowohl von Softwaregestaltung als auch Softwareanwendung entscheiden. Der Einfluss erlaubt zuletzt eine direkte Partizipation der Anwendenden – wenn auch begrenzt. Zweitens verlaufen die Konflikte zwischen den beiden Arbeitsprozessen innerhalb des EVU (zwischen Fachbereich und IT-Abteilung), wodurch es diese eigenständig lösen kann. Allerdings muss sie diese Konflikte innerhalb bestehender Hierarchien lösen, in denen das Management das Sagen hat.

Die unabhängige Arbeitsgestaltung zeigt sich beim Einfluss auf vierfache Weise. Erstens zeigt sie sich am **Kontrollverhältnis**: Die Softwaregestaltung hat die Kontrolle über die energiewirtschaftlichen Prozesse, weil die Anwendenden zwar die Möglichkeit haben, an der Gestaltung mitzuwirken, aber nicht, diese aufzuhalten, und weil sie grundlegende Vorgaben (wie die Arbeitssteuerung über mobile Endgeräte) nicht ändern können. Das zeigt sich auch daran, dass die vier Product Owner:innen, die jeweils u.a. für unterschiedliche Funktionalitäten der Instandhaltungssoftware zuständig sind, die bestehenden Hierarchien aus Fachbereichen und IT nicht berücksichtigen müssen, was zudem den Entscheidungsaufwand verringert:

> »Unsere POs, die können Dinge entscheiden. Man muss nicht wegen jedem blöden Knopf zu drei Chefs rennen und sagen: Ist der jetzt grün oder blau? Relativ umfangreiche Dinge entscheiden wir einfach selbst und machen das und das geht auch ganz gut.« (Programmierer)

Zweitens gestaltet das EVU nicht nur eine individuelle Software, sondern **reorganisiert** die Arbeit in der Softwareanwendung. Zum einen tut es das, weil das Gestaltungsnetzwerk aus Anforderungsmanagenden und Key User:innen Teil der Fachbereiche geworden ist. Zum anderen ändert der Arbeitsprozess der Softwaregestaltung iterativ den Arbeitsprozess der anwendenden Instandhalter:innen und Dispatcher:innen, indem er diesen nicht nur abbildet, sondern in Frage stellt und ändert. So gibt es z.B. die Position des Meisters nicht mehr. Wechselseitig entstehen eine individuelle Software und eine veränderte anwendende Organisation. Mittlerweile hat sich durch das iterative Vorgehen eine Sättigung an möglichen Anforderungen vor allem bei der zentralen App für die Arbeitssteuerung der Monteur:innen eingestellt, weswegen neue Bereiche wie die Arbeit der Dispatcher:innen in den Blick genommen werden.

Drittens hat sich das EVU zum Ziel gesetzt, die Software ausgehend vom eigenen, althergebrachten Status quo der Softwareanwendung zu gestalten. Das heißt, das EVU hat als Ziel, die Anwendung inkrementell zu verändern. Es dockt dabei zwar an das Standardpaket von SAP an, kann dabei aber seine eigenen Ziele verfolgen, weil es die individuelle Erweiterung unabhängig von SAP gestaltet.

Zuletzt zeigt sich die unabhängige soziotechnische Arbeitsgestaltung durch die Softwaregestaltung an der **Partizipation**. Denn das EVU kann selbst entscheiden, wer mitgestaltet. Ausschließlich eigene Mitarbeitende gestalten mit. Anwendende können moderiert über Anforderungsmanagende oder Product Owner:innen Anforderungen aufgeben. Das heißt nicht, dass die internen Hierarchien nicht mehr gelten. Die Managementziele bleiben unangetastet. Die Beteiligten weichen nicht von Zielen wie der Kostenoptimierung ab. Die Partizipation durch Anwendende und deren Anforderung sind vor allem nützlich und steigern die Akzeptanz, ohne die Ziele des Managements in Frage zu stellen. Die Partizipation war vom Betriebsrat gewünscht, aber auch Teil eines bewussten Change Managements, wodurch das Management Konflikte mit Beschäftigten vermeiden wollte und auch vermieden hat.

Der kontinuierliche Fortschritt bei der Entwicklung hängt davon ab, dass die Softwareanwendung und -gestaltung durch die bestehende Konstellation entstehende **Konflikte** löst. Diese sind in dem Fall rein intern und so kann das EVU sie eigenständig lösen.

Sie zeigen sich zum einen an der zwiespältigen Lage von Product Owner:innen und Anforderungsmanagenden, weil Erstere in der IT-Abteilung, Letztere im Fachbereich angesiedelt sind, Erstere Anforderungen des Managements aufnehmen und Letztere jene aus dem Fachbereich. Zum anderen zeigen sich die Konflikte in der Zusammenarbeit mit dem Betriebsrat.

Die Fachabteilung Instandhaltung ist nur Auftraggeber und nicht steuernd. Der Haupt-Product-Owner aus der IT-Abteilung entscheidet über Ressourcen wie Budget und Priorisierung. Weil nun das obere Management über den Haupt-Product-Owner Anforderungen einbringt und unterschiedliche Vorstellungen über die Priorisierung größerer Themen bestehen, kommt es zu Konflikten. Die Anforderungsmanagerin aus dem Fachbereich hat es schwer:

> »Also, so wie wir heute die Hierarchien haben und leben, könnte ich das nie und nimmer durchsetzen, wenn ich jetzt da eine andere Vorstellung hätte.« (Anforderungsmanagerin)

Bei diesem Konflikt zwischen Unternehmens- und Anwendendensicht hat das letzte Wort der Haupt-Product-Owner. Nur aufgrund guter Kontakte kann der Fachbereich seine Vorstellungen einbringen, trotz bestehender Hierarchien zwischen den Product Owner:innen aus der IT-Abteilung und dem Fachbereich. Letztendlich hätte der Fachbereich gern die allgemeine Entscheidungshoheit (über Priorisierung, Budget, Aufgabenverteilung). Es ist aber noch offen, ob er diese in Zukunft bekommen wird.

Für den Betriebsrat bietet die unabhängige Arbeitsgestaltung durch die interne Softwaregestaltung mehrere Vorteile. Seine Forderung nach stärkerer Partizipation der Monteur:innen wurde umgesetzt. Er kann bereits im Lauf der Entwicklung darauf achten, dass keine individuelle Leistungskontrolle möglich wird. Einige der Befragten (Programmierer, Architekt) berücksichtigen deswegen bereits bei der Konzeption, dass sie auf die Interessen des BR eingehen. Zudem ist er gut informiert, weil er in den entsprechenden Treffen wie andere Stakeholder der Softwareentwicklung dabei ist. Andererseits gesteht er ein, dass er auf bestimmte Entscheidungen, die dort getroffen werden, keinen Einfluss hat. Der Spielraum des BR, was die Gestaltung anbelangt, ist auch bei dieser internen Softwaregestaltung beschränkt.

8.5.2.2. INTERN2: fachbereichsübergreifende Softwaregestaltung, unabhängig

Wie bei INTERN1 kann auch in diesem Fall das EVU unabhängig die Software und damit den eigenen Arbeitsprozess der Softwareanwendung gestalten. Jedoch ist der Einfluss der Softwaregestaltung auf die Softwareanwendung geringer. Zum einen liegt das daran, weil sie nur eine der Softwarelösungen, welche die Anwendenden verwendet, gestaltet. Zum anderen ist allein die Software im Fokus und weniger die Anwendungsorganisation insgesamt. Das liegt vor allem daran, dass die Softwaregestaltung die Organisation mehrerer Fachbereiche verändern müsste – anders als bei INTERN2. Es gibt aber die Idee, die Team- und Abteilungssilos aufzulösen und ein integriertes Prozessteam für die Softwaregestaltung zu schaffen (siehe dazu die Diskussion zur softwaregestaltenden Prozessorganisation 8.5.3.4). In diesem Fall verlaufen die Konflikte vor allem zwischen

den Fachbereichen. Allerdings kann das EVU diese, wie auch jene mit dem Betriebsrat, eigenständig lösen, weil es interne Konflikte sind.

Auch wenn die Softwaregestaltung insofern Einfluss auf die Anwendung hat, als dass sie iterativ die Software ändert, gibt es Unterschiede zu INTERN1. Prinzipiell ist der Spielraum geringer als bei INTERN2, weil das EVU keine eigenständige Erweiterung programmiert, sondern den SAP-Standard anpasst. Das **Kontrollverhältnis** ist anders als bei INTERN1, weil die Softwaregestaltung dem Arbeitsprozess der Softwareanwendung hierarchisch nicht übergeordnet ist, sondern mit den Hierarchien der Anwendung und Abteilungsgrenzen zu kämpfen hat. Das EVU **reorganisiert** die Softwareanwendung nur dahingehend, dass es dezentrale IT-Teams in den Fachbereichen und Gestaltungsnetzwerke für die Anforderungsrunden etabliert hat. Tiefergehende organisatorische Veränderungen, wie die IT- und Fachabteilung aufzulösen und interdisziplinäre Teams zu bilden, meidet das EVU. Das EVU setzt die **Ziele** der Softwaregestaltung nur teilweise selbst. Was den softwaretechnischen Zuschnitt anbelangt, ist sie von SAP abhängig, das einen Standard anstrebt. Was die organisatorische Ausrichtung betrifft, kann das EVU selbst entscheiden, vom Status quo aus zu gestalten und nicht die komplette Organisation in Frage zu stellen. Doch wie bei INTERN1 erlaubt der Einfluss eine direkte **Partizipation** der Anwendenden, auch wenn sich hier mehrere Fachbereiche einigen müssen und damit die Fachbereiche voneinander abhängig sind. Allerdings meint ein befragter Anwender, er würde in Entscheidungsprozesse über Anforderungen nicht eingebunden, vor allem bei solchen, bei denen es um die Usability geht und darum, die Anwendung zu vereinfachen. Für ihn sind Priorisierungen nicht nachvollziehbar und was aus welchem Bereich umgesetzt oder nicht umgesetzt wird.

Ein Beispiel für den mangelnden Einfluss der Softwaregestaltung, was Reorganisation und Kontrollverhältnis aufgrund einer fehlenden prozess- bzw. fachbereichsübergreifenden Sicht anbelangt, ist der Fachbereich Netzanschluss. Dort gibt es zwar eine dezentrale Fachbereichs-IT, die sich um die Digitalisierung des Fachbereichs kümmert. Sie ist aber hierarchisch den anderen (Nicht-IT-)Teams des Fachbereichs gleichgestellt und nicht direkt bei der Bereichsleitung angesiedelt. Sie ist im Team der Regionalleitung. Diese ist disziplinarisch verantwortlich, müsste Entscheidungen fällen und Aufträge vergeben, was die Softwaregestaltung anbelangt. Die Regionalleitung macht das jedoch nicht:

> »[D]. h. Prozesse, die eigentlich waagerecht laufen, die werden dauernd durch irgendwelche Hierarchien getrennt und der Leiter sagt halt: ›Nicht mein Bier. Ich kümmere mich nicht um die Schnittstelle, weil mir tut es ja nicht weh.‹ Also, es fehlt die übergeordnete Verantwortung für einen Prozess aus Kundensicht oder aus Dienstleistersicht. Wir kucken eigentlich immer nur nach: Wie sind wir strukturiert aus der Historie?« (Product Owner Netzanschluss)

Ein weiteres Beispiel für den geringen Einfluss der Softwaregestaltung auf die Softwareanwendung aufgrund der Abteilungsgrenzen sind die unterschiedlichen Grade der Automatisierung. Unter anderem stellen die Schnittstellen zwischen Teams, die mit dem Modul Work-Management bzw. Auftragsverarbeitung arbeiten, einen Bruch dar, der nicht digital überbrückt und damit automatisiert ist. Bereiche wie Zählerwesen oder Netz-

planung haben jeweils für sich eine eigene interne Organisation. Sie haben z.B. unterschiedlich viel Budget für IT. Das erschwert ein einheitliches Vorgehen. Jeder einzelne Bereich kann sich für sich selbst optimieren und denkt erst einmal an die Veränderungen bei sich. Um den gesamten End-to-end-Prozess der Auftragsverarbeitung zu optimieren, müssten alle Nachbarabteilungen mitmachen und Teamgrenzen überwunden werden. Das Silo-Denken verdeutlicht der Umstand, dass aktuell niemand den gesamten End-to-end-Prozess Netzanschluss auswerten kann, weil er sich auf mehrere Abteilungen und Systeme erstreckt.

Doch hat sich durch die gemeinsame Anforderungsrunde zumindest die Wahrnehmung auf die Organisation und die Gestaltungsmöglichkeiten verändert, indem die Beteiligten vom gemeinsamen (Software-)Prozess aus auf die Firma blicken:

> »Ja, den Eindruck hatte ich auch, dass praktisch durch diese Software, diese alten Strukturen auf einmal feststellen: Wir haben ja was gemeinsam. Was vorher halt einfach nicht war, weil man fachlich getrennt war, weil es eben diese Fachexperten gab für Montage oder Hausanschluss oder was weiß ich, was es da alles gibt. Und jetzt haben die aber doch irgendwie so bestimmte Sachen gemeinsam oder müssen sich jetzt versuchen abzustimmen und das passt nicht mehr zu den alten Strukturen. Das ist zumindest den Eindruck, den ich habe.« (Anforderungsmanager)

Neben den organisatorischen Brüchen innerhalb und zwischen den Fachbereichen und Abteilungen beschränken die Vielzahl verschiedener Softwarelösungen den Einfluss des Arbeitsprozesses der Softwaregestaltung. Der befragte Anwendende aus dem Fachbereich Netzanschluss verwendet zwei Systeme und zwei Oberflächen, hinter denen zwei getrennte IT-Bereiche stecken, die jeweils getrennt Anpassungen vornehmen. Das heißt, der Arbeitsprozess der Softwaregestaltung in Form der Anforderungsrunde betrifft nur eine der angewendeten Softwarelösungen.

Es war eine Voraussetzung für die Etablierung des Softwaregestaltungsprozesses, dass zwischen den Fachbereichen und der IT-Abteilung keine grundsätzlichen **Konflikte** vorhanden waren (siehe die Entstehungsgeschichte der Abstimmungsrunde unter 8.3.2.2). Inhaltliche Interessengegensätze scheinen die Fachbereiche in der zentralen Abstimmungsrunde und dezentral in ihren Teams nun konstruktiv zu verhandeln.

Die Befragten sehen den **Betriebsrat** nicht als inhaltlich gestaltend, sondern intervenierend. Bei der Anforderungsrunde macht er nicht mit. Der Betriebsrat prüft vielmehr allgemein neue Lösungen, ob sie eine individuelle Leistungskontrolle ermöglichen. Bei Prozessoptimierungen muss das Management ihn einbinden. Manche aus der IT-Abteilung sehen, so ein Befragter, den Betriebsrat als Behinderung an, weil man ihm die Lösungen vorschlagen und mit ihm diskutieren muss. Für einen anderen Befragten hat er bisher in der Softwaregestaltung noch keine Rolle gespielt.

8.5.2.3. KOOP1: zentraler Standard oder dezentral individuell – Wo endet die Kooperation der EVU?

Bei der soziotechnischen Arbeitsgestaltung sind die EVU in diesem Fall größtenteils abhängig. Das liegt zum einen daran, dass mehrere EVU gemeinsam vermittelt über ein IT-DL kontinuierlich und kooperativ verhandeln, was sie als Standard und was sie indivi-

duell gestalten. Der Einfluss dieser zentralisierten Softwaregestaltung auf die Softwareanwendung beschränkt sich im Wesentlichen auf die Software selbst. Die EVU entscheiden für sich, wie sie sich organisieren. Das bedeutet in diesem Fall, von ihrem jeweiligen Status quo ausgehend und sich nicht radikal auf die Softwaregestaltung ausrichtend. Das gemeinsame Ziel, einen Standard zu gestalten, widerspricht in manchen Fällen den Zielen einzelner EVU, individuell ihre Softwareanwendung zu optimieren. Doch lässt die zentralisierte Verhandlung zu, dass EVU zumindest Teile ihrer Software unabhängig von anderen gestalten. Inhaltliche Konflikte verhandeln EVU und IT-DL zwar im gemeinsamen Anforderungsmanagement. Was die Softwaregestaltung allgemein und damit die soziotechnische Arbeitsgestaltung betrifft, sind einzelne EVU durch die Marktbeziehung abhängig vom IT-DL und den anderen EVU.

Bei KOOP1 besteht durch das **Kontrollverhältnis** keine Unterordnung der Anwendenden unter die Bedarfe der Softwaregestaltung. Sie sind vielmehr einem Softwarestandard untergeordnet: dem von SAP und dann noch dem industriespezifischen, den die EVU kooperativ gestalten. Die EVU kontrollieren die Softwareanwendung unabhängig von der Softwaregestaltung dieses Standards. Bei jenen EVU, die verstärkt dezentral selbst Software gestalten, gibt es Fälle, die den Arbeitsprozess der Softwareanwendung an die Bedürfnisse der Softwaregestaltung ausrichten.

Um zumindest Einfluss auf die Software zu haben, haben EVU im Zeitverlauf eigene Strukturen zur Softwaregestaltung ab- und wieder aufgebaut. In EVU3 und EVU2 wurden 2020 neu Stellen geschaffen, um intern Synergien zwischen mehreren Fachbereichen in der Gestaltung zu erkennen. Im EVU2 existiert ein eigenes Team, das sich mit dem Thema Digitalisierung beschäftigt (Strategien entwickeln, eigene Fachkräfte für IT-Projekte etc.). Das ist für den Digitalisierungsmanager der Firma Teil einer Wellenbewegung:

> »Also, so das Thema Make-or-Buy oder zentral, dezentral. Also, was habe ich zentral und was habe ich dezentral? Das ist so kurvenartig. So alle zehn Jahre schwingt das um. Wir müssen wieder alles dezentralisieren hin zu zentralisieren. Wahrscheinlich ist in zehn Jahren wieder alles zentral bei uns. Und, ja, ich glaube, da gibt es keine feste Kurve, wo sich das hin entwickelt.« (Digitalisierungsmanager EVU2)

Auch andere EVU verzichten auf Synergien durch die Kooperation zugunsten individueller Gestaltungswege. So hat das EVU1 eine andere Software für das Energiedatenmanagement und die Instandhaltung, um diese alleine, unabhängig von den anderen EVU, die mit dem IT-DL zusammenarbeiten, zu gestalten.

Aufgrund der abhängigen Arbeitsgestaltung ist die **Reorganisation** nur getrennt für Softwareanwendung und -gestaltung möglich. Es gibt zwar individuelle Anpassungen bei KOOP1, jedoch führen sie nur vereinzelt zu Reorganisationen: Ein EVU von KOOP1 betreut die Bereiche für Geschäfts- und Privatkundschaft nicht mehr durch getrennte IT-Teams, sondern gemeinsam durch ein IT-Team, um die Prozesse beider Bereiche zu vereinheitlichen. Die Prozesse zur Kundschaft sollen von den Softwaregestaltenden end-to-end überblickt und optimiert werden können. Der Anstoß für die Reorganisation war, dass die verschiedenen Bereiche ähnliche Anforderungen an die Softwaregestaltung gestellt haben:

> »Und da ist halt dann aufgefallen, dass es teilweise bzw. eigentlich nicht teilweise, sondern der übergreifende Teil immer der war, dass es die gleichen Anforderungen gibt, die aber letztendlich losgelöst voneinander umgesetzt worden sind oder genau der andere Fall, dass Team A irgendetwas umgesetzt hat, was bei Team B dazu führte, dass irgendwann nicht mehr funktioniert hat. Da ist man halt zum Entschluss gekommen, dass man eigentlich diese beiden Teams oder Funktionen zusammenbringen muss, um letztendlich dort Synergien zu entwickeln.« (Prozessmanager EVU3)

Hier haben Beschäftigte und Management intern die softwaretechnischen Gestaltungsmöglichkeiten erkannt und reagieren darauf mit einer Reorganisation.

Die Abhängigkeit bei der soziotechnischen Arbeitsgestaltung zeigt sich auch an den **Zielen**. Bei KOOP1 haben vor allem die kleinen EVU das Ziel, eine Standardware zu nutzen, die günstig sein soll. Allgemein behalten die EVU ihre fachlichen Abteilungen und hierarchisch gegliederten Strukturen bei. Wie auch schon bei den anderen Dimensionen des Einflusses der Softwaregestaltung auf die Softwareanwendung nutzen einzelne EVU die Möglichkeit, mehr individuell selbst zu gestalten und sowohl was die Software als auch die eigene anwendende Organisation betrifft, unabhängig zu entscheiden.

Die Partizipation erfolgt beim kooperativen Anforderungsmanagement bürokratisch reglementiert. Es ist ein formalisierter Prozess, in dem EVU im Anforderungsmanagement Anforderungen verhandeln und der Standard einen Konsens darstellt. In einem EVU prüft das Management intern noch jede Anforderung, die zum IT-DL in das Anforderungsmanagement wandert. Oder es legt z.B. das IT-Budget für Anforderungen fest und entscheidet, welche Anwendenden mitgestalten.

Der Fall ist ein Beispiel für eine abhängige soziotechnische Arbeitsgestaltung durch Softwaregestaltung, weil die **Konflikte** extern sind. Konflikte über die inhaltliche Gestaltung lösen die EVU zwar über das IT-DL im Ablauf des Anforderungsmanagements. Wenn kein Kompromiss gefunden wird, sieht die Softwarearchitektur und die Kooperationsvereinbarung vor, dass einzelne EVU unabhängig davon gestalten können. Das Kernproblem der Softwaregestaltung via Marktbeziehung (wenn auch kooperativ) bleibt bestehen: Zwischen Kooperation, Innovation und Kostenkalkül müssen EVU und IT-DL eine gemeinsame Basis finden. Die Folgen für die Gestaltung sind, dass die Kooperation beschränkt bleibt und teilweise von EVU in Frage gestellt wird.

Aus Sicht von möglichen Synergien in der Softwaregestaltung wäre es ein großer Vorteil, wenn die EVU besser kooperieren und ihre Unabhängigkeit dafür aufgeben würden. Doch ist es für das IT-DL schwierig, etwas kooperativ zu bewegen, wenn die EVU intern sich nicht über ihre IT-Strategie einig sind. Einen konkreten Plan oder Vorgehen, um eine gemeinsame Strategie zu entwickeln, gibt es aktuell nicht. Das IT-DL muss sich mit den EVU individuell zusammensetzen. Die EVU haben z.B. eigene Strategien, welche Softwarepakete sie einsetzen (auch von anderen IT-DL). So gibt es u.a. beim CRM-Einsatz keine Einigkeit im Verbund, weswegen EVU3 seinen eigenen Weg geht. Ein Mindestmaß an Kooperationsbereitschaft besteht jedoch. Dies zeigt sich im Anforderungsmanagement, wo sich die EVU zumindest in einigen strategischen Fragen, wie z.B. der Umstellung auf eine neue Version des ERP-Systems, einigen können.

Neben den Konflikten darüber, was Standard und was individuell ist, zeigt sich die Abhängigkeit der EVU in der Umsetzung der Softwaregestaltung. Das betrifft die Dau-

er, den Support und die Qualität der Umsetzung. Das hat auch Folgen für die Anwendenden, weil diese in ihrer Arbeit von einer funktionierenden Software abhängen. Für viele Befragte ist die Reaktionszeit und der Service des IT-DL nicht immer befriedigend. Das IT-DL verweist manche EVU bei Fragen an die Servicehotline oder sie müssen auf Rückrufe warten. Die EVU erleben sich als Teil einer größeren Kundschaft. Manche Anforderungen erledigt das IT-DL trotz existierender regulatorischer Fristen zu langsam. Es herrscht Misstrauen, was die Aufwandsschätzung anbelangt, weswegen einige EVU intern wieder Know-how haben, um sie prüfen zu können.

Für die **Betriebsräte** in den EVU ergeben sich durch die Zusammenarbeit mit einem IT-DL, die Projektarbeit und den stetigen Wandel der IT-Systeme verschiedene Einflussmöglichkeiten. Allerdings betrifft das weniger die inhaltliche Softwaregestaltung. Diese geht nicht darüber hinaus, eine Leistungskontrolle zu verhindern. Bei EVU4 ist der Betriebsrat dank einer Betriebsvereinbarung für IT-Projekte in diese eingebunden. Er sitzt in allen Lenkungsausschüssen und ist deshalb gut informiert. Er darf bei großen Projekten auch bei der Beraterauswahl mitentscheiden. Bei mehreren IT-Projekten hat er die Leistungs-/Verhaltenskontrolle geprüft und verhindert. In dem EVU ist der Betriebsrat auf Augenhöhe mit dem Management und fährt mit zu Führungskräftetagungen, bei denen es u.a. um die Digitalisierungsstrategie geht. Aus seiner Sicht kann er bei der Softwaregestaltung nicht mitmachen, weil ihm das nötige Wissen fehlt und er nur einen eingeschränkten internen Machtbereich hat.

Der Betriebsrat aus EVU2 meint dazu:

> »Da sind Leute, die sprechen eine Sprache. Ich kann mich mit denen nicht unterhalten. Also, ich kann mit denen ein Bier trinken. Aber fachlich… Ich kenn kein Wort von denen. Die Rede auch kein Deutsch. Das ist natürlich eine andere Welt für uns, völlig andere Welt.« (Betriebsrat EVU2)

In dem EVU sitzt der BR in einem Gremium, das sämtliche neuen IT-Systeme prüft. Laut dem Applikationsbetreuer von EVU2 muss für jedes neue IT-System eine Betriebsvereinbarung geschlossen werden. Zudem versucht der Betriebsrat von EVU2, das IT-DL zu kontrollieren. Einmal hat er mitgewirkt, einen externen Berater zu tauschen. Er sieht seine Firma in der Verantwortung, die Belegschaft vor negativem Einfluss von außen zu schützen. Gleichzeitig sieht er keinen Grund, nicht auf Wettbewerb zwischen den IT-DL zu setzen.

> »Wir haben unseren ganzen IT-Bereich in mehrere Lose verteilt und die werden ausgeschrieben und entweder kriegen sie es oder kriegen es nicht […] Deswegen: Wir nehmen den, der am besten ist, ganz deutlich.« (Betriebsrat EVU2)

Der Betriebsrat von EVU3 sagt klar, dass er die Software nicht mitgestaltet. Stattdessen hat er Obergrenzen durchgesetzt, was die Arbeitsstunden anbelangt, und auch schon Projekte ausgesetzt, weil die Belastung zu groß wurde. Aus seiner Sicht hat der Datenschützer mehr Mitspracherechte bei IT-Themen als er. Auch er konzentriert sich darauf, eine individuelle Leistungskontrolle zu verhindern.

8.5.2.4. KOOP2: Prekär-kooperativ – wieder unabhängiger vom IT-DL?

Im Gegensatz zu KOOP1 gestalten einige EVU in diesem Fall wieder unabhängig von anderen ihre Software und damit die Arbeit ihrer Anwendenden – auch wenn es, wie bei KOOP1, meist nur die Anpassung oder Erweiterung einer Standardsoftware ist. Statt zentral einen Industriestandard wie bei KOOP1 zu gestalten, gestalten manche EVU das bestehende ERP-System selbst individuell – durch eigene Programmierungen oder Softwarelösungen anderer Softwarefirmen. Doch reizen diese EVU die Möglichkeiten der Arbeitsgestaltung nicht aus, weil sie sich auf die Gestaltung der Software der Anwendenden konzentrieren und die anwendende Organisation unangetastet lassen. Wobei es wie bei KOOP1 Unterschiede gibt, wie stark sich einzelne EVU auf die Softwaregestaltung einlassen und damit auf eine individuelle und unabhängige soziotechnische Arbeitsgestaltung. Ein befragter Sachbearbeiter zeigt die für diesen Fall typischen vielfältigen Formen der Softwaregestaltungen in den EVU in einer Extremform: Er muss mit einer Cloud-Lösung von SAP arbeiten. Das hat zur Folge, dass er und sein EVU komplett abhängig sind, was die Gestaltung der Software anbelangt.

Der Einfluss auf die soziotechnische Arbeitsgestaltung ist in diesem Fall gering, obwohl einige EVU unabhängig Software gestalten. Das ist so in EVU2 (ob bei einem größeren IT-Projekt, einer dezentralen Softwaregestaltung durch den Fachbereich Marktkommunikation oder einer Softwaregestaltung in einzelnen Fachbereichen über deren IT-Koordinierenden) und ebenso in EVU3 der Fall (wo über ein zentrales IT-Team, das eine Entwicklungsplattform betreibt, dezentral Fachbereiche kleinere Apps gestalten). Warum ist der Einfluss gering? In all diesen Beispielen kontrolliert der Arbeitsprozess der Softwaregestaltung nicht jenen der Softwareanwendung. Durch die dezentrale und individuelle Softwaregestaltung wäre zwar ein anderes **Kontrollverhältnis** möglich. Die EVU nutzen es aber nicht. Sie sind intern auf die Softwareanwendung ausgerichtet. Vielmehr geht es erst einmal darum, was sich vor allem bei EVU2 zeigt, intern Strukturen wie eine Stabstelle für Projektmanagement aufzubauen, um eigenständig Software zu gestalten. Zwar sind die IT-Koordinatoren der Fachbereiche im EVU2 für größere Bereiche zuständig. Sie sind aber hierarchisch nicht so angesiedelt, dass sie fachübergreifende Themen vorantreiben oder gar Reorganisationen durchsetzen könnten. Stärker noch als bei KOOP1 zeigt der Fall, dass trotz der dezentralen Softwaregestaltung in den EVU keine grundlegende **Reorganisation** erfolgt. Die bestehende Organisation aus Abteilungen, Teams und Hierarchien bleibt unangetastet. Wie für eine Matrixorganisation typisch, betreiben die EVU temporär IT-Projektarbeit oder arbeiten kontinuierlich mit Standardsoftwarefirmen von Speziallösungen zusammen. Es geht primär darum, Software zu gestalten und weniger die Organisation der Anwendenden auf die Softwaregestaltung auszurichten. Wie auch bei INTERN1, INTERN2 und KOOP1 verwenden die EVU das Standard-ERP von SAP und dieses ergänzende Standardpakete. Sie sind damit einerseits von den **Zielen** der Softwarefirma abhängig, ein Standardprodukt profitträchtig zu verkaufen. Andererseits sind sie unabhängig, wenn es um Anpassungen und Erweiterungen dieser Standardsoftwarelösungen geht. Das nutzen sie für ihre Ziele, die eigene Organisation nicht zu stark ändern zu müssen und die Software selbst gestalten zu können. Die EVU, die wieder eigenständig Software gestalten, können eine **Partizipation** ihrer Mitarbeitenden ermöglichen, wie es für eine unabhängige Arbeitsgestaltung typisch ist. Dabei sind bei EVU2 die internen Hierarchien prägend. Die Anwendenden

können nicht unabhängig von diesen Hierarchien (und Abteilungsgrenzen) die Software und damit einen Teil ihrer Arbeit gestalten. So bespricht die Teamleitung der Marktkommunikation mit der Softwarefirma die Anforderungen, oder IT-Koordinator:innen sammeln Anforderungen aus den Fachbereichen ein.

Eine klare Abhängigkeit in der Arbeitsgestaltung erlebt der befragte Sachbearbeiter von EVU1. Er muss sich einer neuen, von SAP entwickelten Cloud-Lösung für Marktkommunikation unterwerfen. Weil es eines seiner zentralen Werkzeuge für die Arbeit ist, ist er komplett von der Softwarefirma SAP, die sie zur Verfügung stellt, abhängig. Mehr als Fehler aufnehmen und warten, bis sie die Softwarefirma behoben hat, kann er nicht.

In dem Fall verlaufen die **Konflikte** sowohl zwischen IT-DL und EVU als auch innerhalb der EVU, weil diese selbst Software gestalten. Die Konflikte zwischen IT-DL und EVU sind weniger inhaltlicher Natur als vielmehr Beziehungskonflikte, die auf unterschiedlichen Erwartungen beruhen. Die Konflikte zeigen sich vor allem daran, dass sich die beteiligten EVU und das IT-DL nicht auf eine zentrale Gestaltung einigen konnten. Diese Konflikte führten zum Ausstieg einzelner EVU aus der kooperativen Softwaregestaltung oder zur sukzessiven Rückverlagerung der Softwaregestaltung in die EVU.

Trotzdem sind die EVU noch vom IT-DL abhängig, aber weniger inhaltlich und mehr in der Umsetzung. Ein Thema ist die Priorisierung von Anforderungen beim IT-DL. Ein Befragter hat den Verdacht, dass seine Tickets vom IT-DL langsamer bearbeitet wurden, weil sein EVU eine Pauschale bezahlt und ein anderes EVU pro Ticket, wodurch der Anreiz für das IT-DL höher sei, Letzteren schneller zu bedienen. Ein anderer Befragter ist der Ansicht, dass das IT-DL den Personal-Wünschen seiner gesellschaftenden EVU[21] entgegenkommt (z.B. bekommen diese EVU ihre Wunschkandidaten für Projekte). Ein Befragter des IT-DL sagt, er behandelt alle EVU gleich, egal ob gesellschaftendes EVU oder nicht. Andere Konfliktpunkte sind, dass die EVU meinen, dem IT-DL fehle die Sicht seiner Kundschaft, es sei nicht lösungsorientiert, würde Tickets zu langsam bearbeiten und hätte zu wenige Mitarbeitende.

Innerhalb der EVU fällt auf, dass sie zwar die Möglichkeit hätten, die Konflikte zwischen Gestaltung und Anwendung eigenständig zu lösen. Jedoch zeigt sich, dass der Einfluss der Fachbereiche auf die IT-Abteilung oder der IT-Koordinierenden auf die Fachbereiche gering ist. Dies ist womöglich auch ein Grund, warum keine organisatorischen Änderungen stattfinden und sich manche befragte Softwaregestaltende zwar unzufrieden mit den bestehenden Strukturen zeigen, mit diesen aber leben müssen oder das Unternehmen verlassen (wie das zwei Befragte einige Monate nach den Interviews getan haben).

Der **Betriebsrat** ist, wie in den vorhergehenden Fallstudien auch, intervenierend tätig und hat die Möglichkeit mitzugestalten, weil die Softwaregestaltung wieder stärker intern stattfindet. Der BR von EVU2 hat eine Betriebsvereinbarung IT gemacht. Diese regelt, dass IT-Projektanträge durch den BR genehmigt werden müssen. Zudem achtet der BR darauf, dass keine privaten Daten der Mitarbeitenden zugänglich werden und durch Datenauswertungen des IT-Systems nicht auf einzelne Mitarbeitende und deren Arbeit geschlossen werden kann. Aufgrund der immer vielfältigeren IT-Landschaft würde dem BR mehr IT-Wissen bei der Arbeit helfen.

21 EVUs, die Gesellschaftende, d.h. deren Eigentum die IT-DL sind.

8.5.2.5. PAKET: abhängig von der Standardsoftwarefirma

In dem Fall sind die EVU von der Softwarefirma abhängig, was die soziotechnische Arbeitsgestaltung anbelangt. Warum? Zum einen wenden die EVU eine industriespezifische Standard-ERP-Lösung an, die sie nicht ändern können (und wollen). Zum anderen bietet die Software zwar Einstellungsmöglichkeiten. Die gibt aber der Standard vor. Selbst die wenigen EVU, die beim industriespezifischen Standard über Arbeitskreise oder einzelne Projekte mitwirken, setzen einen Standard ein, den alle anderen EVU auch einsetzen. So können die EVU nur den Arbeitsprozess der Softwareanwendung selbst kontrollieren und reorganisieren. Andererseits zeigt der Fall, dass die Einstellungsmöglichkeiten an der Standard-ERP-Software sehr umfassend sind: Wenn die EVU diesen Teil der Softwaregestaltung nutzen, kann dies organisatorische Veränderungen in der Softwareanwendung mit sich bringen, und zwar weitergehender als bei einigen EVU von KOOP1, KOOP2 oder INTERN2 – nämlich auf eine Prozessorganisation hin. Die Abhängigkeit der EVU bei der soziotechnischen Arbeitsgestaltung zeigt sich auch an den Konflikten, bei denen es weniger um die inhaltliche Softwaregestaltung geht als um die Abhängigkeit vom gelieferten Standardprodukt.

Der Einfluss der Softwaregestaltung auf den Arbeitsprozess der Softwareanwendung ist, was das **Kontrollverhältnis** anbelangt, in diesem Fall gering. Die EVU können nur die Softwareanwendung kontrollieren und weniger den Gestaltungsprozess, der außerhalb der EVU stattfindet. Vielmehr prägt die fertige Standardsoftware die Anwendung der EVU. Mit dem geringeren Gestaltungsspielraum (vom Management bis zu den Anwendenden in den EVU) geht eine Unterordnung der Anwendung unter die Standardsoftware einher. Sie dient als Referenz und zur Disziplinierung von Veränderungswünschen seitens der EVU – ob der eigenen Anwendenden oder der Kundschaft. So nutzt EVU4 den Standard dafür, Wünsche seiner Kundschaft zu beschränken, z.B. indem es auf die Grenzen der Abrechenbarkeit komplexer Erzeugungsanlagen durch die Software verweist. Eine Extremform der Unterordnung der Anwendung unter die Standardsoftware ist, wenn die Softwarefirma oder EVU die ERP-Software als Anwendungsplattform einsetzen. Aufgrund des Funktionsumfangs und der hohen Automatisierung der Software steckt so viel Wissen in ihr und die Aufgaben, die sie erledigt, sind so umfangreich, dass sie eine Anwendungsplattform darstellt. Jeden, der die Software besitzt (wie bspw. die Softwarefirma), trennt nicht mehr viel von einem EVU:

> »Also theoretisch können wir auch einfach unseren eigenen Stromversorger aufmachen.« (Programmierer Softwarefirma)

So liegt es nahe, dass die Softwarefirma Business Process Outsourcing (BPO) anbietet.

> »Es gibt aber auch Unternehmen, gerade so im Bereich der kleineren Lieferanten, so Start-ups, wo das Unternehmen eigentlich aus einem Geschäftsführer, paar Vertrieblern und Marketingleuten besteht. Und die gesamte Abwicklung: Abrechnung, Marktkommunikation, EDM[22], Buchhaltung etc. pp. läuft dann oder geht dann über unsere

22 Energiedatenmanagement

[BPO-Firmeneinheit]. Da spielen wir sozusagen Stadtwerk.« (Lösungsarchitekt Softwarefirma)

Manche EVU bieten anderen den Betrieb der industriespezifischen ERP-Software an. In einem Fall bedeutet das, dass die anwendenden EVU die Einstellungen, welche das betreibende EVU an der Software vorgenommen hat, 1:1 übernehmen. Das heißt, es richtet sich in der Anwendung nicht nur an der Standardsoftware aus, sondern auch an den Standardeinstellungen des betreibenden EVU.

Von den Extremfällen abgesehen, haben die EVU Einfluss auf die Einstellungen am Standard und ob sie den Standard einsetzen. In einem Fall hat sich ein Fachbereich gegen ein Modul der Standardsoftware entschieden (auch wenn der befragte IT-Leiter dagegen war). Um Einstellungen vorzunehmen, brauchen EVU Zeit und das entsprechende, qualifizierte Personal. Nicht in allen EVU ist es möglich, sich intern optimal auf die Software auszurichten, weil die Softwarefirma die Updates zu kurzfristig ausliefert. Das heißt, auch hier sind die EVU abhängig von der Softwarefirma. Aus Sicht eines Teamleiters (EVU2) sind sie zu sehr mit den regelmäßigen Updates der Software beschäftigt und haben keine Zeit und kein Geld für Optimierungen. Er wird häufig von Kolleg:innen gefragt, ob er etwas an der Software optimieren könnte. Das würde aus seiner Sicht viel bringen in puncto Effizienz. Ein anderer Befragter, der Teamleiter und Sachbearbeiter zugleich ist, hat auch keine Zeit für Optimierungen, weil sich regelmäßig die Regulierung verändert. Manchmal kommt das Update der Softwarefirma für Änderungen erst einige Tage vor der Frist, in der eine Regulierung in der Software umzusetzen ist. Sie ist dann noch nicht komplett fehlerfrei und so werden Fehler auch noch Wochen nach der Frist behoben. Er hat keine Zeit für Tests und ist froh, wenn er die Arbeit schafft, obwohl sie mit jedem neuen Gesetz mehr wird.

Einige der befragten EVU, welche selbst Einstellungen an der Software vornehmen, konstatieren einen Mangel an Positionen, die übergreifende Veränderungen an der Software vornehmen können:

»Oh, sehr viele. Ja, erst mal… Also aktuell ist es so, dass wir immer noch den Wunsch haben nach einem übergeordneten Projektleiter, der eben genau solche Aufgaben übernimmt. Aktuell versuchen wir das so ein bisschen zu machen, so nebenbei noch, was schwierig ist, weil wir ja auch gar nicht weisungsbefugt sind.« (Anwendungsbetreuung EVU3)

Um solche Einstellungen vornehmen zu können, gibt es bei mittelgroßen EVU in begrenztem Umfang **Reorganisationen** in Form von neuen Stellen und einer Prozessausrichtung. Es entstehen neue Aufgaben für Führungskräfte und teamübergreifende Positionen, um für teamübergreifende Prozesse Einstellungen an der Standardsoftware vorzunehmen. So hat EVU3 erst vor ein paar Jahren angefangen, Prozesse aufzunehmen, und die Projektarbeit professionalisiert, d.h. Stellen geschaffen für Projekt- und Prozessmanagement.

»Ich kann mich noch erinnern… vor ein paar Jahren angefangen habe, da war das so: ›Prozesse, was ist das? Das habe ich ja noch nie gehört. So was braucht doch kein

> Mensch.‹ Und jetzt ist es unumgänglich […] [H]eute machen wir alles in Prozessen. Und so ist das auch mit Projekten. Früher: ›Projekte, was ist das?‹« (Gruppenleiter EVU3)

Auch EVU2 hat festgestellt, dass Projekte mittlerweile häufig abteilungsübergreifend sind. EVU5 hat eine größere Reorganisation in Richtung Prozessorganisation vorgenommen. Die Organisation teilt sich nicht mehr nach Abrechnungsarten auf (Tarifkundschaft, Sonderkundschaft, Einspeiser etc.), sondern nach Prozessschritten, die sämtliche Abrechnungsarten betreffen (Marktkommunikation, Abrechnung, Zahlungsverkehr). In Terminen der betroffenen Bereiche optimiert das EVU die Gesamtprozesse.

Jedoch steht das **Ziel** der Standardsoftwaregestaltung über den individuellen Wünschen einzelner EVU. Das ist für jene EVU in Ordnung, die vor allem für wenig Geld Software einsetzen wollen. Einerseits erwarten die EVU ein funktionierendes System, mit dem sie ihre Arbeit erledigen können und welches die Regulierung abdeckt:

> »Dadurch, dass wir ein Lizenz-System haben, ist der Softwareanbieter dafür verantwortlich, die regulatorischen Bedingungen im System umzusetzen. Also, wenn man einen Software-Dienstleister darauf hinweisen muss, dass er eine regulatorische Umsetzung umsetzen muss, hat er seinen Job verfehlt. […] [Das] System ist ein rein energiewirtschaftliches System.« (Gruppenleiter EDM EVU4)

Andererseits hat das Standardpaket Nachteile, weil die Softwarefirma Anforderungen nur umsetzt, wenn sie auch andere EVU betreffen. Eine individuelle Entwicklung wäre schneller in der Umsetzung:

> »Also, es ist schon einfacher für uns, wenn wir die Entwicklungsveränderungen im Haus vornehmen können und das dann einfach umgesetzt wird, als wenn wir darauf angewiesen sind, dass es anderen auch noch so geht. Oder dass erst mal aufwendige Beschreibungen und Tests durchgeführt werden müssen, damit der Softwareanbieter erkennt, wo die To-dos liegen.« (Betriebsrat EVU6)

Einige EVU haben nicht nur das Ziel einer möglichst günstigen Software, sondern wollen auch individuelle Ideen umsetzen und möglichst günstige, individuelle Prozesse haben. Laut dem Gruppenleiter aus EVU3 würden individuelle Entwicklungen helfen, interne Prozesse zu optimieren. Ein Befragter aus EVU1 meint, sie haben gar nicht die Ressourcen, um von einem Standard abzuweichen, und hätten deshalb sämtliche individuellen Entwicklungen beseitigt. Das sieht auch der IT-Leiter aus EVU2 so.

In dem Fall **partizipieren** nur ausgewählte EVU und Fachleute u.a. über Arbeitskreise an der Softwaregestaltung. Branchenfachleute haben ein größeres Gewicht bei der Frage, wie Anforderungen zustande kommen, und weniger die Anwendenden mit ihren individuellen Anwendungskontexten. Partizipation ist nur bei den Einstellungen möglich. Die überlassen die Anwendenden aber lieber anderen:

> »Die Anwender sind dann einfach nur die, die sagen: ›Ich bin hier zum Abrechnen da und mich interessiert eigentlich nicht, wie das System funktioniert, wie es funktionieren sollte. Für mich muss es laufen und wenn es eine Neuerung gibt, dann muss es mir

> jemand sagen.‹ Also, da muss quasi schon alles vorbereitet sein. Am besten, wenn es eine Neuerung gibt, gibt es eine vorbereitete Doku, die beschreibt, welchen Knopf ich wann drücken muss.« (Anwendungsbetreuung EVU3)

Die Abhängigkeit bei der soziotechnischen Arbeitsgestaltung durch Softwaregestaltung drückt sich in dem Fall vor allem für kleinere EVU darin aus, dass bei inhaltlichen **Konflikten** die Softwarefirma die Oberhand hat. Für viele stellt sich also weniger die Frage, was in den Standard aufgenommen wird. Wenn sie den Standard so nicht wollen, verwenden sie eine andere Software. Neben inhaltlichen Konflikten geht es vielmehr um Leistungen, die über die Softwarelizenz hinausgehen und die EVU erwarten. Wie schon bei KOOP1 und KOOP2 kritisieren EVU die Servicequalität bzw. erwarten einen bestimmten Service: Erstens nehmen die EVU die formale Kommunikation über Ticketsysteme als ineffizient wahr. Zweitens erwarten die EVU, dass die Softwarefirma mehr von sich aus über Möglichkeiten informiert, die die Software bietet. Sie haben intern kein Wissen über die Möglichkeiten der Softwaregestaltung und sind abhängig davon, dass die Softwarefirma darüber informiert.

Wie die EVU haben die **Betriebsräte** wenig Einfluss auf die Softwaregestaltung. Die Konflikte betreffen die Implementierung der Standardsoftware. In EVU6 gibt es eine Betriebsvereinbarung für Softwareanwendung und deren Einführung. Bei IT-Projekten stellt das Management dem Betriebsrat »meistens« (Betriebsrat EVU6) den Ablauf des Projekts und das Schulungskonzept für die Anwendenden vor. Bei der Umstellung auf eine andere ERP-Software (was gerade diskutiert wird) käme es aus Sicht des Betriebsrates zu einer Änderung an der Organisation und dieser ERP-Wechsel wäre damit mitbestimmungsrelevant.

> »Und von daher alles, […] was zu einem auch an Einführung Software bedingter Mittel oder der Digitalisierung der Projekte angeht, unterliegt auch der Mitbestimmung, so dass wir sowieso in letzter Konsequenz immer sagen müssen: ja oder nein. Und da haben wir es einfach für uns so eingeführt, dass der Betriebsrat von Anfang an bei Digitalisierung des Projektes mit dabei ist, damit ich nicht als Bremsklotz hinterher fungiere, sondern als Mitgestalter.« (Betriebsrat EVU6)

Wie weit diese Mitgestaltung geht, konnte das Interview nicht vertiefen.

8.5.2.6. KOOP3: IoT Ko-Produktion – abhängige Standardgestaltung, unabhängige Modulgestaltung

Der Fall ist ein Beispiel dafür, dass Softwaregestaltung nicht immer relevant sein muss für die soziotechnische Arbeitsgestaltung eines Anwendungsbereichs. Zwar liegt in dem Fall die zentralisierte Gestaltung einer Standard-IoT-Software vor. Jedoch hat die dadurch gegebene Abhängigkeit in der Softwaregestaltung geringe Folgen für die soziotechnische Arbeitsgestaltung. Denn es existiert kein Arbeitsprozess der Softwareanwendung. Es ist schlicht keine Anwendungssoftware des täglichen EVU-Geschäfts. Vielmehr stellen die IoT-Software und die mit ihr verbundenen Sensoren eine Erweiterung der bestehenden Dateninfrastruktur innerhalb der EVU dar, als dass sie komplizierte Geschäftsprozesse abbilden.

So spielt beim Einfluss weder das **Kontrollverhältnis** noch die **Reorganisation** der Softwareanwendung eine Rolle. Bei den **Zielen** besteht zwar eine Abhängigkeit zwischen IT-DL und EVU in Bezug auf die IoT-Software. Die Softwarefirma will vor allem ein skalierbares Produkt haben. Doch besteht ein gemeinsames Interesse mit dem IT-DL, neue Anwendungsfelder für IoT zu erschließen und auch außerhalb der Softwarefirma Module entwickeln zu können. Letztendlich lässt die Architektur eine dezentrale Entwicklung in Form von Modulen ebenso zu, wie die IoT-Standardlösung einfach nur ohne tiefergehende IT-Kenntnisse und nach erfolgreicher Implementierung anzuwenden.

Bei der **Partizipation** liefern Expert:innen, die bei der Implementierung der IoT-Lösung mitmachen, die Anforderungen und entwickeln dadurch neue Ideen für Erweiterungen. Das IT-DL gibt die Anforderungen aus Implementierungsprojekten an die Softwarefirma weiter.

Die **Konflikte** bestehen in dem Fall bei der abhängigen Softwaregestaltung zwar zwischen mehreren Organisationen. Doch lösen sie die Beteiligten durch eine kooperative Arbeitsweise. Zudem sind nicht viele Anwendende oder Teams betroffen, weswegen es weniger Konflikte bei der Gestaltung gibt. Nur ausgewählte Mitarbeitende in den EVU nehmen temporär an der Implementierung teil und nutzen dann die Daten aus den Sensoren in ihrer Arbeit. Es gibt jedoch durch die Konstellation angelegte, tiefergehende Konfliktpotenziale, die sich allerdings noch nicht manifestiert haben: Es besteht ein Wettbewerb zwischen IT-DL und Softwarefirma. Denn sowohl Softwarefirma als auch IT-DL führen Implementierungsprojekte durch und das IT-DL bietet den EVU die Einbindung der Daten in deren bestehende Softwarelandschaft an, programmiert selbst Module und verkauft sie.

Der **Betriebsrat** spielt nur beim Thema individuelle Leistungskontrolle und als Stakeholder, der mitgenommen werden sollte, eine Rolle. Das EVU1 hat den Betriebsrat eingebunden und hatte am Anfang der Implementierung einen »relativ intensiven Austausch« (Teamleiter IT). Es ging u.a. darum, dass Daten erhoben werden, ohne dass Mitarbeitende mehr an einen bestimmten Ort fahren müssen, d.h., ihre Tätigkeit hat sich verändert. Der Befragte von EVU2 betont, dass der Betriebsrat in die Projekte eingebunden wird, vor allem wenn die Software Mitarbeiterdaten sammelt. Das IT-DL erklärt im Normalfall dem Betriebsrat der EVU in einem ersten Termin, was gemacht wird. Das reicht meistens aus. Nur in einem Fall, in dem Monteur:innen dauerhaft Daten zur Netzvermessung erhoben haben, kam es zu tiefergehenden Gesprächen, die aber zu einer Lösung führten.

8.5.2.7. STARTUP: Primat der Softwareentwicklung

Der Fall ist ein typisches Beispiel für den Primat der Softwareentwicklung: für eine unabhängige, soziotechnische Arbeitsgestaltung durch Softwaregestaltung. STARTUP gestaltet eine individuelle Software, und zwar anders als in anderen Fallstudien nicht ausgehend von einer Struktur, die historisch bedingt rein auf die Softwareanwendung ausgerichtet ist. Stattdessen dient die gesamte Organisation dazu, eine Software für den Anwendungsbereich zu gestalten. Die Softwareanwendung ist all das, was die entwickelte Software nicht erledigen kann. Wobei sich iterativ mit der Software die Anwendung verändert. So hat allein STARTUP Einfluss auf Software und Organisation.

Der unabhängige Einfluss zeichnet sich durch ein **Kontrollverhältnis** aus, bei dem die Anwendung klar der Softwaregestaltung untergeordnet ist und keine Hindernisse z.B. durch Hierarchien bestehen, die Absprachen mit der Anwendung notwendig machen würden. Durch den Primat der Softwaregestaltung gab es vorher keine Anwenderorganisation ohne Software, sondern die Arbeit der Anwendenden ergibt sich seit Gründung der Firma daraus, was für eine Software existiert und wie sie die Organisation weiterentwickelt. Größere **Reorganisationen** sind aus Sicht der Softwaregestaltung nicht notwendig, weil die Organisation von Beginn an auf diese ausgerichtet ist. Kleinere Veränderungen im Wechselspiel von Software und Anwendung(-sorganisation) finden aber statt. Diese Möglichkeit eigenständig nutzen zu können, zeichnet ja die unabhängige, soziotechnische Arbeitsgestaltung aus. Was noch typisch für eine unabhängige soziotechnische Arbeitsgestaltung ist: STARTUP kann sowohl die **Ziele** der Softwaregestaltung als auch jene der Softwareanwendung festlegen. Primär kann STARTUP dadurch Software für die eigenen Zwecke gestalten und zugleich auch noch sekundäre Ziele verfolgen. Sekundär hat STARTUP eine Kommodifizierung der Software im Blick. Sie stellt einen Teil der Software als White-Label anderen Firmen zur Verfügung. An der Gestaltung **partizipieren** nur interne Mitarbeitende via Kreise oder Chats. Auch wenn Anwendende Tickets aufgeben können oder bei Tests mitmachen: Diese direkte Teilnahme wurde von keiner befragten Person als wesentlicher Beitrag genannt. Die Anwendenden sind aus der Softwaregestaltung größtenteils ausgeschlossen. Das liegt aber an der Entscheidung von STARTUP, dass sie deren Input für die Gestaltung nur in geringem Umfang als notwendig erachten.

Was die soziotechnische Arbeitsgestaltung durch Softwaregestaltung anbelangt, gibt es wenig **Konflikte**, weil die Organisation auf Interdisziplinarität, Softwareentwicklung und ein gemeinsames Ziel ausgerichtet ist. Die Anwendenden waren und sind von Anfang an klar hierarchisch untergeordnet und es gibt auch keinen Betriebsrat, der diese vertreten würde. Wenn es Konflikte gibt, sind alle intern, d.h. allein durch STARTUP lösbar und betreffen die Umsetzung. So gibt es z.B. einen Konflikt zwischen guten Ideen (d.h. offenen Anforderungen) einerseits und der Reduktion der Komplexität der Software, fehlenden Fachkräften und begrenztem Budget andererseits.

8.5.3. Zusammenfassung

Die Softwaregestaltung gestaltet die Software und einen Teil der anwendenden Organisation. Dabei unterscheiden sich die Fallstudien darin, ob die EVU unabhängig oder abhängig jenen Teil von Arbeit und Organisation ihrer Softwareanwendenden gestalten können, der auf die Softwaregestaltung zurückgeht. Die Zusammenfassung ordnet die einzelnen Fälle den Typen einer unabhängigen oder abhängigen soziotechnischen Arbeitsgestaltung zu. Sie fasst die Ergebnisse zusammen – auch zur Rolle des Betriebsrats – und diskutiert am Schluss zwei Thesen zum Einfluss der Softwaregestaltung auf die Softwareanwendung: Inwiefern können Anwendende direkt partizipieren? Wie würde eine reorganisierte Prozessorganisation aussehen, welche die Möglichkeiten der Softwaregestaltung für einen abteilungsübergreifenden Prozess ausschöpft?

8.5.3.1. Einordnung der Fälle zwischen abhängig und unabhängig

Die Fallstudien INTERN1, INTERN2 und STARTUP können dem Typ einer unabhängigen soziotechnischen Arbeitsgestaltung der Softwareanwendung und KOOP1, KOOP2, KOOP3 und PAKET jenem einer abhängigen zugeordnet werden. Doch zeigen die Falldarstellungen, dass sie nicht immer klar einem der diametral-gegensätzlichen Typen entsprechen und sich teilweise abhängige und unabhängige Formen mischen.

Weil STARTUP, INTERN1 und INTERN2 unabhängig in der soziotechnischen Arbeitsgestaltung durch die Softwaregestaltung sind, haben sie sowohl Einfluss auf die Software als auch die Organisation von Anwendung und Entwicklung. Doch schlägt sich diese Unabhängigkeit in den Fallstudien nicht in gleicher Weise auf das Verhältnis der Arbeitsprozesse von Softwaregestaltung und Softwareanwendung nieder:

- Die Organisationen können eigenständig die Kontrollverhältnisse zwischen den Arbeitsprozessen bestimmen. Doch ist nur bei STARTUP der Arbeitsprozess der Softwaregestaltung klar führend und die Softwaregestaltung entscheidet allein darüber, wie die Softwaregestaltung und wie die Anwendenden arbeiten. Bei INTERN2 hingegen haben die Anwendenden ihre eigenen Führungskräfte mit eigenem IT-Budget, über das sie verfügen. Bei INTERN1 treibt die Softwaregestaltung in kleinen Schritten Veränderungen im anwendenden Fachbereich auch gegen Widerstände voran.
- Es wäre eine wechselseitige Reorganisation von Softwaregestaltung und Softwareanwendung möglich. STARTUP ist bereits von Anfang an auf die Softwareentwicklung ausgerichtet, weswegen von Beginn an die Organisation den Arbeitsprozess der Softwareanwendung so verändert, wie es aus Sicht der Softwaregestaltung sinnvoll erscheint. Bei INTERN2 findet keine Reorganisation der direkten Anwendung statt. In den anwendenden Fachbereichen existieren zu den bestehenden Strukturen ergänzende Rollen wie Product Owner. Befragte der Fallstudie sehen in Zukunft eine Reorganisation aller beteiligten Fachbereiche, um die Möglichkeiten übergreifender Softwaregestaltung verwirklichen zu können. Wie das aussehen könnte, skizziert 8.5.3.4 weiter unten. INTERN1 hat sich reorganisiert, weil die mobilen Endgeräte die Position der Meister ersetzt haben, und auch dort gibt es im anwendenden Fachbereich extra ergänzende Stellen für die Softwaregestaltung.
- Die anwendenden Organisationen können sowohl die Ziele der Softwaregestaltung als auch der Anwendung festlegen. INTERN1 und INTERN2 entscheiden sich dafür, vom Status quo aus inkrementell Software und Softwareanwendung weiter zu gestalten. STARTUP hat sich für den Primat der Softwareentwicklung entschieden und dafür, zusätzlich einen Teil der Software anderen Organisationen als Standard zu verkaufen.
- Die Organisationen können selbst entscheiden, wer mitgestaltet. Jedoch bedeutet dies nicht, dass damit die Anwendenden alle mitgestalten dürfen. INTERN1 und INTERN2 beziehen ausgewählte Anwendende ein, wobei diese dann nicht das letzte Wort darüber haben, ob das EVU die Anforderung genau so umsetzt. Bei STARTUP liefern die Softwaregestaltenden maßgeblich den Input für Anforderungen und die Anwendenden geben Feedback zu fertigen Umsetzungen oder melden Fehler.

In den Fällen PAKET, KOOP3, KOOP2 und KOOP1 sind die EVU abhängig in der soziotechnischen Arbeitsgestaltung der Softwareanwendung. Softwarefirmen und IT-DL gestalten die Software. Beim Einfluss auf die beiden Arbeitsprozesse zeigt sich die Abhängigkeit auf vierfache Weise:

- Wegen des Kontrollverhältnisses der beiden Arbeitsprozesse zueinander können die anwendenden EVU nur die Softwareanwendung kontrollieren. Das bedeutet bei KOOP1, KOOP2 und PAKET für die EVU eine Unterordnung unter eine Standardsoftware. Bis auf einzelne EVU von KOOP1, KOOP2 und KOOP3 kontrollieren die Arbeitsprozesse der Softwaregestaltung IT-DL oder Softwarefirmen.
- Die EVU können nur die Softwareanwendung reorganisieren. Das nutzen EVU unterschiedlich. So berichten Befragte eines EVU von PAKET darüber, dass sie sich organisatorisch immer mehr an Prozessen ausrichten und weniger nach fachlicher Spezialisierung.
- Die EVU können nur die Ziele der Softwareanwendung festlegen, z.B. welche Rechnungen sie noch manuell prüfen bzw. wie hoch die Automatisierung ist. Die Softwarefirma entscheidet unabhängig z.B. darüber, was Teil des Standards wird. Doch können über die IT-DL die EVU von KOOP1 und KOOP2 über die Ziele der zentralen Softwaregestaltung mitentscheiden und es ist vor allem bei KOOP1 von ihrer Kooperation abhängig, wie umfassend der Standard wird und als solcher auch noch anderen EVU angeboten werden kann.
- Bei PAKET und KOOP3 entscheidet die Softwarefirma, welche EVU mitgestalten und welche Anforderungen in den Standard einfließen. Ideen und Ausarbeitungen von Anforderungen kommen zu einem großen Teil von Fachleuten. Die Abhängigkeit bei KOOP1 und KOOP2 ist insofern abgeschwächt, weil die EVU Gesellschaftende des IT-DL sind und auch strategische Fragen der Softwaregestaltung mitentscheiden. Einzelne Anwendende der EVU können aber nur vermittelt über formale Prozesse Anforderungen aufnehmen (z.B. über das Ticketsystem) und es gibt ein zentrales Anforderungsmanagement, das über Anforderungen abstimmt (bei KOOP2 betrifft das nur einen kleineren Teil der Softwaregestaltung).

KOOP1 und KOOP2 gehören zwar grundsätzlich zum Typ abhängig. Jedoch gestalten einzelne EVU ihre Software unabhängig vom IT-DL oder der Softwarefirmen, Anwendende können direkt partizipieren und sie setzen ihre eigenen Ziele durch, indem sie Teile der Software individuell gestalten. Bei KOOP1 gibt es einen Bereich in einem EVU, der sich reorganisiert, um besser Software gestalten zu können. Ein EVU von KOOP2 führt eigenständig Softwaregestaltung durch, z.B. in Form von Projekten. Allerdings reorganisiert sich dieses EVU weniger, als dass es vielmehr die bestehende Organisation um zusätzliche Stellen ergänzt. Bei KOOP3 ist die IoT-Software keine Anwendungssoftware, sondern liefert Daten, die andere Anwendungen verarbeiten. Daher betreffen die Folgen weniger die Gestaltung eines Anwendungsbereiches bzw. die Arbeit/Organisation der Softwareanwendung und mehr die Gestaltung einer Software.

Die Tabelle 18 gibt einen Überblick über Kontrollverhältnis, Reorganisation, Strategie und Partizipation in den einzelnen Fallstudien:

Tabelle 27: Einfluss von Softwaregestaltung auf Softwareanwendung

Fall	Typ	Kontrollverhältnis Softwareanwendung (SA) – Softwaregestaltung (SG)	Reorganisation SA wegen SG	Ziele SG für SA	Partizipation SA an SG
INTERN1	unabh.	Phasenweise Kontrolle SA durch SG	teilweise Reorg.	SG für Status quo	ausgewählte Anw., moderiert
INTERN2	unabh.	Kontrolle SA und SG getrennt	Teilgestaltung je Fachbereich	SG für Status quo	ausgewählte Anw., moderiert
KOOP1	abh./unabh.	EVU: Unterordnung SA unter Standard, teilw. Kontrolle SG und SA	nur vereinzelt Reorg.	Standardware + koop. SG für Status quo	bürokratisch, moderiert, verhandelt zwischen EVU
KOOP2	abh./unabh.	EVU: Unterordnung SA unter Standard, teilw. Kontrolle SG und SA	keine Reorg.,	Standardware + intern SG für Status quo	bürokratisch in EVU und zwischen IT-DL und EVU
PAKET	abh.	in EVU: Unterordnung SA unter Standard	ggf. Reorg. für übergreifende Einstellungen am Standard	Standardware + Status quo	ausgewählte EVU und Fachleute
KOOP3	abh./unabh.	keine Anwendersoftware, Kooperation	keine Reorg.	Standardware, neue Anwendungsfelder	Fachleute vermittelt über IT-DL
STARTUP	unabh.	Unterordnung SA unter SG	Optimierung SG	Primat Softwareentwicklung + Standardware	verhandelt in Kreisen und (Chat-)Foren

Was die Konflikte zwischen Softwaregestaltung und Softwareanwendung anbelangt, sind sie bei den Fällen vom Typ unabhängig intern und damit durch die Organisation selbst kontrollierbar. Bei INTERN1 besteht der Konflikt vor allem darin, dass der Fachbereich gern selber die Hoheit über die Softwaregestaltung hätte. Aktuell hat das letzte Wort der Haupt-Product-Owner aus der IT-Abteilung. Bei INTERN2 mussten die Fachbereiche inkl. der IT-Abteilungen erst bereit zur Kooperation sein und nun verhandeln die Beteiligten viele Themen in der gemeinsamen Anforderungsrunde. Bei STARTUP haben die Befragten keine Konflikte genannt (leider fehlt die Perspektive der Anwendenden aus dem Fall, weil keine Interviews mit ihnen vorliegen).

In jenen Fällen, in denen die EVU in der soziotechnischen Arbeitsgestaltung der Softwareanwendung abhängig sind, bestehen die Konflikte vor allem zwischen Organisationen. Bei KOOP2 haben Konflikte dazu geführt, dass ein EVU die Kooperation verlassen hat und andere EVU die Softwaregestaltung nun wieder selbst in die Hand nehmen. Diese Aufkündigung der Kooperation hatte die Folge, dass z.B. bei einem der nun wieder

selbst gestaltenden EVU von KOOP2 die Konflikte nun intern anfallen. KOOP1 hat ein Set an Abläufen, Rollen und Beziehungen institutionalisiert, welche helfen, auf Konflikte zu reagieren und stetig Erwartungen abzugleichen. Sowohl bei KOOP1, KOOP2 und PAKET gibt es Konflikte, was die Servicequalität anbelangt, und z.B. bei PAKET finden anwendende Fachbereiche aus den EVU, dass die Softwarefirma Fehler zu langsam behebt. Bei KOOP3 bestehen weniger Konflikte zwischen Anwendung und Gestaltung und wenn überhaupt, dann innerhalb der Gestaltung, weil IT-DL und Softwarefirma miteinander konkurrieren.

8.5.3.2. Inwiefern handelt es sich um einen intervenierenden Betriebsrat?

In den Fallstudien nimmt der Betriebsrat eine intervenierende Rolle ein. Eine direkte Mitgestaltung im Sinne einer Mitwirkung an Konzepten findet nicht statt. Vielmehr verhindert er eine individuelle Verhaltenskontrolle, lässt sich über den Stand in IT-Projekten informieren, entscheidet bei der Auswahl des IT-DL mit oder setzt durch, dass die Beschäftigten bei der Softwaregestaltung einbezogen werden.

INTERN1 hat die Forderung des Betriebsrates nach einer stärkeren Partizipation der Monteur:innen umgesetzt. Weil das EVU Software intern entwickelt, hat der BR bereits im Prozess der Entwicklung die individuelle Leistungskontrolle verhindert und hat diese nicht wie bei Standardsoftwareprodukten erst nachträglich ausschalten lassen. Er wird durch die Teilnahme an den entsprechenden Sitzungen ebenso informiert wie die anderen an der Softwareentwicklung Beteiligten. Bei INTERN2 macht der Betriebsrat nicht bei der Anforderungsrunde mit. Der Betriebsrat prüft vielmehr allgemein neue Lösungen, ob sie eine individuelle Leistungskontrolle ermöglichen. Die einen Befragten sehen ihn als Hilfe an, wenn er frühzeitig eingebunden, überzeugt und partnerschaftlich mit ihm umgegangen wird. Andere nehmen ihn als Hindernis wahr.

Bei KOOP1 hat im EVU4 der Betriebsrat eine Betriebsvereinbarung für IT-Projekte, die ihm garantiert, dass er in allen Projekt-Lenkungsausschüssen sitzt und gut informiert ist. Teilweise entscheidet er bei der Beraterauswahl mit. Wie in den anderen Fällen prüft er, ob eine Software eine individuelle Leistungskontrolle ermöglicht. Er sieht sich zwar auf Augenhöhe mit dem Management. Eine inhaltliche Gestaltung der Software ist für ihn aufgrund fehlenden Wissens und eingeschränkten internen Machtbereichs nicht möglich. Für den Betriebsrat aus EVU2 sprechen die ITler eine andere Sprache, die er nicht spricht. Auch im EVU3 gestaltet der BR nicht mit. Er hat dafür gesorgt, dass die Stunden bei Projekten begrenzt werden, damit die Mitarbeitenden nicht zu viel arbeiten, und verhindert eine Leistungskontrolle. Bei KOOP2 gibt es eine Betriebsvereinbarung für IT-Projekte. Der BR achtet darauf, dass das Management keine Datenauswertung auf Ebene der Mitarbeitenden machen kann. Für eine Mitgestaltung an der Software fehlt ihm das Wissen.

Der Betriebsrat eines EVU des Falls PAKET kritisiert, dass von den Mitarbeitenden zu viel Learning by Doing verlangt wird. Sein EVU sollte wieder mehr Schulungen anbieten. Auch er hat eine Betriebsvereinbarung für IT-Projekte und wird über deren Ablauf und die in deren Zuge stattfindenden Schulungen informiert. Er sieht sich als Mitgestalter. Doch müssten noch Nachfragen gestellt werden, ob er konkret über die Auswahl von Softwarepaketen mitentscheidet oder Softwareänderungen vorschlägt.

Bei KOOP3 verhindert der Betriebsrat in den EVU eine Leistungskontrolle. In einem Fall wurde er am Anfang eines IoT-Projektes eingebunden und es gab einen intensiven Austausch. Aus Sicht des IT-DL reicht es im Normalfall, wenn er den BR über das Projekt und dessen Stand informiert.

Im STARTUP gibt es keinen Betriebsrat.

8.5.3.3. Nehmen die Anwendenden an der Gestaltung teil?

Anstatt dass die Anwendenden direkt an der Gestaltung der Software beteiligt sind, sind es meist ihre Repräsentant:innen oder energiewirtschaftliche Expert:innen.

Zwar haben in manchen Fällen sämtliche Anwendende theoretisch in bestimmten Phasen die Möglichkeit, Anforderungen zu stellen oder Tickets aufzunehmen. Es ist auch die Rede davon, dass IT-Abteilungen, IT-DL oder Softwarefirmen Anforderungen umsetzen, die viele Anwendende aus den EVU benötigen (KOOP1, INTERN2, PAKET). Doch nehmen letztendlich nur die wenigsten Anwendenden teil und die finale Entscheidung liegt nicht bei ihnen, sondern u.a. in Gremien, Anforderungsrunden oder bei Product Ownern. Bei STARTUP sind die Anwendenden aus den Kreisen, die über Anforderungen sprechen, ausgeschlossen. Bei INTERN1 und INTERN2 benennen Befragte den Konflikt zwischen Unternehmen und Anwendenden explizit, wenn es um Anforderungen geht. Wobei sich das Management letztendlich durchsetzt. Des Weiteren sind sie von den Softwarefachleuten abhängig, um überhaupt etwas über die Möglichkeiten der Softwaretechnik zu wissen und die Kosten abschätzen zu können. Es bleibt eine Ausnahme, dass einzelne Anwendende ihre spezifischen Anforderungen durchbekommen, wie z.B. bei INTERN1, wo es Befragte explizit erwähnt haben, oder bei INTERN2, wo der Product Owner Interviews mit Anwendenden führt, um deren Bedarfe zu ermitteln.

Inwiefern Anwendende mitgestalten, lässt sich anhand von drei Variablen strukturieren: dem Zugang zur Gestaltung (direkt oder über Repräsentant:innen), der Rolle der Expert:innen und wie sich Unternehmens- und Anwenderinteressen zueinander verhalten. Es lassen sich drei Typen bilden:

Bei der direkten Partizipation agieren die Anwendenden selbst als Expert:innen (z.B. als Teil eines interdisziplinären Teams), die Anforderungen stellen und mitgestalten. Sie haben das notwendige Wissen, um eigenständig Anforderungen formulieren zu können, und sind dafür nicht von Expert:innen abhängig. Der Typ geht von einer Produktionsgemeinschaft aus und damit von keinem Interessengegensatz zwischen Softwareanwendung und Softwareentwicklung.

Beim Typ der repräsentativen Technokratie werden Anwendende vertreten: ob durch Key User:innen, IT-Beratende oder andere Anwendende, die Anforderungen aufnehmen. Die Softwaregestaltung sucht einen Ausgleich zwischen den Interessen der Technokratie und den Anwendenden, indem sie bspw. ein Budget für die Umsetzung von Anforderungen der Anwendenden zur Verfügung stellt.

Bei der kapitalistischen Technokratie geben die Anwendenden maximal Feedback zu einer von Expert:innen entwickelten Software (z.B. durch Tests oder eingegebene Daten). In diesem Fall ist das Wissen (ob explizites oder implizites) der Anwendenden für Anforderungen nicht relevant und die Software bildet einen Best-Practice-Prozess ab. Der Klassenkonflikt zwischen Interessen von Unternehmen und Anwendenden schlägt sich in unterschiedlichen und unvereinbaren Anforderungen an die Software nieder.

Tabelle 28: Typen der Partizipation von Anwendenden an der Softwaregestaltung

Nr.	Partizipationszyp	Zugang der Anwenden-de	Rolle Expert:innen	Interessen Unternehmen vs. Anwendende
1	**Direkte Partizipation**	direkt	keine	kongruent
2	**Repräsentative Technokratie**	indirekt	beratend, vermittelnd	ausgeglichen
3	**Kapitalistische Technokratie**	keinen	alleinig softwaregestaltend	Konflikt

Die analytisch gebildeten Typen sollen verdeutlichen, dass keine der Fallstudien einer reinen, direkten Partizipation entspricht. Es gibt immer noch Vermittelnde zwischen Anwendenden und Programmierenden, der Betriebsrat interveniert gegen eine individuelle Leistungskontrolle und bestimmte Gestaltungswünsche des Managements haben Priorität. Ansonsten sind die Fallstudien Mischformen zwischen Typ 2 und Typ 3. Bei Fallstudien wie KOOP1 und KOOP2 hängt es von den einzelnen EVU ab, inwieweit sie überhaupt noch mitgestalten. INTERN1 und INTERN2 suchen zwar einen Ausgleich zwischen den Anwendenden- und Unternehmensinteressen, einzelne Anwendende können direkt Anforderungen stellen und Befragte berichten von einer Zufriedenheit mit der Softwarelösung. Das Unternehmensinteresse übertrumpft aber letztendlich die Interessen der Anwendenden. Beim STARTUP sind die Anwendenden weitestgehend ausgeschlossen und Fachleute der Softwaregestaltung wie Product Owner:innen gestalten die Software. Bei PAKET und KOOP3 fließt die Expert:innenmeinung im Wesentlichen über Arbeitskreise ein und der einzelne Anwendende in einem EVU hat wenig direkte Mitsprache.

Letztendlich verteilt sich Partizipation wie z.B. bei KOOP1, wo verschiedene Ebenen entscheiden: ein strategisches Gremium des Managements für größere Projekte und grundlegende Entscheidungen und operative Gremien für einzelne Anforderungen, die aus unterschiedlichen EVU kommen.

8.5.3.4. Wie würde eine auf Softwaregestaltung ausgerichtete Prozessorganisation aussehen?

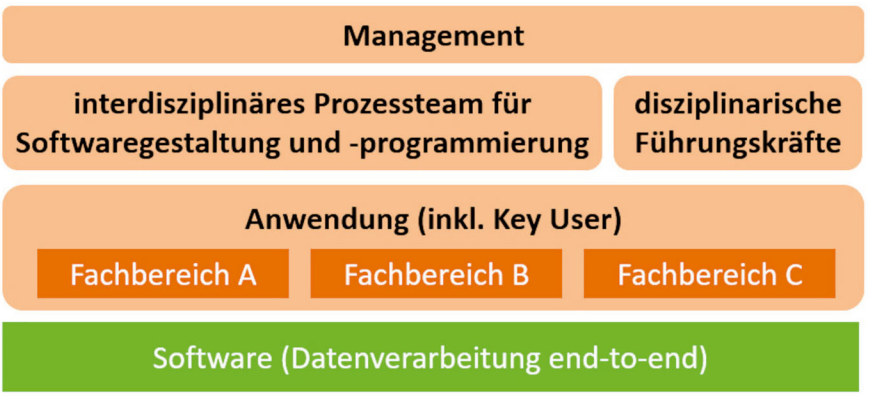
Die soziotechnische Transformation durch Softwaregestaltung stellt in einigen Fällen die bestehende Organisation in Frage. Bei INTERN2 werden die Fachbereiche (»Silos«) in Frage gestellt und ein befragter Manager will eine team- und fachbereichsübergreifende Prozessoptimierung. Dafür soll es interdisziplinäre Teams geben, bei denen Softwaregestaltende und -programmierende ohne Barrieren durch anwendende Teams oder Fachbereiche die Arbeit der Anwendenden gestalten.

Bei INTERN1 betrifft die Softwaregestaltung bereits zwei Prozessteile (Instandhaltung und Auftragsverarbeitung). Bei KOOP1 gibt es in einem EVU ein Team, das sich reorganisiert, um für Geschäfts- und Privatkundschaft zusammen die Software zu gestalten, weil im Zuge der Softwaregestaltung das EVU viele Gemeinsamkeiten zwischen

beiden festgestellt hat. Bei PAKET gibt es in einem EVU Prozessmanagende, die teamübergreifend die Software durch Einstellungen am Standard optimieren.

Zwar hat sich dieser neue Typ von Organisation in keinem der Fälle verwirklicht. Er sei hier aber kurz skizziert, um einige Kernthesen der Arbeit zu verdeutlichen: Es wäre eine soziotechnische Prozessorganisation, bei der interdisziplinäre Teams an und mit einer Software arbeiten. Programmierende und andere IT-Fachleute wären nicht mehr Teil einer IT-Abteilung, sondern Teil der Fachabteilung. Fachleute für die energiewirtschaftlichen Prozesse, für ERP-Systeme und Anwendende würden kontinuierlich eng zusammenarbeiten. Es wäre eine technikentwicklungsinduzierte Reorganisation, weil sich das EVU so aufstellen würde, dass es die Möglichkeiten der Softwaregestaltung optimal nutzen kann. Kern der Prozessorganisation wäre ein End-to-end-Prozess (bspw. vom Auslesen der Zählerdaten bis zum Erstellen der Rechnung), der in einer Software abgebildet ist. Statt mehrerer Teamleitender für einzelne Prozessabschnitte gibt es ein Prozessteam, das eine funktionsübergreifende Perspektive auf den ganzen Prozess einnimmt und die entsprechende Macht hat, Änderungen in jedem fachlichen Anwendungsbereich entlang des Prozesses umzusetzen. Der Ablauf wäre an Scrum ausgerichtet und mit einem Anforderungsmanagement verbunden, deren Teilnehmende in die Teams hinein vernetzt sind. Er würde zwischen Anwendung und Programmierung für Feedbackschleifen inkl. Tests und Prototyping sorgen. Eine direkte Kommunikation zwischen den Beteiligten wäre möglich, z.B. durch regelmäßige Sprechstunden oder Chats. Die Beziehungen wären partnerschaftlich, verbindlich, kooperativ. Alle Beschäftigten zeigen individuelle Verantwortungsbereitschaft gemäß ihrer Rolle. Die Anwendenden wären aufgeteilt in Key User:innen und reine Anwendende. Erstere machen aktiver an der Softwaregestaltung mit und unterstützen Letztere, bei denen z.B. die Abarbeitung nicht automatisierter Fälle im Zentrum steht. Es gibt disziplinarische Führungskräfte, die sich vom Prozessteam unterscheiden und die Rolle von Coaches einnehmen. Das Prozessmanagement überwacht die Kennzahlen.

Abbildung 12: Schema einer auf Softwaregestaltung ausgerichteten softwaretechnischen Prozessorganisation



So wäre die anwendende Organisation auf die Softwaregestaltung ausgerichtet. Dabei prüft das Prozessteam kontinuierlich, was sie individuell gestalten oder wo sie eine Standardlösung einsetzen und ob sie Technologien wie maschinelles Lernen nutzen. Dazu gehört, dass sich sowohl das Prozessteam als auch die Anwendung entsprechend den ausgeschöpften Möglichkeiten reorganisieren (z.B. aufgrund von fortschreitender Automatisierung). Weil Softwaregestaltung nicht immer im gleichen Umfang nötig ist, wären manche Gestaltende und Programmierende nur phasenweise dabei und z.B. als Externe engagiert oder noch in anderen IT-Projekten oder Prozessteams tätig.

Der Organisationstyp soll noch einmal verdeutlichen, dass nicht nur die Software, sondern auch die Softwaregestaltung zentral für die anwendende Organisation ist und dass die EVU in den Fallstudien im Wesentlichen noch so organisiert sind, wie es aus rein energiewirtschaftlicher Sicht naheliegt. Eine Reorganisation würde vor allem Führungsebenen wie Teamleitende betreffen. Sie würde klar zwischen Führungskräften unterscheiden, die nur disziplinarisch verantwortlich sind, und solchen, die die soziotechnische Arbeitsgestaltung durch Softwaregestaltung übernehmen. Wie sehr sich dabei die Interessen des Managements jenseits der Nutzung der Technologie für die betrieblichen Zwecke durchsetzen, würde von betrieblichen Bedingungen abhängen, bspw. ob ein Betriebsrat vorhanden ist oder nicht.

8.6. Synthese, Zusammenfassung und Diskussion des Fallvergleichs

Die sieben Fallstudien stellen Formen und Folgen der Softwaregestaltung in der Energiewirtschaft dar und werfen einen neuen Blick auf die Arbeit und die Organisationen der Branche. Sie zeigen die Softwaregestaltung als eigenständigen Arbeitsprozess, der in verschiedenen soziotechnischen Konstellationen stattfindet und der es den Organisationen erlaubt, die Möglichkeiten der Softwareentwicklung (Standard oder individuell) mit den Bedarfen der Energiewirtschaft zusammenzuführen. Sie zeigen, dass sich mithilfe des Analyserahmens die soziotechnische Konstellation, der Arbeitsprozess der Softwaregestaltung, die Folgen für die Arbeit der Softwaregestaltenden und die soziotechnische Arbeitsgestaltung der Softwareanwendung für die Fallstudien darstellen, vergleichen und Zusammenhänge zwischen ihnen untersuchen lassen. Damit haben die Fallstudien gezeigt, dass neben Methoden wie Scrum oder IT-Projekten auch noch andere Dinge zu einer adäquaten konzeptionellen Beschreibung von Softwaregestaltung und ihren Folgen gehören, wie z.B. organisationsübergreifende Abstimmungen, Karrierewege für Softwaregestaltende und die Reorganisation von Abteilungen oder zumindest die Ergänzung der anwendenden Organisation um Rollen und Abläufe für die Softwaregestaltung. Software erhält als wesentliche Entität einer Organisation einen zentralen Stellenwert dabei, Handeln in Strukturen zu erklären und zu verstehen. Zudem zeigen die Fallstudien die Situativität des Arbeitsprozesses: Rollen, Abläufe, digitale Werkzeuge und kommunikative Beziehung zeichnen Anpassung, Reflexivität, Phasenverläufe, Verhandlungen und Lernprozesse aus.

Aus der Empirie konnte das Kapitel allgemeine Unterschiede für den Arbeitsprozess der Softwaregestaltung (zentral oder dezentral), die Arbeit der Softwaregestaltenden (in einer Matrix- oder reinen Netzwerkorganisation) und die soziotechnische Ar-

beitsgestaltung der Anwendung (ob unabhängig oder abhängig) herausarbeiten. Wobei die letztere Unterscheidung zwischen unabhängiger und abhängiger Arbeitsgestaltung sich nicht auf die zwei grundsätzlich unterschiedlichen organisatorischen Ausrichtungen von In- und Outsourcing bzw. Make-or-Buy reduzieren lässt. Sie ist immer verbunden mit einem softwaretechnischen Zuschnitt (individuell/Standard) (siehe weiter unten, 8.6.1.3). Die Fallstudien zeigen die genannten drei idealtypischen Unterschiede nicht immer in Reinform. Um die Fälle zu sortieren und die Ergebnisse zu systematisieren, stellt das Empirie-Kapitel Idealtypen vor, welche die Unterschiede zuspitzen: jeweils vier für die soziotechnische Netzwerkarbeit und die soziotechnische Arbeitsgestaltung der Softwareanwendung. Sie lassen sich in 4-Felder-Matrizen einsortieren (siehe Abbildung 15 und Abbildung 16 unten), wodurch sie auf einen Blick veranschaulichen, welche unterschiedlichen Formen und Folgen der Softwaregestaltung es gibt.

Zudem fasst der abschließende Abschnitt des Kapitels die Ergebnisse der Fallstudien für die einzelnen Kategorien des Analyserahmens zusammen und gleicht sie mit der Literatur und den Konzepten aus dem 6. Kapitel ab. Er stellt ausführlich den technikentwicklungsbezogenen Rationalisierungstyp vor, der noch einmal untermauert, dass Softwaregestaltung Anwendungsbereiche rationalisiert und für das Verständnis der Rationalisierung der Branche wichtig ist. Abschließend diskutieren die letzten beiden Punkte 8.6.4 und 8.6.5 zwei Thesen: inwiefern die Softwaregestaltenden eine Konkurrenz zum Management in den EVU darstellen und inwiefern es eine industriespezifische Softwaregestaltung gibt.

8.6.1. Synthese: Typen soziotechnischer Netzwerkarbeit und soziotechnischer Arbeitsgestaltung

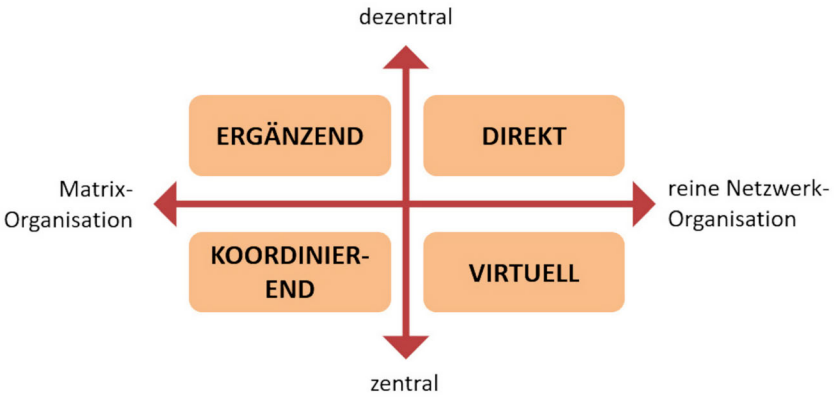
8.6.1.1. Die vier Grundtypen der soziotechnischen Netzwerkarbeit

Bei der Frage, wie Organisationen die Arbeit der Softwaregestaltenden kontrollieren, spielen sowohl der Arbeitsprozess als auch die Arbeitsbedingungen der Softwaregestaltenden eine Rolle. Aus beiden zusammen lässt sich eine 4-Felder-Matrix bilden, in die sich die unterschiedlichen Fallstudien einsortieren lassen: zwischen Matrix- und reiner Netzwerkorganisation für die Arbeit der Softwaregestaltenden einerseits und zwischen dezentralem und zentralem Arbeitsprozess der Softwaregestaltung andererseits. Wie die untenstehende Abbildung 15 darstellt, entsprechen den vier möglichen Kombinationen dieser Unterschiede vier Idealtypen: DIREKT, ERGÄNZEND, KOORDINIEREND und VIRTUELL. Die Eigenschaften der Idealtypen ergeben sich entsprechend aus den idealtypischen Unterscheidungen der Fallstudien, wie sie zu Beginn der jeweiligen Abschnitte zu den Teilen des Analyserahmens aufgeführt sind: unter 8.3.1 für den Arbeitsprozess und 8.4.1 für die Arbeit der Softwaregestaltenden.

Im ersten Quadranten der 4-Felder-Matrix befinden sich dezentrale Arbeitsprozesse der Softwaregestaltung, bei denen die Softwaregestaltenden in einer reinen Netzwerkorganisation arbeiten. Der Idealtyp der soziotechnischen Netzwerkarbeit für diesen Quadranten ist **DIREKT**. Bei DIREKT arbeiten die Beteiligten kontinuierlich interdisziplinär zusammen, und zwar dezentral in Abteilungen, Teams oder, wie im Fall von STARTUP, in Form von Kreisen. Die direkte Anforderungsaufnahme, -ausarbeitung und

-übergabe an die Programmierenden ist weder durch Markt oder Hierarchie getrennt, wie es z.B. für ein Scrum-Team typisch ist.

Abbildung 13: Matrix Arbeitsprozess (dezentral – zentral) und Arbeitsbedingungen der Softwaregestaltenden (Matrixorganisation – reine Netzwerkorganisation) und die vier Idealtypen der soziotechnischen Netzwerkarbeit



Im zweiten Quadranten der 4-Felder-Matrix befinden sich dezentrale Arbeitsprozesse der Softwaregestaltung, bei denen die Softwaregestaltenden in einer Matrixorganisation arbeiten. Der Idealtyp für diesen Quadranten ist **ERGÄNZEND**. INTERN1 und INTERN2 sind Beispiele für diesen Typ, bei denen es vor allem darum geht, trotz Abteilungsgrenzen direkt im Anwendungsbereich Anforderungen aufzunehmen, auszuarbeiten und von den Programmierenden der IT-Abteilung umsetzen zu lassen.

Im dritten Quadranten der 4-Felder-Matrix befinden sich zentrale Arbeitsprozesse der Softwaregestaltung, bei denen die Softwaregestaltenden in einer Matrixorganisation arbeiten. Der Idealtyp für diesen Quadranten ist **KOORDINIEREND**, weil sich mehrere Organisationen oder Abteilungen über den softwaretechnischen Zuschnitt abstimmen. Wie für eine zentrale Softwaregestaltung typisch, sorgt der Arbeitsprozess für die Koordination der Zusammenarbeit mehrerer Organisationen bzw. Organisationseinheiten. Die Organisationen, Abteilungen oder Teams verhandeln über Anforderungen und lösen die dabei entstehenden Konflikte. KOOP1, KOOP2 und KOOP2 sind nur zum Teil Beispiele dafür, weil in diesen Fallstudien neben dem zentralisierten Arbeitsprozess vor allem bei KOOP2 noch dezentrale Arbeitsprozesse innerhalb der EVU existieren. Allen ist jedoch gemein, dass die Softwaregestaltenden Organisations- und Abteilungsgrenzen überwinden müssen. Bei PAKET ist das der Fall, wenn die Softwarefirma in Arbeitskreisen oder in Entwicklungsprojekten mit EVU zusammenarbeitet. Allerdings existieren innerhalb der Softwarefirma sowohl interdisziplinäre Teams für einzelne Module der Standardlösung, die einer soziotechnischen Netzwerkarbeit vom Typ DIREKT nahekommen, als auch mehrere Teams betreffende interne Projekte für größere Anforderungen, die eher dem Typ ERGÄNZEND entsprechen.

Im vierten Quadranten der 4-Felder-Matrix befinden sich zentrale Arbeitsprozesse der Softwaregestaltung, bei denen die Softwaregestaltenden in einer reinen Netzwerkorganisation arbeiten. Der Idealtyp für diesen Quadranten ist **VIRTUELL.** Dabei gestalten Akteur:innen zentralisiert Software, z.B. auf einer frei zugänglichen Online-(Open-Source-)Entwicklungsplattform. Es handelt sich um ein reines Netzwerk, da die einzige Hürde zur Teilnahme darin bestehen kann, ein Benutzer:innenkonto für die Plattform zu haben. Wer eins hat, kann den Quellcode einsehen und sehen, wer gerade an welchem Teil der Software arbeitet. Der Austausch über branchenspezifische Anforderungen und softwaretechnische Möglichkeiten sowie die Lösung von Konflikten erfolgt digital. Die Beteiligten wählen selbst, wo sie mitarbeiten bzw. welchen Beitrag sie leisten. Keine der Fallstudien entspricht diesem Typ.

8.6.1.2. Zwei organisationale Kernstrategien der Arbeitsgestaltung in den EVU: Ausrichtung auf Softwareanwendung vs. auf Softwareentwicklung

Auch für die soziotechnische Arbeitsgestaltung durch die industriespezifische Softwaregestaltung lassen sich vier Grundtypen und eine 4-Felder-Matrix bilden. Was die soziotechnische Arbeitsgestaltung durch die Softwaregestaltung anbelangt, haben sich in der Empirie die Fallstudien dahingehend unterschieden, ob bei ihnen eine unabhängige oder abhängige Arbeitsgestaltung vorliegt. Dahinter verbirgt sich aber ein Verhältnis zwischen Softwaregestaltung und Softwareanwendung, das sowohl die Organisation als auch die Software der Softwareanwendung betrifft.

Doch zeigt bereits eine rein eindimensionale, rein organisatorische Sicht auf die Unterschiede zwischen den Fallstudien einige gravierende Folgen für die Arbeitsgestaltung. Dazu dient die Gegenüberstellung von Make-or-Buy bzw. ob sich ein EVU auf die Softwareanwendung oder die Softwareentwicklung ausrichtet. Nachteile der Fremdentwicklung bzw. organisatorische Abhängigkeiten treten hier deutlich hervor: Es bestehen Interessengegensätze und Machtungleichgewichte. Eine anwendende Organisation wie ein EVU kann nur die Softwareanwendung kontrollieren und entsprechend ist auch nur für die Softwareanwendung eine interne Rationalisierung möglich. Intern ist kein eigenes Wissen über die softwaretechnischen Möglichkeiten vorhanden. Im Wettbewerb muss die anwendende Organisation primär auf das Mittel der Spezialisierung setzen. Diese Nachteile existieren in den Fallstudien, in denen die EVU nicht selbst gestalten (vor allem bei PAKET).

Die Wettbewerbsstrategie des *Competing on Complexity* dank selbst entwickelter Software zeichnet laut Bessen (2022) die Marktführer verschiedenster Branchen aus (wie bereits unter 4.1 ausgeführt). Sie ist nur zu verwirklichen, wenn sich eine Organisation auf die Softwareentwicklung ausrichtet. Die Fallstudie STARTUP zeigt noch am ehesten diese Strategie, wobei die Organisation eine Nische bedient und, was die Umsätze anbelangt, mit EVU nicht konkurrieren kann. INTERN1 und INTERN2 zeigen, dass die Typen der soziotechnischen Arbeitsgestaltung auch für einzelne Teile einer Organisation gelten können. Das heißt, einzelne Fachbereiche können selbst anfangen, Software zu entwickeln.

Tabelle 29: Unterschiede zwischen integrierter und desintegrierter Softwareentwicklung aus EVU-Sicht

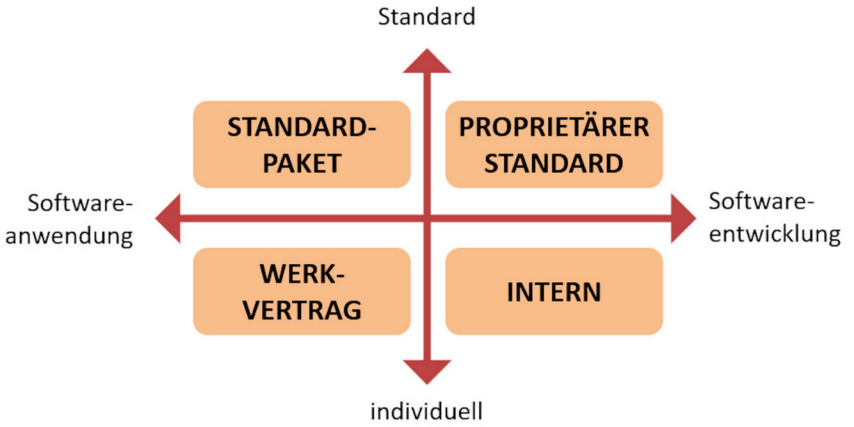
Dimension aus EVU-Sicht	**Ausrichtung auf Softwareanwendung**	**Ausrichtung auf Softwareentwicklung**
Arbeitsteilung Softwareanwendung und -gestaltung	desintegriert	integriert
Interessen	Gegensätze, Abhängigkeiten	Kongruenz, Möglichkeiten ausschöpfen
Kontrollverhältnis	Kontrolle Softwareanwendung durch (fremdgefertigte) Software	Kontrolle Softwareanwendung durch -gestaltung möglich: direkte Gestaltung Anwendungsbereich
Professionalisierung und Rationalisierung	Softwareanwendung, -gestaltung, -programmierung getrennt	gleichzeitig: Softwareanwendung, -gestaltung, -programmierung
Dynamik (branchen)fachliche und softwaretechnische Veränderungen	getrennt voneinander	integriert, Wechselspiel
Zentrale Wettbewerbsstrategie	Competing on Core Competency (Anwendung oder Entwicklung)	Competing on Complexity (Anwendung und Entwicklung)
Mögliche Kommodifizierung der Software durch	Softwarefirma	Anwenderfirma (deren Geschäftsfeld primär nicht der Verkauf von Software ist)

Die zwei Strategietypen berücksichtigen nur die Unterscheidung auf organisationaler Ebene zwischen der Ausrichtung auf Softwareanwendung oder Softwareentwicklung. Es fehlt die Unterscheidung zwischen Individual- und Standardsoftwaregestaltung. Dies geschieht in den vier Grundtypen der soziotechnischen Arbeitsgestaltung der Softwareanwendung, die über die klassische Unterscheidung von Make-or-Buy hinausgehen und soziotechnisch definiert sind.

8.6.1.3. Die vier Grundtypen der soziotechnischen Arbeitsgestaltung durch Softwaregestaltung

Die Unterschiede im Verhältnis der Arbeitsprozesse von Softwaregestaltung und Softwareanwendung treten besonders deutlich hervor, wenn sie auf vier Typen reduziert werden. Ausgehend von den softwaretechnischen Gestaltungsmöglichkeiten und den Fallstudien, lassen sich vier Idealtypen der soziotechnischen Arbeitsgestaltung bilden. Sie lassen sich auf den Achsen des soziotechnischen Zuschnitts zwischen individuell und Standard und der organisatorischen Ausrichtung zwischen Softwareanwendung und -entwicklung verorten.

Abbildung 14: Matrix softwaretechnischer Zuschnitt (Standard – individuell) und organisatorische Ausrichtung (Anwendung – Entwicklung) und die vier Idealtypen der soziotechnischen Arbeitsgestaltung (als Verhältnis von Softwaregestaltung zu -anwendung)



Beim Idealtyp **STANDARDPAKET** gestaltet wie in der Fallstudie PAKET eine Softwarefirma einen industriespezifischen Standard. Die Softwarefirma schlägt Profit aus der Skalierung des Standardprodukts. Dafür braucht sie (Branchen-)Fachwissen für den Standard und weniger Wissen aus individuellen Arbeitskontexten. Die anwendenden EVU richten sich auf diesen Standard aus. Sie spezialisieren sich auf die Softwareanwendung und zielen nicht darauf ab, sich durch eine individuelle Softwaregestaltung von Wettbewerbern zu differenzieren. Im Arbeitsprozess der Softwaregestaltung entscheiden die Beteiligten für jede Anforderung, ob sie Teil des Standards wird oder nicht. Zumindest was den kooperativ gestalteten Standard anbelangt, gehören KOOP1 und KOOP2 zu diesem Typ. Ebenso gehört KOOP3 dazu, da die IoT-Softwarefirma Standardmodule entwickelt.

Beim Typ **WERKVERTRAG** entwickelt eine Softwarefirma oder ein IT-Dienstleistungsunternehmen (IT-DL) eine industriespezifische Software im Auftrag eines EVU. Die anwendenden EVU richten sich auf diese für sie individuell gestaltete Software aus. Es besteht eine beidseitige, hohe Abhängigkeit. Dies liegt zum einen am fehlenden Wissen über Softwareentwicklung im EVU, was u.a. zu Schwierigkeiten dabei führt, dem finanziellen Aufwand der Umsetzungen zu schätzen oder etwas gegen eine zeitlich verzögerte oder mangelhafte Entwicklung zu tun. Zum anderen liegt es daran, weil der Softwarefirma oder dem programmierenden IT-DL das Wissen über den industriespezifischen Anwendungsbereich fehlt. Das ist vor allem dann problematisch, wenn stetig Änderungen notwendig sind und es sich um eine komplexe, branchenspezifische Wissensdomäne handelt. Keine Fallstudie entspricht diesem Typ. Die EVU wirken immer noch mit an der Softwaregestaltung, wenn auch nur durch Konzepte oder Tests. Bei KOOP1 und KOOP2 gibt es EVU, die das IT-DL mit einzelnen individuellen, kleineren Programmierungen beauftragen, wobei das IT-DL längerfristig mit den EVU zusammenarbeitet und über EVU- und Branchenwissen verfügt, weswegen die Abhängigkeiten geringer sind.

Beim Typ **INTERN** entwickeln EVU wie in den Fallstudien INTERN1 und INTERN2 eine individuelle industriespezifische Software. Theoretisch könnten sich die EVU so organisieren, dass es für den Arbeitsprozess der Softwaregestaltung nützlich ist, und z.B. Abteilungsgrenzen auflösen. Jedoch findet in den beiden Fallstudien keine umfassende Reorganisation für die Softwaregestaltung statt. Die EVU gestalten Software von althergebrachten Strukturen bzw. ihrem Status quo aus. Zudem erweitern und adaptieren die EVU ein Standard-ERP-System und die meisten Programmierenden sind Externe. Allerdings zeigt sich ein Lernprozess, der sich in den EVU durch unterschiedliche Grade an organisatorischen Ergänzungen auszeichnet: von einzelnen neuen Rollen und Methoden wie IT-Projekte oder Scrum bis hin zu umfassenden Arbeitsprozessen der Softwareentwicklung, die jenen einer darauf spezialisierten Softwarefirma ähneln (wenn auch z.B. über Abteilungsgrenzen hinweg organisiert).

Beim Typ **PROPRIETÄRER STANDARD** schöpft eine Organisation die Möglichkeiten der Softwaregestaltung maximal aus. Sie gestaltet individuell eine eigene Software und verkauft sie zudem auch noch anderen Unternehmen. Die Organisation ist auf die Softwareentwicklung ausgerichtet. Ausgehend vom Primat der Softwareentwicklung besteht der Anwendungsbereich aus jener Arbeit, den Software nicht erledigen kann. Die Arbeit der Anwendenden reorganisiert die Organisation bei Bedarf beliebig und kontinuierlich im Wechselspiel mit der Weiterentwicklung der Software. Sowohl eine Differenzierung von Wettbewerbern durch individuelle Softwaregestaltung als auch eine Kommodifizierung für die Teile der Software, die anderen (auch Wettbewerbern) zur Nutzung angeboten werden, ist möglich. Zu diesem Typ gehört die Fallstudie STARTUP.

Wie für Idealtypen üblich, weichen die realen Fälle davon ab. PAKET ist zwar vom Typ STANDARDPAKET, jedoch ergänzen einzelne EVU die Softwarelösung noch um Systeme anderer Softwarefirmen und es sind umfangreichere Einstellungen an der Standardlösung möglich, die einige EVU eigenständig vornehmen. Der Fall STARTUP bietet nur einen Teil seiner Lösung zur Verwendung für andere an (Teil-Kommodifizierung). Die anderen Fallstudien passen immer einen bestehenden Standard an: Bei KOOP1 gelingt es dem IT-DL, einen eigenen Standard auszuprägen, stetig weiterzuentwickeln und zu kommodifizieren, indem die beteiligten EVU für diesen bezahlen. Manche EVU von KOOP1 gestalten zusätzlich noch selbst intern. Einige EVU, insbesondere kleinere, sind rein anwendend. Das heißt, bei KOOP1 gehören einzelne EVU auch zum Typ INTERN und andere EVU gehören nur zum Typ STANDARDPAKET. Bei KOOP2 ist dies ähnlich. Wobei sich dort einzelne EVU klar aus der Kooperation verabschiedet haben und unter den Typ INTERN fallen. KOOP3 trennt durch Modularisierung der Software den Standardteil von individuellen Entwicklungen. Einige EVU verwenden das IoT-Standardpaket, während ein befragtes EVU selbst individuelle Erweiterungsmodule entwickelt.

8.6.1.4. Zusammenhänge zwischen den einzelnen Teilen des Analyserahmens

Es zeigen sich zwei wesentliche Zusammenhänge zwischen den Teilen des Analyserahmens, wenn es um den Arbeitsprozess der Softwaregestaltung und die soziotechnische Arbeitsgestaltung der Softwareanwendung geht: (1.) Wie Organisationen die Möglichkeiten der Softwaregestaltung nutzen und den Arbeitsprozess kontrollieren können. (2.) Wie effizient Organisationen die Möglichkeiten der Softwaregestaltung zum Einsatz bringen.

Zuerst zu den typischen Zusammenhängen zwischen dem Arbeitsprozess der Softwaregestaltung und der soziotechnischen Arbeitsgestaltung der Softwareanwendung. Unter 8.3.1 ist der Idealtyp des dezentralen Arbeitsprozesses der Softwaregestaltung mit dem softwaretechnischen Zuschnitt individuell und jener des zentralen Arbeitsprozesses mit Standard definiert. Entsprechend zeigen sich die Zusammenhänge in den Fallstudien:

- In den Fallstudien INTERN1 und INTERN2 liegen eine soziotechnische Netzwerkarbeit vom Typ ERGÄNZEND und eine individuelle soziotechnische Arbeitsgestaltung der Anwendung vom Typ INTERN vor.
- In der Fallstudie STARTUP liegen eine soziotechnische Netzwerkarbeit vom Typ DIREKT und eine individuelle soziotechnische Arbeitsgestaltung der Anwendung vom Typ INTERN vor.
- In den Fallstudien KOOP1, KOOP2, KOOP3 und PAKET liegen eine soziotechnische Netzwerkarbeit vom Typ KOORDINIEREND und eine soziotechnische Arbeitsgestaltung der Anwendung vom Typ STANDARDPAKET vor.

Allerdings sind diese Zusammenhänge in den Fallstudien nicht eindeutig, weil in einigen Fallstudien mehrere Arbeitsprozesse der Softwaregestaltung existieren oder einzelne Arbeitsprozesse nicht den Idealtypen entsprechen.

- Bei INTERN2 arbeiten mehrere Fachabteilungen zusammen und stimmen sich ab. Sie müssen sich teilweise auf einen gemeinsamen Standard einigen.
- STARTUP bietet die individuell gestaltete Software anderen Firmen als Standard an, weshalb die Fallstudie noch mehr zum Typ PROPRIETÄRER STANDARD gehört als zum Typ INTERN.
- Bei KOOP1, KOOP2 und KOOP3 gibt es innerhalb der EVU Softwaregestaltung, weswegen eine soziotechnische Netzwerkarbeit des Typs ERGÄNZEND für eine soziotechnische Arbeitsgestaltung des Typs INTERN existiert. Zugleich gestalten die IT-DL für manche EVU individuelle Software, weswegen eine soziotechnische Netzwerkarbeit des Typs ERGÄNZEND in Kombination mit der soziotechnischen Arbeitsgestaltung vom Typ WERKVERTRAG existiert.
- PAKET gestaltet unabhängig von den EVU Teile des STANDARDPAKETs mit soziotechnischer Netzwerkarbeit vom Typ ERGÄNZEND oder DIREKT.

Somit besteht zwar ein enger Zusammenhang zwischen soziotechnischer Konstellation (Arbeitsteilung, Grundkonstellation und Architektur), soziotechnischer Netzwerkarbeit (Arbeitsprozess, Arbeit der Softwaregestaltenden) und soziotechnischer Arbeitsgestaltung. Abweichungen sind aber möglich und vorhanden.

Neben diesen Zusammenhängen, welche die bloße Umsetzung der Softwaregestaltung betreffen, zeigen die Fallstudien Zusammenhänge hinsichtlich der effizienten Umsetzung dieser Möglichkeiten der Softwaregestaltung.

Erstens bietet für die Softwaregestaltung die reine Netzwerkorganisation Vorteile. Dann müssen die Softwaregestaltenden keine Hierarchien oder Abteilungsgrenzen überwinden und nicht mit den Spannungen zwischen Organisationen aufgrund von

Marktbeziehungen zurechtkommen. Solche Vorteile zeigen sich in der soziotechnischen Netzwerkarbeit des Typs DIREKT wie bei STARTUP. Dort ist die gesamte Organisation auf die interdisziplinäre Zusammenarbeit für die Softwaregestaltung ausgerichtet und eingespielt. Die Nachteile zeigen sich in der Fallstudie KOOP2 im EVU2. Es muss erst noch die interne Zusammenarbeit in der Matrixorganisation lernen.

Zweitens, was die soziotechnische Arbeitsgestaltung anbelangt, nutzt eine anwendende Organisation die Möglichkeiten der Softwaregestaltung dann maximal, wenn sie selbst individuell eine Software für die eigene Anwendung gestaltet und zugleich diese Software auch noch anderen anbietet und damit Geld verdient wie beim Typ PROPRIETÄRER STANDARD. Auch diese Vorteile nutzt exemplarisch STARTUP. Es ist eine anwendende Softwarefirma, welche an zwei Wertschöpfungsketten teilnimmt: jener der Energiewirtschaft und jener der Software-Industrie.

Doch egal wie effizient die Organisationen die Möglichkeiten der Softwaregestaltung nutzen: Die soziotechnische Netzwerkarbeit der Softwaregestaltung müssen alle hinbekommen. Die folgenden Abschnitte fassen die Ergebnisse für jeden Teil und jede Kategorie des Analyserahmens zusammen und stellen Bezüge zur Forschungsliteratur aus dem 6. Kapitel her.

8.6.2. Zusammenfassung je Teil des Analyserahmens

8.6.2.1. Soziotechnische Konstellation

Die Fallstudien zeigen, dass die soziotechnische Konstellation die Ausgangsbedingungen für die Softwaregestaltung darstellt. Zugleich stellt sie auch deren Grenzen dar, in denen sich die Möglichkeiten der Softwaretechnik verwirklichen lassen: Vom Anwendungsbereich hängt der mögliche Digitalisierungsbeitrag der Softwaregestaltung ab, von der Softwarearchitektur die Arbeitsteilung zwischen Anwendung und Entwicklung, von der Arbeitsteilung wiederum die Wissensgrenzen und unabhängig davon, wie die Grundkoordination aussieht, müssen die Beteiligten von dieser ausgehend die Zusammenarbeit zwischen Anwendung und Entwicklung etablieren.

Je Fallstudie ist es vom **Anwendungsbereich** abhängig, welche Möglichkeiten für die Softwaregestaltung bestehen. Das betrifft den Anteil der Datenverarbeitung im Anwendungsbereich und ob sie einen gesamten Prozess gestalten kann oder nur Teile davon. Von beidem hängt erstens ab, was der Kern der Rationalisierungsmöglichkeit ist: z.B. entweder eine weitgehende Automatisierung der Datenverarbeitung (so dass keine Anwendenden mehr eingreifen müssen) oder die automatisierte Steuerung von Arbeitskräften z.B. mit mobilen Endgeräten. Zweitens hängt vom Anwendungsbereich ab, wie komplex die abzubildenden Prozesse und Wissensdomänen sind, z.B. ob die Software einen Prozess abbildet, der mehrere Abteilungen betrifft oder nur ein Team in einem EVU.

Aus Sicht der Softwaregestaltung wird die **Arbeitsteilung** zwischen Anwendung und Programmierung zur Wissensgrenze, die überwunden werden muss. Sie kann innerhalb oder zwischen Firmen bestehen. Es gibt Fälle, in denen Anwendung, Gestaltung und Programmierung Teil einer Organisation sind, aber sie Abteilungsgrenzen trennen (INTERN1, INTERN2). Es gibt Fälle, die interdisziplinär arbeiten und in denen die Wissensgrenzen nur noch zwischen einzelnen Beschäftigten bestehen – im STARTUP, in-

nerhalb von interdisziplinären Teams in der Softwarefirma der Fallstudien PAKET und KOOP3 oder den IT-DL von KOOP1, KOOP2 und KOOP3. Bei den zuletzt genannten fünf Fällen betrifft die engere interdisziplinäre Zusammenarbeit jedoch Gestaltung und Programmierung. Anwendende sitzen vorwiegend (außer jene der BPO-Bereiche von IT-DL und Softwarefirma) in den EVU (PAKET, KOOP1, KOOP2, KOOP3) oder sind aus den Kreisen ausgeschlossen, in denen schwerpunktmäßig die Softwaregestaltung stattfindet (STARTUP).

Von der **Grundkoordination** ausgehend etablieren die Beteiligten die Softwaregestaltung als soziotechnische Netzwerkarbeit. Als primäre Koordinationsformen zeigen sich in den Fällen zwischen Anwendung und Entwicklung Markt, Hierarchie oder Netzwerk. Egal welche Koordination vorliegt, die Beteiligten müssen immer einen Weg finden, um für die Softwaregestaltung zusammenzuarbeiten. Es reicht aus Sicht der vorliegenden Arbeit nicht aus, dass mehrere Organisationen oder Mitarbeitende einer Organisation quer zur Hierarchie und Abteilungen in einer Matrixorganisation zusammenarbeiten, um von einer Netzwerkkoordination zu sprechen. Denn zur Grundkoordination von Markt, Hierarchie oder Netzwerk gehören in der Empirie immer auch noch Kontrollelemente wie IT-Budget, Service Level Agreements (SLA), interne Projektverträge oder Quellcode-Reviews. Zum Beispiel sind die bei KOOP1 bestehenden Marktbeziehungen zwischen den Firmen durch SLA detailliert vertraglich fixiert. Bei KOOP3 hingegen bestehen zwar auch Marktbeziehungen zwischen IT-DL und Softwarefirma. Jedoch spielen Verträge eine untergeordnete Rolle, weswegen die Grundkoordination in diesem Fall unterm Strich das Netzwerk ist. So existieren in den Fallstudien gemischte Koordinationsformen, wie dies auch schon die Forschung zu IT-Projekten (6.3) oder Autoren wie Lamoreaux/Raft/Temin (2003), Bradach/Eccles (1989) oder Wiesenthal (2000) allgemein festgestellt haben. Was zeichnet nun eine Netzwerk-Grundkoordination aus? Sie zeichnet aus, dass wie bei STARTUP und KOOP3 ein Arbeitsprozess der Softwaregestaltung aus Rollen, Abläufen und interpersonalen Beziehungen existiert, der ungehindert von hierarchisch-formalen Wegen, vertraglichen Einschränkungen oder Marktkalkülen für die Zusammenarbeit zwischen Anwendung und Programmierung sorgt.

Die IT-Abteilung der softwareanwendenden Organisationen ist in den meisten Fallstudien Teil der Arbeitsteilung zwischen Anwendung und Entwicklung. Sie ist für den Betrieb der IT-Systeme zuständig, stellt die Programmierenden zur Verfügung, koordiniert die als Externe eingebundenen Programmierenden und in einigen Fällen sind die internen Softwaregestaltenden ihr zugeordnet. Auch wenn in einzelnen Fällen die Fachbereiche eine bessere Zusammenarbeit von ihr erwarten, sind in den Fallstudien die IT-Abteilungen insgesamt kooperativ, was die Softwaregestaltung anbelangt.

Anders als in manchen anderen Untersuchungen (vgl. Ortmann et al. 1990, Symon 2000, Silva 2005) zeigen die Fallstudien nur wenige Interessenkonflikte oder Machtkämpfe. Nur bei INTERN1 hätte der Fachbereich gern die Hoheit bei der Softwaregestaltung, bekommt sie jedoch von der IT-Abteilung nicht. Die Anforderungsmanagerin aus dem Fachbereich muss sich den Entscheidungen des in der IT-Abteilung angesiedelten Haupt-Product-Owners fügen. Interessenkonflikte oder Machtkämpfe existieren, wie dies auch einige Studien thematisieren (vgl. Peled 2001, Flecker/Holtgrewe 2008, Kaniadakis 2012, Mezihorak 2018), mehr zwischen IT-DL und EVU. Wobei es nur bei KOOP2 zum Bruch kommt und ein EVU gar nicht mehr mit dem IT-DL arbeitet. Bei PA-

KET thematisieren EVU zwar die Abhängigkeit von der Softwarefirma. Allerdings geht es dabei mehr um die Qualität und weniger um Kosten und es spricht keine der befragten Personen davon, die softwarezuliefernde Firma zu wechseln. Sonst zeigen die Fallstudien, dass die Organisationen auf unterschiedlichen Wegen die Kooperation zwischen IT-DL und EVU aufrechterhalten, z.B. durch regelmäßige Treffen oder Mediatoren. Bereits die Forschung zum IT-Alignment schlägt regelmäßigen Austausch zwischen IT-Abteilung und Fachbereichen, Co-Lokation, gemeinsame Planung, gemeinsame Projekte und Bildung von sozialem Kapital zur besseren Zusammenarbeit vor (vgl. Reich/Benbasat 2000, Chan/Reich 2007, Masak 2006, Schlosser et al. 2015, Valorinta 2011). Ein Befragter von EVU2 der Fallstudie KOOP2 thematisiert, dass die IT nicht die Probleme der Fachabteilungen kennen würde, und weist damit auf ein mangelndes IT-Alignment hin.

Die **Softwarearchitektur** hat sich in allen Fällen als prägend für die Softwaregestaltung erweisen: Erstens entscheidet ihre Aufteilung z.B. in Module darüber, wie sich die Arbeit innerhalb und zwischen den Organisationen verteilt. Zweitens bestimmt ihr softwaretechnischer Zuschnitt (Standard, individuell), welche Abhängigkeiten zwischen Organisationen oder Teilen einer Organisation bestehen.

Die Arbeitsteilung bestimmt die Softwarearchitektur durch ihren Aufbau vor allem für die Programmierenden. In den Softwarefirmen, den IT-DL und den EVU sind die Programmierenden spezialisiert und immer für bestimmte Teile der Software zuständig. Die dazugehörigen Fachleute mit dem branchenspezifischen Wissen sind ebenso meist spezialisiert auf einzelne Teile der Software und arbeiten enger mit den jeweiligen Programmierenden zusammen. Insofern ist einerseits Conways Law richtig (vgl. Conway 1968) und die Entwicklungsorganisation prägt die Architektur (Spiegelungshypothese). Wobei in den Fallstudien die Architektur eine Ausgangsbedingung ist, welche die Beschäftigten fortführen und nicht ändern (z.B. gibt sie SAP vor oder die IoT-Softwarefirma). Andererseits gilt die Spiegelung nicht immer. Bei INTERN1 unterbindet die Anforderungsmanagerin, dass Programmierende Insellösungen schreiben, d.h. mehrmals das Gleiche programmieren. Bei KOOP1, KOOP2 und KOOP3 tauschen sich mehrere Organisationen regelmäßig darüber aus, was sie individuell und was als Standard gestalten. Das heißt, hier findet Kommunikation über die Softwaregestaltung jenseits des für ein Modul zuständigen Teams aus Gestaltenden und Programmierenden statt. Wobei wir schon beim zweiten Punkt wären:

Die Softwarearchitektur prägt in der Mehrzahl der Fälle durch ihren Zuschnitt (individuell, Standard) und die daraus resultierenden Abhängigkeiten den Kommunikationsaufwand und welche Kommunikationswege zusätzlich zur Arbeitsteilung zwischen Anwendung und Entwicklung existieren müssen. Wenn z.B. mehrere EVU sich darüber abstimmen, wie sie einen gemeinsamen Teil der Standardsoftware gestalten, dann etablieren sie zusätzliche Strukturen wie in der Fallstudie KOOP1 ein Anforderungsmanagement. Diese Kommunikationswege spiegeln sich allerdings nicht in der Softwarearchitektur. Vielmehr gehen sie auf Abhängigkeiten zwischen unterschiedlichen Stakeholdern eines gemeinsam gestalteten Softwareteils zurück.

Letztendlich ist der Literatur zuzustimmen, dass Conways Law die Kommunikationswege nicht festlegt und umgangen werden kann, es aber trotzdem noch prägend ist (vgl. Colfer/Baldwin 2016, Hvatum/Kelly 2005). Die Kommunikationswege weichen

vor allem dann vom Aufbau der Softwarearchitektur ab, wenn Organisationen einen gemeinsamen Standard gestalten und sich deswegen abstimmen müssen oder auf mehrere Organisationen verteilte Anwendende an einer Software mitgestalten, z.B. via Chats oder Ticketsysteme. Wie auch in der Forschung schon festgestellt, können sich durch eine veränderte Architektur die Abhängigkeiten in einer Organisation verändern und Abstimmungen notwendig machen (vgl. Remer 2008). Das ist bei den Fallstudien wie KOOP1, KOOP2 und KOOP3 vor allem dann der Fall, wenn sich ein EVU entscheidet, eine Software individuell zu gestalten, und sich somit nicht mehr mit anderen abstimmen muss. Das unterstreicht die Bedeutung des softwaretechnischen Zuschnitts als ein Kernproblem der Softwaregestaltung und als eigenständige Kategorie, um den Arbeitsprozess der Softwaregestaltung zu beschreiben. In den Fallstudien verhandeln die Firmen kontinuierlich darüber, teilweise auch bezogen auf einzelne Anforderungen, ob sie diese individuell oder als Standard umsetzen.

Was Conways Law nicht berücksichtigt, aber sich für die Kommunikation als entscheidend in den Fallstudien herausgestellt hat, sind die zwischenmenschlichen Beziehungen. Ein befragter Architekt stellt einen Zusammenhang mit der Softwarearchitektur her:

> »Unabhängig der vielleicht potenziell möglichen Kommunikationsformen ist der Habitus von Abteilungsleitern oder auch ganzen Abteilungen das Wesentliche dafür, ob man Conways Law sehen kann oder nicht. Ich glaube, dass das quasi unabhängig des Mediums ist. Wenn sich zwei Leute mögen, dann siehst du das quasi in der Software, weil die gut funktioniert und gut zusammenarbeiten. Und ich habe auch schon Momente gehabt, da konnte ich in der Software den Weg um die Abteilung herum sehen.« (Software-Architekt)

In der Fallstudie KOOP2 führen die schlechten Beziehungen zwischen dem IT-DL und einem EVU dazu, dass Letzteres sich nicht mehr von ihm betreuen lässt, wodurch auch ein gemeinsamer Standard scheitert. Insofern zeigen die Fallstudien, dass es von einer gelungenen Kooperation abhängt, gemeinsam eine Standardarchitektur hinzubekommen.

Wie der im nächsten Punkt behandelte Teil des Analyserahmens zeigt, sind kommunikative Beziehungen grundsätzlich Teil des Arbeitsprozesses der Softwaregestaltung.

8.6.2.2. Soziotechnischer Arbeitsprozess der Softwaregestaltung

Die Fallstudien vertiefen das Verständnis darüber, wie die im 6. Kapitel beschriebenen vier Ebenen der soziotechnischen Netzwerkarbeit die Arbeit der Softwaregestaltung kontrollieren und dadurch Arbeitskraft transformieren. Neben der soziotechnischen Konstellation aus Arbeitsteilung, Grundkoordination und Architektur basieren die vier Ebenen auf dem Arbeitsprozess der Softwaregestaltung: 1. Ein von Rollen und deren Erwartungen kontrolliertes Handeln der Softwaregestaltenden. 2. Ein mehr oder weniger formaler Ablauf mit Gestaltungsnetzwerk, Feedbackschleifen, situativen Anpassungen und Lernprozessen. 3. Interpersonale, kommunikative Beziehungen als Grundlage für Kommunikation und Kooperation. 4. Softwarewerkzeuge und die gestaltete Software selbst, die das Handeln ermöglichen und einschränken, als gemeinsame Bezugspunkte,

die die Arbeit sämtlicher Beteiligten koordinieren und den Wissensaustausch und die Kommunikation dokumentieren. Zudem zeigen die Fallstudien eine Flexibilität der Ebenen von Rollen, Ablauf, kommunikativen Beziehungen und digitalen Werkzeugen. So kompensieren in manchen Fällen z.B. engagierte Beschäftigte fehlende formalisierte Abläufe oder ersetzt direkte Kommunikation jene rein formale über ein Ticketsystem.

Der Fallvergleich legt für die Kategorie der **Rollen** dar, dass, statt en détail die Arbeit der Softwaregestaltenden in dezentralen oder zentralisierten Arbeitsprozessen festzulegen, es darum geht, dass Arbeitende Erwartungen erfüllen. Davon existieren mehrere: Softwaregestaltende sollen bei Bedarf mehrere Rollen übernehmen, Rollen wechseln, unterschiedliche Rollen in den unterschiedlichen Arbeitsprozessen von Softwaregestaltung, -anwendung und -programmierung einnehmen. Dazu gehört, dass sich die Softwaregestaltenden in den Arbeitsprozess der Softwaregestaltung, wie er jeweils ist, einfügen, sich entsprechend anpassen, ggf. selbst herausfinden, was zu tun ist, die Rolleneinhaltung mit anderen verhandeln und sie gegenseitig kontrollieren – z.B. dadurch, dass Person A an Person B Erwartungen formuliert. Der Arbeitsprozess ist eine Sozialisationsinstanz für das Lernen von Rollen, wobei in den meisten, aber nicht in allen Schulungen Methoden wie Scrum oder IT-Projektmanagement Ausgangspunkte dafür sind. Eine weitere Erwartung ist, mit den Graubereichen der Verantwortlichkeiten und mit Erwartungs-/Rollenkonflikten umzugehen und die eigenen Spielräume zu nutzen. Das rührt daher, dass in vielen Fällen eine Mischung aus hierarchie-/markt- und rollenbasierter Organisation vorliegt. Zum einen sind Softwaregestaltende zwar oftmals Teil von Hierarchien, in denen z.B. Führungskräfte Entscheidungen treffen. Zum anderen sind die Softwaregestaltenden selbst nicht hierarchisch organisiert, weswegen sie Entscheidungen im Dialog treffen oder Erwartungen abgleichen müssen, ohne auf Hierarchien zur Durchsetzung zurückgreifen zu können. Somit erlauben Rollen es, situativ und unabhängig von Hierarchien zu agieren. In Fallstudien wie KOOP3 oder STARTUP gibt es gar keine formalen Hierarchien innerhalb des Arbeitsprozesses der Softwaregestaltung.

Für den **Ablauf** der Softwaregestaltung zeigen die Fallstudien, dass dieser zwar formalisiert ist – aber unterschiedlich stark. Er ist mit einem Netzwerk an Beziehungen für die Anforderungsgewinnung verbunden. Er ist vereinbar mit verschiedenen Handlungsorientierungen, wobei der Primat der Kooperation gilt, um über Hierarchien und Marktbeziehungen hinweg zusammenzuarbeiten. Der Ablauf stellt in den Fallstudien Feedbackschleifen zwischen Softwareentwicklung, -anwendung und -gestaltung her und sicher: ob durch Treffen, E-Mails, Tests, persönliche Gespräche oder Methoden wie Prototyping oder Resonanzgruppen. Er fügt sich in die jeweilige soziotechnische Konstellation ein. Was seine Ausgestaltung anbelangt, ist er in sämtlichen Fallstudien durch Lernprozesse geprägt und kann temporär oder langfristig existieren. Die Informalität des Ablaufs besteht mal in Ad-hoc-Absprachen via Chat, direktem, persönlichem Austausch oder dass nicht formal festgelegt ist, wie Entscheidungen gefällt werden. Seien es Entscheidungen über den Einsatz von Methoden, wer an einem Termin teilnimmt, die Priorisierungen und Aufwandsschätzung von Tickets, wie viele Anforderungen die Programmierenden in einem bestimmten Zeitraum erledigen können, oder die konkreten Verantwortlichkeiten einer Rolle. Die in der Forschung untersuchten Lernprozesse bei der Steuerung von organisationsübergreifender Zusammenarbeit (vgl. Mola et al. 2017, van Fenema/Keers/Zijm 2014) zeigen sich vor allem bei KOOP1. Bei KOOP2 ist die Koope-

ration daran gescheitert, dass die Beteiligten keine Mittel fanden, um Konflikte zu lösen und Erwartungen abzugleichen.

Die Kategorie der **kommunikativen Beziehungen** zeigt, dass ein bestimmtes Maß an Beziehungsfähigkeit Teil des Arbeitsprozesses der Softwaregestaltung ist und Rollen, Abläufe oder digitale Werkzeuge nicht ausreichen. Damit ist Kommunikation kein abstrakter, rein kognitiver Informationsaustausch. Sie gründet auf zwischenmenschlichen Beziehungen, die Befragte mit Begriffen wie »partnerschaftlich«, »kooperativ«, »auf Augenhöhe«, »Familie«, »Offenheit« oder »Geben und Nehmen« beschreiben. Dazu gehören langfristige Beziehungen und direkte Kommunikation zwischen zwei Personen verschiedener Abteilungen oder Organisationen trotz Hierarchien oder Marktbeziehungen. Dazu gehört eine gemeinsame Sprache, um die interdisziplinären Wissensgrenzen zwischen IT- und Energiewirtschaft zu überwinden. Aber auch gemeinsame Begriffe, wenn mehrere Teams innerhalb eines EVU oder mehrere EVU zusammenarbeiten, gehören dazu, ebenso wie die Fähigkeit, sich verständlich auszudrücken. Wenn Beziehungen so eine wichtige Rolle dafür spielen, dass miteinander zum Zwecke der Softwaregestaltung geredet wird, ist es naheliegend, dass in einem Fall ein professioneller Mediator unterstützt (KOOP1). Erstaunlich ist, dass nur in einem Fall eine Befragte eine Schulung zur Förderung der Kommunikation erwähnt (EVU2, KOOP2[23]). Damit sind die Ergebnisse zum einen anschlussfähig an die Forschung darüber, dass Projektstrukturen allein für die kooperative Zusammenarbeit nicht ausreichen und Beziehungsfaktoren wie Kooperationsbereitschaft, Sozialkompetenz (vgl. Rüegg-Stürm/Young 2001) oder Vertrauen (vgl. Powell 1990) notwendig sind und informelle Strukturen der Kooperationen die Organisationen gezielt durch verschiedene Methoden erst etablieren müssen (vgl. Bolte/Porschen 2007).

Was die **digitalen Werkzeuge** anbelangt, sind in sämtlichen Fällen Ticketsysteme und zudem oftmals Entwicklungs- und Testumgebung Teil der betrieblichen Realität. Manchmal werden sie kombiniert mit Chat-Gruppen, Excel-Dateien oder Microsoft Sharepoint und natürlich: E-Mails. Je nach Fall fungieren sie neben der Koordination zur Kontrolle der Arbeit durch Kundschaft oder Führungskraft, weil sie für diese transparent machen, welche Anforderungen es gibt, wie lang ihre Umsetzung dauert und welchen Status sie haben. Aber der Zeitrhythmus wird nicht durch einen digital-maschinellen Takt bestimmt, sondern durch Erwartungen von Kundschaft und Führungskraft, die Aufwandsschätzung durch einzelne Mitarbeitende wie Programmierende, IT-Beratende oder Product Owner, durch die Ad-hoc-Umsetzung einer dringenden Anforderung, Umsetzungsfristen durch die Regulierung, die Dauer eines Sprints in Scrum oder durch in Verträgen (SLA) festgelegte Reaktionszeiten für Tickets. Durch die Dokumentation der Softwaregestaltung im Ticketsystem oder anderen Softwarelösungen und dem Quellcode selbst entsteht Transparenz über sie. Auch wenn keine befragte Person von einer individuellen Leistungskontrolle berichtet, sind die Voraussetzungen dafür gegeben, dass der Panoptikum-Effekt (bzw. »electric panopticum« nach Zuboff 1988) wirkt. Wie im 6. Kapitel über die Rolle von Software bei der Arbeitskontrolle ausgeführt (6.4.2.1), zeigen die Fallstudien, dass nicht nur eine Software eine bestimmte Funktion

23 Die Befragte hat die Working-Out-Loud-Methode (WOL) genannt.

erfüllt: Vielmehr verwenden die Beteiligten im Arbeitsprozess mehrere Softwarelösungen gleichzeitig und permanent und die Software wirkt sowohl ermöglichend als auch einschränkend, wobei die Besonderheit in der Wissensarbeit die Objektkontrolle durch Software (vgl. Rennstam 2012) ist.

Über den **softwaretechnischen Zuschnitt** als Teil der softwaretechnischen Gestaltungsmöglichkeiten (siehe 3.2) entscheidet der Arbeitsprozess der Softwaregestaltung. Er ist ein kritischer Punkt im Arbeitsprozess, wenn es um Fragen von Konflikten, Effizienz und Partizipation geht. Es zeigen sich in den Fallstudien Unterschiede, wo und wer über den Zuschnitt entscheidet, wie dadurch Synergien gehoben und wie Prioritäten gesetzt werden. Es können sich verschiedene Gruppen bei der Entscheidung gegenüberstehen und verhandeln müssen. Die Entscheidungsprozesse können institutionalisiert sein oder an einzelnen Rollen hängen, die bspw. Synergien bei der Programmierung erkennen (oder eben nicht). An der Fallstudie KOOP1 zeigt sich, dass mehrere EVU dank des gemeinsamen, ausgefeilten Anforderungsmanagements über mehr als 10 Jahre hinweg kontinuierlich darüber verhandeln, wo sie Synergien sehen und was sie individuell machen wollen. Das heißt, mehrere EVU sind fähig, organisationsübergreifend Synergien zu erkennen. Bei INTERN2 gibt es zum Erkennen von Synergien die Anforderungsrunde mehrere Fachbereiche. Im EVU3 von KOOP1 ist es Aufgabe des Prozessmanagers, dies für die beiden getrennten Anwendungsbereiche von Privat- und Geschäftskundschaft in seiner Abteilung zu tun. Die Anforderungsmanagerin von INTERN1 versucht zu verhindern, dass die Programmierenden Insellösungen schreiben.

Die **Flexibilität der Kommunikation** in der Softwaregestaltung (6.4.4) zeigt sich auch in den Fallstudien. Es gibt ganz unterschiedliche Faktoren, die beeinflussen, wie die Kommunikation in den Fallstudien abläuft. Bei KOOP3 ist die IoT-Software modular, weswegen EVU1 und IT-DL eigenständig Erweiterungen programmieren können, was die Kommunikation mit der Softwarefirma reduziert. Bei INTERN1 kann der Fachbereich die Anforderungen mittlerweile so formulieren, dass die Programmierenden verstehen, was gemeint ist, wodurch weniger direkte Kommunikation notwendig ist. Bei INTERN2 gibt es Treffen vor der Anforderungsrunde, um Anforderungen so auszuformulieren, dass der Austausch nicht nur via Ticketsystem stattfindet. Eine frühzeitige, direkte Kommunikation soll ein späteres Hin und Her via Ticketsystem vermeiden. Bei KOOP1 ist das Anforderungsmanagement formalisiert, eingespielt und es gibt spezialisierte Rollen dafür. Bei KOOP2 im EVU2 muss der Projektmanager erst noch die Abläufe etablieren und die Beteiligten des Projekts eine gemeinsame Wissensbasis aufbauen, um effektiv Anforderungen erarbeiten zu können. Damit zeigen sich auch die von Srikanth/Puranam (2014) genannten drei Elemente aus Common Ground, modularer Softwarearchitektur und direkter Kommunikation, die je nach Kontext unterschiedlich die Kommunikation bestimmen.

Wie in der Forschung bereits beschrieben, kommen vielfältige Kommunikationskanäle zum Einsatz (vgl. Heidenreich/Kirch/Mattes 2008) und die Beschäftigten kombinieren direkte Kommunikation mit digitaler (Chats, Ticketsysteme, geteilte Dokumente). Wobei alle Organisationen verbale Kommunikation, die für die Programmierung relevant wird, verschriftlichen – sei es im Ticketsystem, E-Mails oder anderen Softwarelösungen zur Dokumentation. Auffällig ist, dass viele Befragte die direkte Kommunikation in der Softwaregestaltung bevorzugen.

8.6.2.3. Folgen für die Arbeit der Softwaregestaltenden

Die Fallstudien zeigen, dass hinter der Softwaregestaltung eine eigenständige Gruppe von Beschäftigten steht, was Beschäftigungssystem, Kontrolle und Wissen anbelangt. So verändert der Arbeitsprozess der Softwaregestaltung die EVU und die Energiewirtschaft allein schon dadurch, dass ausgehend von ihm drei Gruppen zu unterscheiden sind: Anwendende, Gestaltende und Programmierende. Auch wenn die Arbeitsprozesse der Softwareanwendung und -programmierung nicht Gegenstand der vorliegenden Untersuchung sind, dienen die Aussagen aus den Interviews zur Arbeit der Anwendenden und Programmierenden dazu, den Blick für die Eigenheiten der Softwaregestaltung zu schärfen und zu prüfen, inwiefern sie sich voneinander unterscheiden. Doch zunächst etwas über die Arbeit der Softwaregestaltenden:

In den Fallstudien fällt auf, dass die Führungskräfte die Softwaregestaltenden vor allem anhand ihrer Ergebnisse und nur bei Auffälligkeiten direkt **kontrollieren**. Führungskräfte kümmern sich um den Rahmen: Sie legen je nach Fall das Budget fest, stellen Mitarbeitende ein, initiieren Projekte oder klären Konflikte innerhalb (z.B. mit anderen Führungskräften oder Teams) oder zu anderen Organisationen (Softwarefirma, IT-DL, EVU). Softwaregestaltende relativieren öfters ihre Arbeitsbelastung (z.B. weniger schlimm als in der Beratung) und stellen sie als verhandelbar dar. Sie sprechen öfters davon, dass sie intrinsisch motiviert sind. Sie sind hierarchisch höhergestellt als Anwendende. Dabei prägt die Grundkoordination die Kontrolle ihrer Arbeit: Von ihr hängt ab, ob mehr das Management (Hierarchie), die Kundschaft (Markt) oder die Kollegenschaft (Netzwerk) kontrolliert. Sie sind einer *qualitativen Intensivierung* ausgesetzt, weil sie in vielen Fällen IT-, energiewirtschaftliches und Methodenwissen kombinieren oder zumindest mit den jeweiligen Expert:innen kommunizieren können müssen, öfters gleichzeitig oder wechselnd in unterschiedlichen Projekten und an verschiedenen Softwarelösungen arbeiten. Die *quantitative Intensivierung* ist bei den Gestaltenden nicht eindeutig. Die einen sprechen davon, dass sie phasenweise mal mehr und mal weniger zu tun haben, die anderen davon, dass sie selbst Grenzen setzen müssen und können, wieder andere davon, dass sie geregelte Arbeitszeiten haben.

In einigen Fallstudien sind die Softwaregestaltenden in puncto **Beschäftigungssystem** Teil eines IT-DL oder einer Softwarefirma und arbeiten mit mehreren EVU zusammen. Dadurch aber auf eine höhere Arbeitsbelastung zu schließen, lassen die Aussagen der Befragten nicht zu. Denn diese stellt sich für einzelne Befragte egal welcher Gruppe als sehr unterschiedlich dar und ist nicht einfach auf einen Faktor zurückzuführen, wie z.B. mit mehreren EVU zusammenzuarbeiten oder in einem kleinen EVU angestellt zu sein. Sonst zeichnen sich die Gestaltenden durch ein geringeres Interesse an einer disziplinarischen Karriere aus, mehr (Arbeits-)Marktmacht und eigenen Arbeitsmärkten. Die Organisationen setzen sie flexibel ein – ob innerhalb einer Matrixorganisation oder in anderen Organisationen. Vor allem tun die Organisationen dass, wenn sie spezifisches Methodenwissen haben (z.B. Scrum) oder tiefere Kenntnisse einer auch in anderen Branchen genutzten Softwareumgebung (z.B. SAP). Das zeigt sich bei KOOP1 daran, dass die Softwaregestaltenden das IT-DL als Durchgangsstation in ihrer Karriere sehen. Wenn EVU anfangen, Software zu gestalten, gibt es teilweise neue Karrierewege für Softwaregestaltende (»Kompetenzkarriere«). Wobei die herkömmlichen Karrieremuster als Aufstieg auf einer Leiter in einer Hierarchie mit disziplinarischer Verantwortung weiter-

bestehen. Das IT-DL von KOOP2 hat schon die Kompetenzkarriere eingeführt, was dem EVU2 aus der gleichen Fallstudie schwerer fällt. Schafft ein EVU es nicht, entsprechende Karriereperspektiven für Softwaregestaltende zu schaffen, verlassen diese wie im Falle des EVU2 von KOOP2 die Organisation. Wenn ein IT-DL die Softwaregestaltung für mehrere EVU organisiert, dann kann es einfacher mehr Möglichkeiten für die Beschäftigten bieten (unterschiedliche Projekte, unterschiedliche EVU). Bei STARTUP gibt es keine formalen Hierarchien und die Segregation zwischen den Anwendenden und dem Rest fällt auf: Die Anwendenden sind 450-Euro-Kräfte und haben nicht studiert.

Der Wissensaustausch im Gestaltungsnetzwerk entscheidet, welches Wissen in die Software einfließt. Das Lernen ergibt sich situativ in einer Praxis, in der Software(-Objekte) eine wichtige Rolle spielt(en). Die Softwaregestaltenden der unterschiedlichen Fallstudien verfügen über sehr unterschiedliche interdisziplinäre Wissensstände. Das Spektrum reicht von Softwaregestaltenden, die sich vor allem mit der Koordination der Softwaregestaltung beschäftigen und nur noch wenig inhaltliches Wissen benötigen, bis hin zu IT-Beratenden, die nicht nur Anforderungen aufnehmen, sondern auch gleich noch umsetzen. Bei der **Wissensverteilung** der Softwaregestaltung kann von einer interdisziplinären Praxisgemeinschaft gesprochen werden. Die Praxisgemeinschaft zeigt sich zum einen dadurch, dass jemand, der davon ausgeschlossen ist, nicht mehr Softwaregestaltung betreiben kann und von anderen abhängt, weil ihm das Wissen fehlt. Zum anderen zeigt sie sich dadurch, dass sich die beteiligten Personen in kontinuierlichem Austausch befinden, um Software zu gestalten. So entstehen unterschiedliche Praxisbiografien und damit unterschiedliche Lernbiografien für jeden einzelnen Beschäftigten. Hohlmann hat von einem Gestaltungsnetzwerk gesprochen, welches das interdisziplinäre Wissen hat, um die SAP-Standardsoftware über das Implementierungsprojekt hinaus zu gestalten. In Anlehnung an Wengers *Community of Practice* (1999) kann diese als *Community of Practice and Software Objects* bezeichnet werden. Denn für das, was die Softwaregestaltenden wissen, sind weniger Schulungen entscheidend und vielmehr die Praxis mit anderen und der Software. Teil der Wissensverteilung der Softwaregestaltung sind die verwendeten, softwarebasierten Werkzeuge und die Software selbst. In ihnen materialisiert sich das Wissen (z.B. Quellcode, Ticketsystem, E-Mails, Dokumentationen von Umsetzungen und über Funktionalitäten). Nicht nur Personen sind Träger des Wissens, sondern auch die Software mit dem entsprechend hinterlegten Wissen. Auffallend bei den Interviews war, dass sowohl Anwendende, Programmierende und Gestaltende sagen, dass Learning by Doing für sie wichtig ist – ob aus Dokumenten oder mit der Software selbst. In vielen Fallstudien sind Führungskräfte nicht Teil der Softwaregestaltung und haben entsprechend wenig Wissen über sie oder Softwareentwicklung im Allgemeinen.

Sonst gibt es Besonderheiten je Fall, die aber alle unterschiedliche Varianten der interdisziplinären Praxisgemeinschaft sind: Bei STARTUP ist die gesamte Organisation durch die Kreise darauf ausgerichtet, permanenten interdisziplinären Wissensaustausch zu ermöglichen und eine gemeinsame Wissensbasis zu schaffen. Die Anwendenden sind in diesem Fall weitestgehend ausgeschlossen. Was die Softwaregestaltung anbelangt, ist der Großteil der Anwendenden selbst in Fällen wie INTERN1 vom Wissensaustausch ausgeschlossen, wo sie aktiv in die Softwaregestaltung einbezogen werden. Ihr Wissen ziehen Softwaregestaltende und Programmierende nur temporär für die

Softwaregestaltung heran. Bei KOOP1 besteht die Interdisziplinarität zentral im IT-DL und durch langfristige Zusammenarbeit mit den EVU. Bei KOOP2 fällt bei der internen Softwaregestaltung in EVU2 in den IT-Projekten auf, dass interdisziplinäres Wissen zwischen IT und Fachbereichen fehlt. Das ist eine Folge davon, dass das EVU Teile der Softwaregestaltung erst seit einigen Jahren wieder vom IT-DL zurückverlagert hat. Die Softwarefirma von PAKET besteht aus Entwicklungsteams (Programmierenden, Tester, Teamleitung), die auf energiewirtschaftliche Fachbereiche und die dazugehörenden Softwaremodule der Standardsoftware spezialisiert sind. Bei KOOP3 hat das IT-DL und die IoT-Softwarefirma Wissen über IoT und Implementierungskompetenz für die Standardlösung, wobei das IT-DL das branchenspezifische Wissen aus den Implementierungsprojekten mit EVU in Form von Anforderungen an die IoT-Firma weitergibt.

Im Vergleich zu den Softwaregestaltenden fällt bei den Programmierenden und Anwendenden auf, dass sie auch eigenständig arbeiten, die Führungskräfte nur in Ausnahmefällen eingreifen und die Arbeitsergebnisse kontrollieren. Zudem sind sie wie die Softwaregestaltenden Teil eines softwarezentrierten Arbeitsprozesses mit anderen Schwerpunkten, welche Software sie für ihre tägliche Arbeit hauptsächlich verwenden (ERP-System im Kund:innenservice, mobile Endgeräte in der Instandhaltung, Entwicklungsumgebung und Ticketsystem in der Programmierung etc.). Auch für Programmierende und Anwendende prägt die Grundkoordination von Hierarchie, Markt und Netzwerk die Kontrolle. Die **Anwendenden** sind hierarchisch und von der Bezahlung Programmierenden und Gestaltenden untergeordnet. Die Anwendenden sind meist passiv und reagierend: Sie müssen sich auf Änderungen einstellen oder erfahren erst nach Fertigstellung, was sich geändert hat. Ihre Arbeit ist in den meisten Fällen das, was die Software nicht erledigen kann bzw. noch nicht in dieser abgebildet ist. Das bedeutet auch, dass sie fehlerhafte oder nicht ausgereifte Software kompensieren müssen. Sie fungieren als Puffer für mangelhafte Softwareentwicklung. Für einen Teil der Anwendenden findet eine *qualitative Intensivierung* statt, weil sie vermehrt in Prozesszusammenhängen denken müssen (Sachbearbeitende), mehrere Fachgebiete (Key User:innen) oder mehrere Netzgebiete und Sparten kennen müssen (Monteur:innen). Arbeiten Anwendende im BPO-Bereich für mehrere EVU, ist mit einer *quantitativen Intensivierung* zu rechnen. Wobei dies bei einigen EVU auch unabhängig davon aufgrund von Personalmangel oder phasenweise durch IT-Projekte entsteht. Die Arbeit der **Programmierenden** wird durch die Tests anderer und/oder durch Code-Reviews von Kolleg:innen geprüft. Es ist durch die Ticketsysteme transparent, was sie abgearbeitet haben, und falls es regelmäßige Treffen wie Daily bei Scrum gibt, müssen sie den Teammitgliedern Rede und Antwort stehen. Ihre Spezialisierung und die meist schwierige Abschätzung des Arbeitsaufwandes geht mit einer fehlenden detaillierten Kontrolle von außen einher. Eine *qualitative Intensivierung* existiert bei den befragten Programmierenden, weil sie fachliches mit IT-Wissen kombinieren und wenn sie in verschiedenen Projekten, Rollen oder für mehrere EVU tätig sind. Wie bei den Gestaltenden ist es bei den Programmierenden schwierig, pauschal von einer *quantitativen Intensivierung* zu sprechen. Sie arbeiten in den untersuchten Fällen meist Programmieraufgaben ab, für die Prioritäten vorgegeben sind. Das lässt Spielraum, eigene Grenzen zu setzen. Eine

Intensivierung ist aber nicht ausgeschlossen, wenn z.B. eine fixe Deadline für eine zu programmierende Funktionalität existiert.

8.6.2.4. Folgen für die Arbeitsgestaltung in den EVU

Die Fallstudien zeigen, dass eine Folge der Softwaregestaltung ist, dass sie die Arbeit der Softwareanwendung gestaltet. Sie ist damit Teil der Kontrolle der Softwareanwendung. Dies tut sie durch einen soziotechnischen Wandel von Software, Organisation und Arbeit. Dabei dienen die Kategorien von Einfluss und Konflikten dazu, das Verhältnis von Softwaregestaltung und -anwendung zu untersuchen. Der Fallvergleich konnte zusätzlich dazu vier Unterkategorien des Einflusses herausarbeiten, welche das Verhältnis der beiden Arbeitsprozesse von Softwaregestaltung und -anwendung beschreiben helfen: Kontrollverhältnis, Partizipation, Reorganisation und Ziele. Zur Partizipation der Anwendenden und einer möglichen Reorganisation der EVU hin zu einer softwaretechnischen Prozessorganisation hat das Kapitel jeweils allgemeine Thesen diskutiert, die hier noch einmal kurz zusammengefasst werden. Beim Konflikt zwischen Management und Angestellten nimmt der Betriebsrat in den Fällen eine intervenierende Funktion bei der Arbeitsgestaltung ein. Welche Rolle das Management bei der Arbeitsgestaltung durch Softwaregestaltung überhaupt noch spielt, diskutiert Abschnitt 8.6.4 weiter unten.

Was den **Einfluss** der Softwaregestaltung auf die Arbeitsgestaltung anbelangt, unterscheiden sich die Fallstudien dahingehend, ob sie für mehrere EVU und damit viele Anwendende Software gestalten oder nicht. Dabei ändert der Arbeitsprozess der Softwaregestaltung in nur wenigen Fällen nicht nur die Software, sondern auch die Organisation der Softwareanwendung (z.B. INTERN1). Bei den Fällen, in denen EVU intern Software gestalten, fallen die **Konflikte** innerhalb der Organisationen mehr auf (z.B. zwischen Unternehmens- bzw. Management- und Anwendenden-Anforderungen). Bei den Fällen, in denen mehrere Organisationen zusammenarbeiten, treten die Konflikte zwischen diesen besonders hervor. Wobei es dann vor allem darum geht, zwischen Kooperation und individuellen Alleingängen abzuwägen und die Erwartungen dementsprechend abzugleichen. Bei Fallstudien der Grundkoordination Netzwerk sind die Konflikte bezüglich der Arbeitsgestaltung schwach ausgeprägt, wie z.B. bei STARTUP. Weder unterschiedliche Marktagierende wie IT-DL, Softwarefirma und EVU noch unterschiedliche Abteilungen und deren Führungskräfte stehen sich gegenüber.

Je nach Fallstudie ist der **Betriebsrat** unterschiedlich eingebunden, fokussiert sich aber immer auf die Folgen für die Anwendenden und interveniert. Grundsätzlich ist er Teil des Konflikts in der Softwaregestaltung zwischen Beschäftigten und Management. Er kann Rahmenbedingungen setzen, kann Informationen einfordern (bei entsprechenden Betriebsvereinbarungen) und individuelle Leistungskontrolle verhindern. In den Fallstudien fehlt ihm das Wissen, Anforderungen für die Softwaregestaltung zu stellen, und er ist nicht in den Arbeitsprozess eingebunden. Findet die Softwaregestaltung intern statt, kann er dafür sorgen, dass Beschäftigte mitgestalten dürfen. Bei externer Softwaregestaltung gibt es Fälle, bei denen er bei der Auswahl von IT-DL interveniert.

Um den Einfluss der Softwaregestaltung auf die Softwareanwendung tiefergehend zu analysieren, hat das Kapitel die Unterkategorien Kontrollverhältnis, Partizipation, Reorganisation und Ziele verwendet. Erstens ist die soziotechnische Arbeitsgestaltung

des Anwendungsbereichs durch die Softwaregestaltung ein **Kontrollverhältnis**. In der Mehrzahl der Fallstudien kontrollieren die beteiligten Organisationen die Arbeitsprozesse von Softwaregestaltung und Softwareanwendung jeden für sich, z.B. als Teil von unterschiedlichen Teams oder Abteilungen. In solchen Fällen haben die Führungskräfte der Anwendung oftmals genauso viel oder gar noch mehr Macht als jene der Softwaregestaltung. Wenn die gesamte Organisation von Anfang an auf die Softwareentwicklung ausgerichtet ist, wird, wie schon unter 4.1 ausgeführt, vom Primat der Softwareentwicklung gesprochen. Das ist bei STARTUP so, wo die Fallstudie zeigt, wie Organisationen den Arbeitsprozess der Softwareanwendung zum Nutzen für die Softwaregestaltung organisieren, um die Möglichkeiten der Softwareentwicklung auszureizen. Das bedeutet, dass die Arbeit der Anwendenden nicht nur durch die Software kontrolliert wird, sondern auch durch den Arbeitsprozess der Softwaregestaltung.

Eine direkte **Partizipation**, bei der die Anwendenden alle in die Softwaregestaltung einbezogen werden und auch genug Wissen haben, um mitzugestalten, existiert in den Fallstudien nicht. Selbst dann existiert sie nicht, wenn die Anwendenden Anforderungen aufgeben können. Auch dann existiert sie nicht, wenn die EVU den Anwendungsbereich bzw. die Softwareanwendung nicht der Softwaregestaltung unterordnen (indem z.B. die Führungskraft der Softwaregestaltung hierarchisch mit jener der Softwareanwendung gleichgestellt ist). In den meisten Fällen haben Anwendende nur über Vermittler Zugang zur Softwaregestaltung oder es gibt formale Wege, Anforderungen zu stellen, bei denen letztendlich andere entscheiden, welche der Anforderungen wie umgesetzt werden. Oftmals gestalten Fachleute Software und in manchen Fällen suchen die Organisationen einen Ausgleich zwischen Anwendenden- und Unternehmensinteressen, wobei das Management das letzte Wort hat. Die drei analytischen Typen der Partizipation bei der Softwaregestaltung (siehe die Diskussion unter 8.5.3.3) untermauern diese These.

Die Arbeitsgestaltung unterscheidet sich im Fallvergleich dahingehend, welche **Ziele** in puncto Softwaregestaltung die EVU selbst festlegen können – hinsichtlich ihrer Organisation und der Kommodifizierung von Software. Bei Fragen der Organisation geht es z.B. darum, ob ein EVU von seinem Status quo aus anfängt, sich so zu organisieren, ob es selbst Software gestalten kann oder die Softwaregestaltung an einen IT-DL auslagert. Bei der Kommodifizierung geht es darum, ob eine Softwarefirma eine Standardsoftware programmiert, die viele EVU einsetzen und die damit skalierbar ist. Das können einzelne Teile einer Software, Applikationen oder wie bei STARTUP Softwareteile sein, welche eine Organisation selbst nutzt und gleichzeitig anderen anbietet. Auf jeden Fall zeigen die Fallstudien, dass Software strategisch einen Unterschied machen kann und keine Standardware wie Bürostühle oder Seife ist (siehe bereits 4.2.2). Strategisch macht Software vor allem dann einen Unterschied, wenn von ihr die individuelle Wettbewerbsfähigkeit abhängt. Das zeigt sich daran, dass es den EVU leichter fällt, einen Standard für den regulierten Netzbereich einzusetzen, der nicht dem Marktwettbewerb ausgesetzt ist. Für den vertriebs- und wettbewerbsorientierten Lieferbereich fällt es EVU schwerer. Zuletzt betreffen die Ziele, inwiefern eine Organisation mit Software oder Softwaregestaltung Geld verdient (z.B. ob als IT-Dienstleistung oder mit einem fertigen Softwareprodukt). Die IT-DL von KOOP1, KOOP2 und KOOP3 gehören EVU, verdienen Geld mit Softwaregestaltung und Standardsoftware und die Softwarefirma von PAKET lebt allein vom Softwaregeschäft.

In den Fallstudien zeigt sich – viertens – der Einfluss der Softwaregestaltung auf die -anwendung daran, ob sich die EVU **reorganisieren**. Es gibt Fallstudien, in denen organisationale Hürden die Softwaregestaltung im jeweiligen Anwendungsbereich nicht mehr bremsen. Das bedeutet aber nicht in jedem Fall, dass EVU die Anwendung oder Entwicklung reorganisieren. Zum Beispiel verbleiben bei INTERN1 und INTERN2 die Programmierenden in der IT-Abteilung und die bestehende Arbeitsteilung zwischen Teams bleibt bestehen (inkl. getrennter IT-Budgets). In einzelnen Fallstudien ändert die Softwaregestaltung weder etwas an den Arbeitsabläufen noch den Arbeitsstellen, sondern nur etwas an der Software. Im Extremfall des Primats der Softwareentwicklung (Fall STARTUP) gab es keinen energiewirtschaftlichen Status quo und die Organisation baut sich um die Software herum auf und verändert sich mit ihr. Von Anfang an war die Organisation auf die Softwareentwicklung ausgerichtet. Um zu verdeutlichen, wie eine Reorganisation im Sinne der Softwaregestaltung innerhalb eines EVU aussehen könnte, hat die Untersuchung den Organisationstyp der auf Softwaregestaltung ausgerichteten, softwaretechnischen Prozessorganisation ausgearbeitet (siehe 8.5.3.3). Er untermauert die These, dass sich die untersuchten EVU (und wohl viele andere auch) noch gar nicht auf die Softwaregestaltung ausgerichtet haben. Sie denken zwar darüber nach, doch nutzen sie Rationalisierungspotenziale durch eine verstärkte organisationale Ausrichtung auf die Softwaregestaltung nicht – auch wenn das EVU von INTERN2 darüber nachdenkt.

8.6.3. Synthese: Rationalisierungstyp der technikentwicklungsbezogenen Rationalisierung

Die Fallstudien zeigen, dass die EVU auf unterschiedliche Weise Softwaretechnikgestaltung betreiben: mal einen Standard anpassen, mal von Grund auf etwas individuell Neues gestalten; mal durch einen IT-DL gestalten lassen, mal selbst gestalten. Eine andere Variante ist, dass sie wie bei der Fallstudie STARTUP von Beginn an Arbeit und Organisation auf eine individuelle Softwaregestaltung ausrichten. Aber egal, wie sie es machen: Indem sie die Möglichkeiten der Softwaregestaltung nutzen, rationalisieren sie dadurch die Anwendungsbereiche.

Um zu untermauern, dass Softwaregestaltung die Anwendungsbereiche rationalisiert, stellt die vorliegende Untersuchung die Hypothese auf, dass es sich bei den unterschiedlichen Ausprägungen der Softwaregestaltung in den Fallstudien um verschiedene Formen einer technikentwicklungsbezogenen Rationalisierungsform handelt. Dabei rationalisieren die Organisationen – je nach Fall EVU, IT-DL, Softwarefirmen – sowohl die soziotechnische Arbeitsgestaltung der Softwareanwendung durch die Softwaregestaltung als auch die soziotechnische Netzwerkarbeit der Softwaregestaltung selbst. Die Softwaregestaltung zu kontrollieren, ist die Voraussetzung dafür, dass die Organisationen deren Möglichkeiten der Rationalisierung zwischen Standardsynergien und individueller Differenzierung im Wettbewerb nutzen können. Damit ist nicht gesagt, dass es einen »one best way« der Rationalisierung gibt und alle Organisationen z.B. dem Primat der Softwareentwicklung folgen sollten. Technikentwicklungsbezogene Rationalisierung ist kontingent und reflexiv und bedeutet, dass Organisationen einer Branche das Mittel der Softwaregestaltung und -programmierung für die eigenen Zwecke und Bedin-

gungen zweckorientiert im jeweiligen Kontext einsetzen. Weil sich viele Tätigkeiten und Prozesse durch Software effizienter oder überhaupt nur noch so wettbewerbsfähig erledigen lassen, ist diese Form der Rationalisierung Teil vieler Organisationen. Sie zeichnet sich durch eine Dynamik zwischen Standard- und Individualsoftware, dezentraler und zentraler Softwaregestaltung aus.

Auch wenn Kommunikation eine zentrale Rolle bei der Softwaregestaltung spielt, unterscheidet sich die technikentwicklungsbezogene Rationalisierung von der kommunikativen Rationalisierung, wie sie von Rock/Ulrich/Witt (1990) beschrieben wurde. Letztere hat als Rationalisierungsgegenstand die sozialen Interaktionsprozesse selbst, unabhängig davon, ob es dabei um die Softwareentwicklung geht. Das ist bei der Softwaregestaltung anders, wo die Kommunikation Mittel zum Zweck der Softwaregestaltung ist. In Anlehnung an die Gegenüberstellung der drei Rationalisierungstypen (tayloristisch, systemisch, kommunikativ) (vgl. ebd.: 74) führt die unten stehende Tabelle 30 die Merkmale der technikentwicklungsbezogenen Rationalisierung auf. Diese Form der Rationalisierung stellt eine Synthese aus dem Konzept der soziotechnischen Netzwerkarbeit, der soziotechnischen Arbeitsgestaltung, dem Analyserahmen und dem Fallvergleich dar.

Ausgehend von diesem Rationalisierungstyp betrachtet man eine Branche und ihre Organisationen aus einer neuen Perspektive. Gegenstand der Rationalisierung ist die Technikgestaltung selbst und weniger die Techniknutzung. Es geht darum, die Möglichkeiten der Softwaregestaltung für einen Anwendungsbereich zu nutzen. Das bedeutet zum einen, dass den technikentwicklungsbezogenen Rationalisierungstyp eine stetige Dynamik aus individueller und Standardsoftwaregestaltung auszeichnet, die sowohl innerhalb der Organisationen als auch in der Branche zwischen Organisationen wie IT-DL, Softwarefirmen und EVU besteht. Für einen industriespezifischen Anwendungsbereich kann es sich im Zeitverlauf von Jahrzehnten mehrmals ändern, ob dort Unternehmen eine Standardlösung einsetzen oder eine individuelle Lösung. Darüber hinaus können Organisationen Software je nach Anwendungsbereich unterschiedlich nutzen. In Anwendungsbereichen mit einem hohen Anteil an Datenverarbeitung kann Software Arbeit ersetzen. Bei anderen steuert oder überwacht Software Arbeit, die nicht digitalisiert werden kann.

Anders als im Taylorismus steht deshalb nicht die Arbeitsteilung zwischen Hand- und Kopfarbeit im Zentrum der Rationalisierung. Es geht nicht darum, dass Kopfarbeitende die Arbeit der Handarbeitenden kontrollieren, so dass die Handarbeitenden nur noch möglichst wenig denken müssen. Vielmehr geht es unabhängig davon, ob die Anwendenden mehr ihre Hände oder mehr ihren Kopf benutzen, um die Trennung von Softwareanwendung, -gestaltung und -programmierung. Die einen machen die Software, welche die anderen in ihrer Arbeit anwenden, und das in vielen Fällen ihren gesamten Arbeitstag über und ausschließlich.

Dabei sind Grundlage der technikentwicklungsbezogenen Rationalisierung keine klar strukturierten Aufgaben. Es geht um Anforderungen, die Programmierende für die Software eines Anwendungsbereichs umsetzen müssen. Diese müssen erst Softwaregestaltende erarbeiten und den Programmierenden übergeben, so dass dann auch die Software macht, was sie machen soll. Dabei sind die wesentlichen Mittel Wissen und Kommunikation inkl. Feedbackschleifen zwischen Anwendung und Programmierung. Dieser kommunikative Austausch und das Erarbeiten wissensbasierter Anforderungen

gelingt in organisationalen und interpersonalen Netzwerken besser als in hierarchischen Organisationen oder Marktbeziehungen. Die soziotechnische Netzwerkarbeit der Softwaregestaltung zeichnet aus, dass sie keiner Best Practice folgt und nicht detailliert vorgeplant werden kann. Sie ist vielmehr situativ und abhängig von der soziotechnischen Konstellation, zu der die Arbeitsteilung zwischen Anwendung und Entwicklung (z.B. zwischen IT-DL und EVU oder IT-Abteilung und Fachabteilungen innerhalb eines EUV) und die Softwarearchitektur gehört. Letztere legt fest, wer was gestalten kann (z.B. ob Schnittstellen vorhanden oder individuelle Einstellungen an einem Standard möglich sind) und welche Abhängigkeiten bestehen (z.B. wenn mehrere Organisationen sich auf einen Standard verständigen müssen). Das Organisationsprinzip der Situativität zeigt sich an angepassten Rollen (z.B. Product Owner oder IT-Projektleitung) und Abläufen (z.B. Scrum), welche die zu gestaltende Software als Gegenstand haben und softwarebasierte Werkzeuge (z.B. Ticketsysteme) einsetzen.

Die Kernprobleme der soziotechnischen Interdisziplinarität und Gestaltungsmöglichkeiten löst der Arbeitsprozess der Softwaregestaltung entweder als Teil einer Matrix- oder reinen Netzwerkorganisation und zentral oder dezentral. Das hat Folgen für die Softwareanwendung: Die anwendende Organisation kann unabhängig oder abhängig sein und die eigene Organisation auf die Softwareentwicklung ausrichten und für sich entscheiden, ob Synergien durch einen Standard wichtiger sind als Wettbewerbsvorteile durch eine individuelle Software.

Die Dynamik aus Standard – individuell und dezentral – zentral zeigt sich auch bei den möglichen Strategien. Der technikentwicklungsbezogene Rationalisierungstyp zeichnet sich dadurch aus, dass es grundsätzlich zwei unterschiedliche Strategien gibt: 1. Spezialisierung auf Anwendung oder Entwicklung in getrennten Organisationen oder Abteilungen; 2. Anwendung und Entwicklung gemeinsam in einer Organisation in Form des Primats der Softwareentwicklung. Letzteres ermöglicht die Umsetzung der Strategie des Competing on Complexity, die laut Bessen (2022) erfolgreiche Unternehmen in softwareintensiven Industrien auszeichnet (Näheres siehe 4.1). Andererseits kann es für Organisationen am effizientesten sein, für bestimmte Anwendungsbereiche eine Standardsoftware einzusetzen und sich rein auf die Anwendung zu konzentrieren. Dass kann sich aber, wie oben bereits geschrieben, über die Zeit ändern.

Die Softwaregestaltung zeichnet sich dadurch aus, dass selbstständiges Lernen und Eigenmotivation wichtig sind (Subjektivität). Der Führungsstil ist nicht autoritär wie im Taylorismus und ergebnisorientiert und Führungskräfte sind vor allem aktiv, wenn Ausnahmen auftreten oder die Ergebnisse für sie nicht zufriedenstellend sind. Auch wenn es in den Fallstudien nicht vorkam, so wäre der Coaching-Ansatz für die Führung wohl der passende. Den Softwaregestaltenden geht es nicht darum, in Hierarchien aufzusteigen oder disziplinarische Verantwortung zu übernehmen. Sie wollen interessante Projekte machen und etwas dazulernen. Solche Kompetenzkarrieren schließen aber nicht aus, dass es Gehaltsstufen gibt, auf denen Softwaregestaltende aufsteigen können. Die eingesetzten Werkzeuge dienen in erster Linie dazu, den Softwaregestaltungsprozess von Anwendung bis Programmierung digital zu integrieren, die Kommunikation zu ermöglichen und Wissen zu speichern (u.a. Dokumentation von Anforderungen und deren Umsetzung).

Tabelle 30: Rationalisierungstyp der technikentwicklungsbezogenen Rationalisierung, angelehnt an Rock/Ulrich/Witt (1990: 74)

Merkmal	Ausprägung
Rationalisierungsgegenstand	Softwaregestaltung für industriespezifische Anwendungsbereiche
Arbeitsteilung	Anwendung und Entwicklung (inkl. Gestaltung)
Kernaufgaben	Anforderungen[24]
Organisationsziel	Kommunikation inklusive Feedbackschleifen (mündlich oder schriftlich)
Kernkontrollform	Netzwerk statt Markt oder Hierarchie
Organisationsprinzip	rollen-, ablauf- und softwarebasiert; situativ je soziotechnische Konstellation
Besondere Organisationsdeterminante	Softwarearchitektur inkl. softwaretechnischer Zuschnitt (Standard, individuell)
Kernprobleme	- softwaretechnische Interdisziplinarität - softwaretechnische Gestaltungsmöglichkeiten: organisatorische Ausrichtung (nur auf die Softwareanwendung oder sowohl auf Softwareanwendung wie auch -entwicklung), Zuschnitt (Standard/individuell)
Rationalisierungsbereiche	Arbeitsprozess der Softwaregestaltung: - zwischen Matrixorganisation und reiner Netzwerkorganisation - zwischen dezentral und zentral
	soziotechnische Arbeitsgestaltung der Softwareanwendung - zwischen unabhängig und abhängig
Strategien	zwischen *Competing on Core Competency* (Anwendung oder Entwicklung) und *Competing on Complexity* (Anwendung und Entwicklung)
Ausrichtung anwendende Organisation	zwischen Primat der Softwareentwicklung und Anwendung Standardsoftware
Personalentwicklung	selbstständiges Lernen in Bezug auf Software und Softwaregestaltung
Führungsstil	Management by Exception and Results; Coaching
Typische Karriere	fachliche Kompetenzkarriere ohne disziplinarische Verantwortung
Softwarebasierte Werkzeuge	prozessintegrierend von Anwendung bis Programmierung für Kommunikation und als Wissensspeicher

8.6.4. Neue Konkurrenz für das Management durch die Softwaregestaltenden?

Beim Arbeitsprozess der Softwaregestaltung fungieren Führungskräfte mehr als Rahmengeber und sind nur teilweise direkt involviert. Vielmehr sind Softwaregestaltende und Programmierende Träger der Arbeitsgestaltung. Welchen Anteil hat das Management noch an der Arbeitsgestaltung? Inwiefern gibt es Konflikte innerhalb des Managements? Können die Softwaregestaltenden als Teil des Managements betrachtet werden?

24 Andere Begriffe dafür sind Konzepte, Spezifikationen, Stories, Tickets; zu Anforderungen gehört auch mündliches oder schriftliches Feedback wie z.B. durch Tests.

Insgesamt ist das Management abhängig von den Softwaregestaltenden. Sie können nicht allein die optimale Nutzung der Softwaretechnik umsetzen. Bei den Arbeitskreisen von PAKET oder der Anforderungsrunde von INTERN2 sitzen ausgewählte Expert:innen. Bei INTERN1 ist das Management einer neben vielen anderen Anforderungsstellenden. Bei KOOP1 und KOOP2 übernehmen IT-DL oder intern in den EVU IT-Projektleitende, Applikationsbetreuende oder Prozessverantwortliche die Softwaregestaltung. Nur in wenigen Fällen machen Teamleitung oder Gruppenleitung bei der Softwaregestaltung operativ mit: EVU von PAKET, Teamleitung von KOOP2 oder die Geschäftsführung von STARTUP. Dabei gestalten sie dann aber nur Teile einer Software mit.

Vielmehr setzt das Management die Rahmenbedingungen. Die Kontrolle über finanzielle und personelle Ressourcen und strategische Entscheidungen liegt beim Management. Es entscheidet über die betriebsinternen Karrieren – egal ob von Software anwendendem, gestaltendem oder programmierendem Personal. Wenn die Softwareentwicklung komplett ausgelagert ist, dann richtet sich der Fokus des Managements der EVU darauf, die IT-DL, IT-Beratenden oder Softwarefirmen zu kontrollieren. Bei INTERN1 und INTER2 genehmigt das Management die Softwaregestaltung. Bei KOOP1 hat das Management über die Auslagerung der IT entschieden und das Anforderungsmanagement eingeführt. In einem EVU kontrolliert der Digitalisierungsmanager Anforderungen, bevor sie an das IT-DL weitergeleitet werden. Beim strategischen Anforderungsmanagement von KOOP1 machen die leitenden Personen der IT-Abteilungen aus den EVU mit. Bei KOOP2 hat das Management eines EVU entschieden, wieder Teile der Softwaregestaltung vom IT-DL zurückzuverlagern. Bei KOOP3 hat das Management des IT-DL beschlossen, einen IoT-Bereich mit entsprechendem Personal zu gründen. Nur beim STARTUP gibt es keine formalen Hierarchien. Doch basiert der Primat der Softwareentwicklung darauf, dass die Gründer über Wissen zur Softwareentwicklung verfügen und die softwaretechnische Interdisziplinarität verkörpern.

Dann unterscheidet es sich von Fall zu Fall, ob das Management und bestimmte Führungsebenen ein Hindernis für die Softwaregestaltung sind oder diese selbst übernehmen. In Matrixorganisationen sind sie oftmals wichtige Sponsoren z.B. für IT-Projekte, um sie abteilungsübergreifend durchzusetzen. Bei INTERN1 und INTER2 erleben Befragte Führungskräfte auf Ebene der Fachbereichsleitung als hinderlich für die Softwaregestaltung und es gibt Kompetenzkonflikt[e(?)] zwischen Abteilungen (IT, Fachbereiche). Bei KOOP2 werden Hierarchien in Frage gestellt. In einem EVU hat das Management die Hoheit über die Gestaltung vom IT-DL zurückgeholt (u.a. dadurch, dass das EVU wieder das Projektmanagement selbst macht). Bei PAKET gibt es ein EVU, in dem die Führungskraft eines Fachbereiches über die Wahl eines Softwareunternehmens entscheidet, ohne die IT-Abteilung zu fragen. Im STARTUP herrscht ein anderes Führungsverständnis vor. Die Gründer sind höchstens informell führend aufgrund ihrer Kompetenzen. Darüber, wer welche Rollen übernimmt, entscheiden die Mitarbeitenden gemeinsam.

In den Fallstudien treiben die Softwaregestaltenden die Technikgestaltung in den Organisationen voran. Sie helfen damit, ein Kernproblem des Kapitals zu lösen, wodurch sie eine bestimmte Position für sich beanspruchen können (vgl. Armstrong 1985). So bestimmt sich, wie schon bei Hohlmann beschrieben, ihre Position technikinduziert: Sie sind notwendig, wie z.B. die Rolle von Key User:innen, damit die Organisation die Tech-

nologie einsetzen kann (vgl. Hohlmann 2007 zum Einsatz der SAP ERP-Software). Einerseits sind Softwaregestaltende, indem sie Arbeit gestalten, Teil des Managements. Sie helfen, die Arbeit der Anwendenden zu kontrollieren. Andererseits: Wenn es um das Beschäftigungssystem geht, ist bei den hier untersuchten Fällen noch nicht zu erkennen, dass es zu einer direkten Konkurrenz zum existierenden Management kommt. Vielmehr entsteht eine eigenständige Gruppe, die bestehende Karrierewege in Frage stellt und teilweise eigene bekommt. Ob in Zukunft in den EVU vor allem Softwaregestaltende oder -programmierende aufsteigen: Dafür gibt es keine Indizien. In den Fallstudien sehen sie sich nicht als Teil des Managements. Programmierende und Gestaltende wollen größtenteils keine disziplinarische Verantwortung haben und Teil des Managements werden. Laut Pohlmann zeichnet das Management neben Funktionen wie Gestaltung, Kontrolle und Koordination auch eine hierarchische Position aus, mit der Führung und Weisungsbefugnisse einhergehen (vgl. Pohlmann 2017: 207f.). Das ist bei den meisten hier befragten Softwaregestaltenden nicht der Fall und das Management hat die Arbeitsgestaltung durch Softwaregestaltung delegiert oder ausgelagert. Wie STARTUP und die diskutierte softwaretechnische Prozessorganisation (siehe 8.5.3.3) zeigen, werden Teile des Managements überflüssig, wenn eine Organisation sich auf die Softwaregestaltung ausrichtet.

8.6.5. Facetten einer industriespezifischen Softwaregestaltung

Die Organisationen in den Fallstudien gestalten eine industriespezifische Software. Es zeigen sich mehrere Eigenheiten der Softwaregestaltung in der Energiewirtschaft: die industriespezifische Detailtiefe und Komplexität, die große Bedeutung der Regulierung, die Arbeitsteilung zwischen großen und kleinen EVU, die lokale Verankerung der EVU, der zögerliche Wandel und die noch offene Frage, welche Rolle Softwaregestaltung in Zukunft für die Wettbewerbsfähigkeit von Firmen in der Branche spielen wird.

Es gibt vielfältige Wissensdomänen und ein komplexes technisches und regulatorisches Umfeld, die eine Herausforderung für die Softwaregestaltung in der Energiewirtschaft darstellen. Im 7. Kapitel wurden einige der vielen Gesetze und Verordnungen genannt. Dazu kommen technisch komplexe Anlagen – ob Kraftwerke oder Netze mit ihren Zählern, Trafostationen und verschiedenen physikalischen Eigenheiten (Gas, Strom). Dazu kommt, dass trotz Entflechtung viele Stadtwerke noch Konzerne mit unterschiedlichen Geschäftsfeldern inklusive Schwimmbädern und öffentlichem Nahverkehr sind. Typisch für die Industrie ist, dass, obwohl in anderen Industrien ähnliche Anwendungsbereiche (z.B. Instandhaltung) existieren, diese trotzdem individuell sind. INTERN1 kann durch die interne Softwaregestaltung eine industriespezifische Instandhaltungssoftware entwickeln, z.B. mit Notruffunktion für Monteur:innen. Die individuelle Softwaregestaltung ist darauf ausgelegt (wie ein Befragter meint), Netzkonzession zu bekommen. Das heißt, die Softwaregestaltung richtet sich an institutionelle Begebenheiten, die darin bestehen, dass die netzbesitzende Organisation (meist die Kommunen) den Netzbetrieb vergibt und spezifische Erwartungen hat.

Zudem ist die große Bedeutung der Regulierung für die Softwaregestaltung und die damit einhergehenden stetigen Softwareanpassungen etwas Industriespezifisches. Ein großer Teil der Softwaregestaltung wäre ohne den stetigen Wandel der Regulierung nicht

notwendig. Das STARTUP existiert nur aufgrund einer Regulierung (Emissionshandel) und ihre Dienstleistung ist es, diese digital umzusetzen. Für die IT-DL, Softwarefirmen und allgemein die Softwaregestaltenden ist die sich stetig ändernde Regulierung und die neuen Anforderungen durch die Energiewende Geschäftsgrundlage und Existenzberechtigung. Zudem stellt die Regulierung eine Grundlage für Synergien und Kooperationen bei der Softwaregestaltung dar: Sie ist für alle gleich, und so stellt sie einen Ausgangspunkt dar, um zu kooperieren und zumindest für die regulierten Bereiche eine industriespezifische Standardsoftware zu entwickeln. Das zeigt sich in der Softwaregestaltung vor allem bei KOOP1 und PAKET, wo ein IT-DL bzw. eine Softwarefirma für mehrere EVU einen Standard ausprägt. Die Unterscheidung zwischen regulierten und wettbewerblichen Bereichen wird in der Softwaregestaltung zur Frage, was EVU kooperativ und was sie individuell programmieren.

Das Besondere an der Energiewirtschaft in Deutschland sind die vielen Stadtwerke. Sie stellen einen großen Markt für Software dar. Vor allem die kleineren EVU zeigen sich in den Fallstudien als Trittbrettfahrer, wenn es um Softwaregestaltung geht. In den Fallstudien sind es eher die mittleren und großen EVU, die sich an der Entwicklung eines Standards beteiligen. Dies geschieht bei einem IT-DL wie bei KOOP1 oder KOOP2 oder indem EVU mit Softwarefirmen Projekte machen oder extra Personal haben, um Anforderungen zu stellen und zu testen wie bei PAKET. So profitieren die kleineren EVU davon, was an Software entwickelt wird. Ein EVU aus PAKET oder die IT-DL von KOOP1 und KOOP2 stellten die entwickelte Software anderen (meist kleineren) EVU zur Verfügung. Das kann so weit gehen, dass auf dem gleichen Server für mehrere EVU die Anwendungssoftware läuft und sie ein EVU oder ein IT-DL betreibt und gestaltet. Bei KOOP1 und PAKET sprechen Befragte explizit davon, dass IT-DL oder Softwareunternehmen Stadtwerke »spielen« können und dies auch tun. Es wird in den Fällen auch angeboten (wie auch in KOOP2), einzelne Prozesse via BPO für EVU zu übernehmen, d.h. nicht nur die Software zur Verfügung zu stellen, sondern auch die Anwendenden.

Speziell an der Branche ist zudem die Kultur, zu der Kooperationen, ein regionaler Bezug und zögerlicher Wandel innerhalb der EVU gehören. Einerseits erleichtert die Kultur die Kooperation wie bei KOOP1. Die kooperative Kultur stammt wohl aus einer Zeit, in der kein Wettbewerb herrschte und die Stadtwerke die gleichen Herausforderungen hatten, aber keinen gemeinsamen Markt. Immer noch gibt es eine getrennte, regionale Verantwortung und begrenzten Wettbewerb. Gleichzeitig zeigt KOOP2, dass Kooperationen scheitern können. Die Branche zeichnet sich zudem durch einen regionalen Bezug aus, der sich in spezifischen Netzen (INTERN1), regionaler Kundschaft (KOOP1, KOOP2) und regionalen Arbeitsmärkten niederschlägt (einzelne Befragte bei KOOP2 und PAKET). Für die Softwaregestaltung bedeutet das zum einen, die Anforderungen vor Ort aufzunehmen, und zum anderen den Vorteil, Fachkräfte aus mittelgroßen oder kleineren Städten gewinnen zu können, weil die großen Konzerne und IT-Unternehmen dort meist keine Niederlassungen haben. In der Branche zeigt sich ein zögerlicher Wandel wie bei INTERN1 und INTERN2 oder den EVU aus KOOP1 und KOOP2: ob bei der Durchsetzung gemeinsamer Werkzeuge für die Softwaregestaltung, der teamübergreifenden Zusammenarbeit oder der Einführung neuer Methoden wie Scrum. Die EVU versuchen, die Wettbewerbsfähigkeit ihrer althergebrachten Strukturen zu erhalten, und transformieren sie nur inkrementell. Bei KOOP3

fungiert das IT-DL als Mittelsmann in die Branche für das IoT-Softwareunternehmen. Es kann als übersetzende Organisation zwischen der Start-up-Welt und der EVU-Welt gesehen werden. Der Fall zeigt, dass innovativere Technologien wie IoT ausgelagerte oder branchenfremde Organisationen gestalten und die EVU, wenn überhaupt, dann eine fertige Standardlösung erweitern (wie EVU1 von KOOP3). EVU2 von KOOP2 hat eine Person eingestellt, welche die Projektkultur fördern soll. An den althergebrachten energiewirtschaftlichen Strukturen hält das EVU aber fest: Die funktionale Aufteilung nach fachlichen Abteilungen und Teams bleibt bestehen (Abrechnung, Netzanschluss, Energiehandel, IT-Abteilung etc.) und für die Softwaregestaltung ergänzt das EVU die Organisation um Positionen für IT-Projektmanagement, IT-Koordination und Key User:innen.

Nimmt man die Matrix der vier Typen der soziotechnischen Arbeitsgestaltung (8.6.1.3), so ging es in keiner der Fallstudien um reine Auftragsarbeiten oder um softwarebasierte EVU, deren proprietäre Software zu einem Standard in der Branche wird und die sich disruptiv mithilfe von Softwareentwicklung durchsetzen können. START-UP entwickelt zwar eine proprietäre Software, die sie anderen anbietet, jedoch für eine sehr spezielle Funktionalität (Emissionshandel). In sämtlichen anderen Fällen geht es um Standardsoftware, die bis auf PAKET und einige EVU bei KOOP1, KOOP2, KOOP3 die EVU weiterentwickeln. Es ist und bleibt eine offene Frage, welcher der vier Typen der Arbeitsgestaltung aktuell und in Zukunft quantitativ dominiert. Von den untersuchten Fällen steht kein EVU kurz davor, in naher Zukunft aus dem Markt gedrängt zu werden. Somit bestätigen die Fallstudien, was unter 7. geschrieben wurde: Das Überleben in der Branche sichern unterschiedliche Formen der Softwaregestaltung. Wobei es variiert, wie umfangreich der verwendete Standard ist und welche individuellen Anpassungen die EVU vornehmen.

9. Ziel der Untersuchung, wesentliche Befunde und weiterführende Fragestellungen

Ziel der vorliegenden Arbeit war es, die Formen und Folgen von Softwaregestaltung in der Energiewirtschaft zu untersuchen. Die Sichtung der bisherigen Forschung hat gezeigt, dass es für den Arbeitsprozess der Softwaregestaltung und die Arbeit der Softwaregestaltenden noch kein passendes Kontrollkonzept gibt. Zudem sind die Folgen unterschiedlicher Formen von Softwaregestaltung für die Softwaregestaltenden und die Arbeit und Organisation der Anwendung bisher überwiegend vernachlässigt worden. Die Fallstudien stellen die Bedeutung der Softwaregestaltung für Software, Arbeit und Organisation in den EVU und in der Branche dar. Die sich aus der Untersuchung ergebenden zentralen Befunde knüpfen an Debatten zur Kontrolle von (IT-)Wissensarbeit und zur digitalen Transformation in der Arbeits- und Industriesoziologie an. Bevor das Schlusskapitel auf die beiden Debatten im Einzelnen eingeht, fasst der nächste Punkt zunächst noch einmal kurz zusammen, warum Softwaregestaltung für die Forschung relevant ist.

9.1. Softwaregestaltung: ein wenig erforschter Arbeitsprozess der Digitalisierung

Industriespezifische Software spielt in Firmen sämtlicher Branchen eine immer größere Rolle (Kapitel 4.1). In softwareintensiven Industrien, in denen Software nicht das Produkt ist, ermöglicht sie gesteigerte Effizienz, Qualität und Individualität von Produkten und Dienstleistungen.

Um die Möglichkeiten der Softwareentwicklung in industriespezifischen Anwendungsbereichen zu nutzen, ist Softwaregestaltung zentral. Das ist jener Teil der Entwicklung, der erarbeitet, welche Anforderungen Programmierende zu programmieren haben. Mit Softwaregestaltung schaffen Organisationen die Voraussetzungen für industriespezifische Software, indem sie zwei Kernprobleme lösen (Kapitel 4.2):

- Erstens das Problem der **softwaretechnischen Interdisziplinarität**, weil im Zuge der Softwaregestaltung die Beteiligten die softwaretechnischen Möglichkeiten mit den Bedarfen einer Branche bzw. eines branchenspezifischen Anwendungsbereiches abstimmen müssen.
- Zweitens das Problem der **softwaretechnischen Gestaltungsmöglichkeiten**, indem a) Organisationen sich entweder auf die Softwaregestaltung oder die Anwendung einer Standardsoftware ausrichten und b) die Softwaregestaltung den softwaretechnischen Zuschnitt (individuell/Standard) erarbeitet.

In manchen Fällen wählen Unternehmen eine industriespezifische Standardsoftware einer Softwarefirma. Andere Unternehmen entscheiden sich für den Primat der Softwareentwicklung, der für digitale Start-ups typisch ist. Sie arbeiten seit ihrer Gründung softwaretechnisch interdisziplinär und sind von Anfang an organisatorisch auf die Gestaltung einer individuellen Software ausgerichtet.

Damit Organisationen die Möglichkeiten der Softwaregestaltung nützen können, müssen sie deren Arbeit kontrollieren. Um die Kontrolle der Softwaregestaltung zu untersuchen, ist ein passendes Konzept notwendig, das die Eigenarten der Softwaregestaltungsarbeit berücksichtigt. Was zeichnet die Softwaregestaltung als Arbeit aus? Sie basiert auf Wissen und Kommunikation und ist auf Kooperation angewiesen (5.1). Das liegt daran, dass ihre materielle Basis Software ist. Software ist eine zeichenbasierte Technologie, die aus mehreren technischen Schichten wie dem Quellcode oder der Anwendungsoberfläche besteht und für deren Entwicklung sprachliche Strukturierungsmittel wie Programmiersprachen oder Begriffe wie Architektur oder Modell genutzt werden. Zudem ändert sich Software, ist beliebig erweiterbar und kann hochkompliziert werden. Entsprechend ändert sich das Wissen über sie, kann stetig anwachsen und sehr umfangreich sein. Es ist Teil der Softwaregestaltung, sich auf eine sich verändernde, komplexer werdende Software einzulassen und sich gleichzeitig tiefergehend mit branchenspezifischen Themen auseinanderzusetzen. Dabei sind die Perspektiven unterschiedlicher Spezialist:innen zu berücksichtigen, weswegen nicht nur Wissen, sondern auch Kommunikation und Kooperation zentral sind. Die Beteiligten müssen sich auf eine Gestaltung einigen, auch wenn sie alle unterschiedliches Wissen und ihre eigene Perspektive auf die Software haben, die Biografie der Software besser oder schlechter kennen und sie nicht (alle) Teil der anwendenden Organisation sind.

Die Geschichte der Softwareentwicklung hat noch kein allgemeines Konzept zur Analyse der Softwaregestaltung hervorgebracht, das auf die untersuchten Fallstudien passen würde. Dabei ist die Softwaregestaltung in den letzten Jahrzehnten immer wichtiger geworden (Kapitel 5.2). Es gibt ein eigenes Forschungsgebiet zur Phase der Anforderungserarbeitung der Softwareentwicklung, wonach Kooperation, Kommunikation und soziale Kompetenzen wichtig sind, um eine gemeinsame Sprache und Übersetzungsfähigkeit zwischen IT- und Branchenfachleuten herzustellen (vgl. Alvarez 2002, Ross/Chiasson 2011, Kaminski 2012, Alsanoosy et al. 2020).

Die Forschung vor allem aus der Arbeits- und Techniksoziologie zeigt, dass die Organisationen in der Realität unterschiedliche Methoden (z.B. Scrum oder Projektarbeit) oder Kontrollformen (permissive oder direkte Kontrolle) für die Softwareentwicklung nutzen (vgl. Feuerstein 2012, Boes et al. 2018, Schulz-Schaeffer/Bottel 2018). Es gibt den

Versuch, mithilfe abstrakter Variablen wie intellektuellem und sozialem Kapital eine allgemeine Theorie der Softwareentwicklung zu schreiben (vgl. Wohlin et al. 2015). Allerdings vernachlässigt solch ein Ansatz strukturelle Faktoren wie eine unterschiedliche Arbeitsteilung zwischen Anwendung und Entwicklung (ob zwischen IT- und Fachabteilungen, EVU und IT-Dienstleistungsunternehmen) oder Konflikte zwischen beteiligten Gruppen wie Management und Softwareentwickelnden. Ganz zu schweigen davon, dass die in diesem Absatz zitierte Forschung nicht zwischen industriespezifischer und anderer Softwareentwicklung unterscheidet und sich nicht auf die Phase der Softwaregestaltung fokussiert.

Um geeignete, allgemeine Konzepte zur industriespezifischen Softwaregestaltung zu entwickeln, hat die vorliegende Untersuchung die Softwareentwicklung in der Energiewirtschaft erforscht und entsprechende Literatur ausgewertet. Durch Branchenanalyse und Fallstudien ist eine Betrachtung der Dynamik zwischen Individual- und Standardsoftwaregestaltung in der Branche möglich. Die Auswertung der Daten mit der Grounded Theory erlaubt es, den konkreten Arbeitsprozess der Softwaregestaltung zu analysieren und wesentliche Kategorien, entscheidende Kontextfaktoren sowie ihre Folgen herauszuarbeiten. Die Datengrundlage dafür ist die Befragung diverser Beschäftigtengruppen, deren jeweilige Perspektiven detaillierte Beschreibungen von Arbeit und Organisation der Softwaregestaltung liefern. Die gewonnenen Kategorien waren Ausgangspunkt für das Konzept der soziotechnischen Netzwerkarbeit und ergeben den Analyserahmen (8.1.3), der die sieben Fallstudien und deren Vergleich strukturiert. Er hilft, die Unterschiede herauszuarbeiten und Idealtypen zu konstruieren.

9.2. Erster Debattenbeitrag: Softwaregestaltung als soziotechnische Netzwerkarbeit

Wie kann eine konzeptionelle Beschreibung des Arbeitsprozesses der Softwaregestaltung und der Arbeit der Softwaregestaltenden aussehen, die sowohl die unterschiedlichen Konstellationen berücksichtigt, in denen sie stattfindet, als auch die Eigenheiten von kooperativer, kommunikationsintensiver und interdisziplinärer Wissensarbeit? Die Antwort auf diese Frage ist das Konzept der soziotechnischen Netzwerkarbeit.

Das 6. Kapitel hat ein erstes Konzept der soziotechnischen Netzwerkarbeit aus existierender Forschungsliteratur erarbeitet. Es stellt die theoretische Lösung des Transformationsproblems für die Softwaregestaltung dar. Es ermöglicht die begriffliche Beschreibung digitaler Arbeitsprozesse, die über verschiedene Organisationen, Teams oder Abteilungen verteilt sind und bei denen Subjektivität, Wissen, Kommunikation, Kooperation und Software im Mittelpunkt stehen (6.2). Das Besondere an dieser Kontrolle von Wissensarbeit ist ihr Netzwerkcharakter. Um zwischen Anwendung und Programmierung zu vermitteln, sind Subjekt, Betrieb und Organisationsnetzwerk gleichermaßen relevant. Weder die (Software-)Technik noch das Management sind dominierende Faktoren bei der Kontrolle der Softwaregestaltung. Das 6. Kapitel hat noch vereinfacht zwischen vier Ebenen unterschieden. Sie zeigen sich exemplarisch bei IT-Projekten wie ERP-Implementationen (6.3) und wirken zusammen, um die Transformation von Arbeitskraft zu gewährleisten (6.4). Aus Mangel an spezifischer Literatur

zur Softwaregestaltung hat das 6. Kapitel mehr allgemeine Literatur zu Wissensarbeit, Arbeit in Netzwerken und mit Software und IT herangezogen, um das Konzept der soziotechnischen Netzwerkarbeit zu entwickeln.

Das Empirie-Kapitel entwickelt das Konzept der **soziotechnischen Netzwerkarbeit** weiter. Es besteht dort aus dem **Arbeitsprozess der Softwaregestaltung** und der **Arbeit der Softwaregestaltenden**, wobei es auch die dazugehörigen Kategorien differenzierter herausarbeitet. Entsprechend grenzt der aus der Empirie entwickelte Analyserahmen klar ab: zwischen soziotechnischer Konstellation, Arbeitsprozess und Arbeit der Softwaregestaltenden.

So ist es dann auch möglich, die **typischen Unterschiede** in den Fallstudien bei der soziotechnischen Netzwerkarbeit zu konstruieren: Entweder liegt ein **dezentraler** oder **zentraler** Arbeitsprozess vor, arbeiten die Softwaregestaltenden in einer **Matrixorganisation** oder einer **reinen Netzwerkorganisation**. Alte Kontrollunterschiede wie direkte und permissive Kontrolle (vgl. Friedman 1977) oder industrialisiert und nicht-industrialisiert (vgl. Boes et al. 2018) passen hingegen nicht zur soziotechnischen Netzwerkarbeit.

Zudem zeigt sich die Kontingenz der Softwaregestaltung in Bezug auf die **soziotechnische Konstellation** (ausführlicher unter 8.6.2.1). Sie ist die Ausgangsbedingung für die Softwaregestaltung. Zugleich stellt sie auch deren Grenzen dar, in denen sich die Möglichkeiten der Softwaretechnik verwirklichen lassen. Je Fallstudie ist es vom **Anwendungsbereich** abhängig, welche Möglichkeiten für die Softwaregestaltung bestehen. Das betrifft den Anteil der Datenverarbeitung im Anwendungsbereich und ob sie einen gesamten Prozess gestalten kann oder nur Teile davon. Aus Sicht der Softwaregestaltung wird die **Arbeitsteilung** zwischen Anwendung und Programmierung zur Wissensgrenze, die überwunden werden muss. Sie kann innerhalb oder zwischen Organisationen bestehen. Als primäre Formen der **Grundkoordination** zeigen sich in den Fällen Markt, Hierarchie oder Netzwerk. Egal welche (Mischung der) Grundkoordination vorliegt, müssen die Beteiligten immer einen Weg finden, um für die Softwaregestaltung zusammenzuarbeiten. Um von einer Netzwerkkoordination zu sprechen, muss die Zusammenarbeit ungehindert von hierarchisch-formalen Wegen, vertraglichen Einschränkungen oder Marktkalkülen zwischen Anwendung und Programmierung möglich sein (wie bei den Fallstudien STARTUP und KOOP3). Die **Softwarearchitektur** hat sich in allen Fällen als prägend für die Softwaregestaltung erwiesen: Erstens entscheidet ihre Aufteilung z.B. in Module darüber, wie sich die Arbeit innerhalb und zwischen den Organisationen verteilt (vor allem für die Programmierenden). Zweitens prägt die Softwarearchitektur in der Mehrzahl der Fälle durch ihren Zuschnitt (individuell, Standard) und die daraus resultierenden Abhängigkeiten den Kommunikationsaufwand und welche Kommunikationswege zwischen Anwendung und Entwicklung existieren.

Dieser Abschnitt stellt zunächst die aus den Unterschieden zwischen den Fallstudien erarbeiteten Typen der soziotechnischen Netzwerkarbeit vor (8.6.1.1), um dann genauer auf die einzelnen Kategorien einzugehen (8.6.2) und abschließend den Beitrag zur Debatte über die Kontrolle von Wissensarbeit in der Arbeitssoziologie zu vertiefen.

9.2.1. Typische Unterschiede in der soziotechnischen Netzwerkarbeit

Die Unterschiede zwischen den Fallstudien im Arbeitsprozess der Softwaregestaltung lassen sich zu Typen zusammenfassen. Wobei es sich um netzwerkspezifische Unterschiede in der Kontrolle des Arbeitsprozesses und der Arbeit der Softwaregestaltenden handelt:

1. Der Arbeitsprozess der Softwaregestaltung findet entweder dezentral statt, z.B. in einem Team für eine individuelle Softwaregestaltung oder zentralisiert in einer Organisation wie einer Softwarefirma, die ein Standardprodukt gestaltet (8.3.1).
2. Die Softwaregestaltenden arbeiten in einer Matrixorganisation, in der sie für ihre Arbeit Hindernisse überwinden müssen, oder in einer reinen Netzwerkorganisation, die ihnen ihre Arbeit einfacher macht, weil dort weder Hierarchien noch Marktbeziehungen die Zusammenarbeit zwischen Anwendung und Programmierung stören (8.4.1).

Die vier Grundtypen der soziotechnischen Netzwerkarbeit sind von der soziotechnischen Konstellation abhängig, die sich aus den typischen Unterschieden der soziotechnischen Netzwerkarbeit ergeben (8.6.1.1): KOORDINIEREND (zentral in einer Matrixorganisation), DIREKT (dezentral in einer reinen Netzwerkorganisation), ERGÄNZEND (dezentral in einer Matrixorganisation) und VIRTUELL (zentral in einer reinen Netzwerkorganisation).

Der Typ **KOORDINIEREND** ist ein **zentralisierter** Arbeitsprozess der Softwaregestaltung in einer **Matrixorganisation**. Wie für einen zentralen Arbeitsprozess typisch, konzentrieren sich **Rollen** wie Anforderungsmanagende auf die Koordination der Anforderungsaufnahme in unterschiedlichen Organisationen. Es existiert ein **Ablauf** in Form eines Anforderungsmanagements, in dem die Beteiligten über Anforderungen verhandeln und Konflikte lösen. **Software**lösungen wie Ticketsysteme dienen dazu, Transparenz zwischen den Organisationen herzustellen und Abstimmungen zu organisieren. Die **kommunikativen Beziehungen** müssen die Spannungen zwischen den unterschiedlichen Interessen ausgleichen können und die teils bürokratischen Abläufe ergänzen, z.B. indem auf persönlicher Ebene ein direkter Austausch stattfindet. Indem softwaregestaltende Organisationen wie IT-DL auf die Softwaregestaltung spezialisiert sind, können sie ein **Beschäftigungssystem** etablieren, das für eine entsprechende Auslastung sorgt. Sie können z.B. den Wünschen der Softwaregestaltenden nachkommen und ihnen Karrieren unabhängig von Hierarchien ermöglichen (z.B. eine Karriere als Fachexpert:in ohne personelle Verantwortung zu machen und auf diesem Wege in eine höhere Vergütungsgruppe zu kommen). Aufgrund der Marktbeziehung zwischen z.B. IT-DL und EVU **kontrollieren** Letztere die Arbeit der Softwaregestaltenden des IT-DL. In Hierarchien sind es die Führungskräfte, welche die Arbeit der Softwaregestaltenden kontrollieren. Zudem führt die Matrixorganisation aufgrund der **Wissensverteilung** zu einer Abhängigkeit, weil eine stetige softwaretechnische Interdisziplinarität fehlt: Einerseits fehlt den EVU bzw. einzelnen Abteilungen das Wissen, um selbst die Möglichkeiten der Softwaregestaltung einschätzen zu können. Andererseits fehlt der programmierenden Abteilung oder Organisation Wissen über die Anwendungsbereiche.

KOOP1, KOOP2 und KOOP2 sind nur zum Teil Beispiele für diesen Typ, weil einzelne Kategorien anders ausgeprägt sind.

Charakteristisch für den Typ **DIREKT** ist ein **dezentraler** Arbeitsprozess der Softwaregestaltung in einer **reinen Netzwerkorganisation**. Typische Merkmale einer dezentralen Softwaregestaltung sind **Rollen**, die direkt Anforderungen von Anwendenden aufnehmen. Die Rollen sind Teil eines **Ablaufs**, der unterschiedliche Methoden wie Scrum, Resonanzgruppen und Workshops nutzt, um Anforderungen direkt aufzunehmen, auszuarbeiten und den Programmierenden übergeben zu können. Die **kommunikativen Beziehungen** sind offen, direkt, langfristig und tief verankert im Anwendungsbereich, nicht getrennt durch Abteilungsgrenzen und Hierarchien und bei Bedarf mit engem Kontakt zu den Anwendenden. Die verwendeten **Werkzeuge** wie Ticketsystem und Chat-Software erlauben es Beteiligten, dezentral Anforderungen ohne bürokratische Hindernisse (wie z.B. fehlende Berechtigungen oder eine formal vorgegebene Abstimmung mit der Führungskraft) aufzunehmen. Ein typisches Element der Arbeit in einem reinen Netzwerk ist ein **Beschäftigungssystem**, das es den Mitarbeitenden erlaubt, in der rollenbasierten Organisation interessante und für Softwaregestaltende adäquate Aufgaben zu bekommen. Es geht nicht darum, in einer Hierarchie aufzusteigen (die es formal auch gar nicht gibt). Es ist eine Organisation, die auf die Softwaregestaltung ausgerichtet ist. Weil dem so ist, findet z.B. die Softwaregestaltung nicht nur temporär statt. Die **Kontrolle** passiert im Wesentlichen horizontal durch die Kollegenschaft und ermöglicht Eigenmotivation, individuelle Steuerung der Arbeitsbelastung und selbstständiges Arbeiten (z.B. wie es Scrum vorsieht). Durch den kontinuierlichen, iterativen Austausch im Netzwerk für die Softwaregestaltung ist eine gemeinsame **Wissensbasis** gesichert. Weil das Wissen nicht hierarchisch oder über einen Markt verteilt ist, bestehen keine Hindernisse, um an es heranzukommen.

Der Typ **ERGÄNZEND** kombiniert einen dezentralen Arbeitsprozess mit einer Matrixorganisation. Bei **VIRTUELLL** existiert ein zentraler Arbeitsprozess der Softwaregestaltung in einer reinen Netzwerkorganisation. Die beiden Typen sind hier nicht ausführlicher vorgestellt, weil sich die bei den beiden Typen oben ausgeführten idealtypischen Eigenschaften wiederholen – nur eben anders kombiniert. Zudem ist keine der Fallstudien vom Typ VIRTUELL und es wäre noch eine weitere Untersuchung mit Fallstudien zu dieser Form industriespezifischer Softwaregestaltung notwendig.

Die vier Idealtypen unterstreichen nochmal, dass es abhängig von der Konstellation ist, welche Rolle Management und Führungskräfte bei der Softwaregestaltung spielen. So haben sie in der Aufbauorganisation einer Matrixorganisation ihre feste Position mit entsprechenden Entscheidungsbefugnissen, Ressourcen und den Fokus auf die Ergebniskontrolle. In den Fallstudien zeigt sich allerdings, dass sie auch dort mal mehr und mal weniger an der Softwaregestaltung teilnehmen.

9.2.2. Gemeinsame Kategorien der soziotechnischen Netzwerkarbeit

Die soziotechnische Netzwerkarbeit besteht aus dem Arbeitsprozess der Softwaregestaltung und der Arbeit der Softwaregestaltenden. Für beide konnte die Analyse der Fallstudien die zentralen Kategorien herausarbeiten.

9.2.2.1. Arbeitsprozess der Softwaregestaltung

Die Transformation der Arbeitskraft im Arbeitsprozess der Softwaregestaltung basiert auf den Rollen, einem Ablauf, kommunikativen Beziehungen, digitalen Werkzeugen und dem softwaretechnischen Zuschnitt.

Das Konzept der **Rolle** hat sich als nützlich erwiesen, dem Netzwerkcharakter der Kontrolle von Softwaregestaltungsarbeit gerecht zu werden. Weil Rollen das Handeln durch Erwartungen leiten und nicht die konkrete Arbeitsausführung festlegen, erlauben sie einen eigenständigen, situativ passenden Beitrag der Einzelnen und dass diese sich je nach Konstellation der Arbeitsteilung zwischen Anwendung und Programmierung einfügen. Es gibt zwar feste Rollen wie Product Owner:in oder IT-Projektleitung, denen in einigen Fällen Schulungen zugrunde liegen. In der Praxis passen die Beschäftigten ihre Rolle in Bezug zu den anderen Beteiligten aber an. Für viele, die bei der Softwaregestaltung mitmachen, gehört es zum Arbeitsalltag, mehrere und wechselnde Rollen einzunehmen. Zuletzt sind die Erwartungen nicht abhängig von Hierarchien, weshalb sie von diesen unabhängig das Handeln leiten können. Neben den Erwartungen an spezifische Rolle wie z.B. IT-Projektleitung kontrollieren allgemeine Erwartungen die Arbeit im Arbeitsprozess, wie das 6. Kapitel gezeigt hat: kooperativ sein, selbstorganisiert arbeiten, aktiv mit Softwareobjekten interagieren und sich ausgehend von ihnen zu koordinieren, mit Nicht-Wissen umgehen können und sich im Netzwerk bewegen. Subjektivierung (vgl. Minssen 2011) und subjektivierendes Arbeitshandeln (vgl. Böhle 2010, Bolte 2017a, Bolte 2017b, Weishaupt/Hösl 2017) sind Kernbestandteile der Kontrolle der Arbeitskraft durch solche Rollen.

Der Netzwerkcharakter des **Ablaufs** zeigt sich allgemein bei IT-Projekten, die Organisationen in ganz unterschiedlichen Kontexten einsetzen. Dabei ist der Grad der Formalisierung sehr unterschiedlich (vgl. Heidling 2018: 224) und nicht entscheidend für die Kontrolle von Arbeit. Vielmehr ist wichtig, dass selbst bei stärkerer Formalisierung Handlungs- und Entscheidungsspielräume für einzelne Projektmitarbeitende bestehen (vgl. Kalkowski/Mickler 2005) und während und nach ERP-Implementierungen ein abteilungsübergreifendes Gestaltungsnetzwerk vorhanden ist, auf dem der Ablauf basiert (vgl. Hohlmann 2007: 353). In den Fallstudien trägt der Ablauf zur Netzwerkarbeit bei, indem er für die Kooperation (ob temporär oder langfristig) und die Kommunikation sorgt und beides nicht z.B. ökonomische Kalküle oder einzelne Führungskräfte untergräbt. Das zeigt sich an drei wesentlichen Aspekten: A) Der Ablauf schafft Möglichkeiten für Feedback(-Schleifen) zwischen Softwareanwendung, -gestaltung und -programmierung durch den Einsatz diverser Methoden – ob durch Treffen, E-Mails, Tests, Chat-Programme, persönliche Gespräche oder Methoden wie Prototyping oder Resonanzgruppen. B) Er bindet situativ die im organisationalen Netzwerk verteilten Beteiligten z.B. über Workshops ein. C) Er pflegt die Beziehungen durch regelmäßige Treffen, Mechanismen zum Abgleich von Erwartungen und zur Lösung von Konflikten. In den Fallstudien passiert dies z.B. durch Strategie-Treffen zwischen IT-Dienstleistungsunternehmen (IT-DL) und EVU, Anforderungsrunden mehrerer Fachbereiche eines EVU, durch Arbeitskreise einer Softwarefirma mit ihrer Kundschaft (EVU) oder Projektlenkungskreise. Grundlage für den Ablauf können unterschiedliche Methoden wie Scrum, Projektarbeit oder Anforderungsmanagement sein.

Neben einem Ablauf zeichnet sich die soziotechnische Netzwerkarbeit der Softwaregestaltung durch **kommunikative Beziehungen** zwischen Organisation(seinheiten) und Einzelpersonen aus. Wie schon die Forschung festgestellt hat, reichen für eine kooperative Zusammenarbeit z.B. Projektstrukturen allein nicht aus (vgl. Rüegg-Stürm/Young 2001) und informelle Strukturen der Kooperation sind wichtig (vgl. Bolte/Porschen 2007). Zudem helfen Beziehungen, die längerfristig sind, weil Vertrauen erst mit der Zeit aus reziproken Beziehungen und durch geteilte Interessen entsteht (vgl. Powell 1990, Uzzi 1997).

In den Fallstudien wird deutlich, welchen Beitrag interpersonale, kommunikative Beziehungen zur Netzwerkarbeit leisten und dass hierfür technische Lösungen wie Ticketsysteme allein nicht ausreichen. Durch kommunikative Beziehungen, die partnerschaftlich, kooperativ, auf Augenhöhe, offen oder von einem Geben und Nehmen geprägt sind, können Softwaregestaltende reine Marktbeziehungen und Barrieren innerhalb von Organisationen (seien es Hierarchien oder Abteilungs- und Teamgrenzen) überwinden. Weil es bei der Softwaregestaltung darum geht, ein gemeinsames Verständnis über energiewirtschaftliche Bedarfe und softwaretechnische Möglichkeiten zu sichern, ist die Kommunikationskompetenz wichtig. Der Mediator in einer Fallstudie zeigt die Verschränkung von Beziehungs- und Kommunikationskompetenz exemplarisch: Er arbeitet an Beziehungen, z.B. indem er Konflikte löst, sorgt für den kommunikativen Austausch und hat auch noch das fachliche Vokabular und Wissen, um sich verständlich machen zu können und für ein gemeinsames Verständnis unter den Beteiligten zu sorgen. Rollen wie Scrum Master:innen kümmern sich ebenso um die zwischenmenschlichen Beziehungen. Bei einem IT-DL ist die Rolle Key Account Management dafür zuständig, die Beziehung zu den EVU zu pflegen und bei Missverständnissen die inhaltliche Klärung herbeizuführen.

Neben diesen interpersonalen Beziehungen spielen auch jene zwischen Organisationen und Organisationseinheiten wie Abteilungen in den Fallstudien eine Rolle. Diese Beziehungen pflegt, wie oben beschrieben, auch der Ablauf auf unterschiedliche Weise. Darüber hinaus basiert die erfolgreiche längerfristige Zusammenarbeit zwischen IT-DL und EVU bei KOOP1 auf einem langjährigen Lernprozess. In den Arbeitsprozessen der Softwaregestaltung der Fallstudien sind die in der Forschung beschriebenen Kooperationshemmnisse oder -hindernisse (siehe 6.4.1) zwar Thema. Sie führen aber nicht zum Abbruch des Arbeitsprozesses, weil grundsätzlich die Kooperationsbereitschaft bestehen bleibt, auch wenn z.B. EVU überlegen, das IT-DL zu wechseln. Insofern ist der Forschung recht zu geben, dass für eine Zusammenarbeit kooperative Beziehungen bestehen müssen. Die Fallstudien zeigen, dass dies für ganz unterschiedliche Konstellationen von Arbeitsteilung und Grundkoordination zwischen Anwendung und Entwicklung gilt.

Neben Ablauf und Beziehungen basiert die Kontrolle der Arbeit im Arbeitsprozess der Softwaregestaltung auf der Software. Sie ist Arbeitsgegenstand und -mittel. Wie die Literatur zeigt (siehe 6.4.2), wirkt sie nicht nur einschränkend, sondern auch ermöglichend. Die zu gestaltende Software koordiniert als gemeinsam gestaltbares, zeichenbasiertes Bezugsobjekt die Wissensarbeit (vgl. Nicolini/Mengis/Swan 2012, vgl. Barrett/Oborn 2010, Carugati et al. 2018, Ponte/Rossi/Zamarian 2009, Bolici/Howison/Crowston 2009 und 2016). Zudem verlangen sie und die verwendeten Software-Werkzeuge wie Ti-

cketsysteme von den Anwendenden wissensevozierendes Engagement und Interaktion (vgl. Darr 2019, Rennstam 2012). Software schränkt die Softwaregestaltung ein, indem sie Eingabemöglichkeiten und Abläufe vorgibt (vgl. Kleemann/Matuschek 2008), digital in einen Prozess integriert (vgl. Sauer 2018) und soziale Strukturen wie Rollen oder Routinen softwaretechnisch stützt (vgl. Mutch 2010, Volkoff et al. 2007). Sie macht Arbeit transparent und bietet Möglichkeiten der Überwachung (vgl. Zuboff 1988). In der Softwareentwicklung geben die softwarebasierten Werkzeuge keinen (software)maschinellen Takt vor, sondern menschliche Intervention macht den digital-basierten Arbeitsprozess aus (vgl. Andrews et al. 2005, Barrett 2005).

In den Fallstudien ermöglichen Softwarelösungen erstens als **digitale Werkzeuge** der Softwaregestaltung die verteilte Teilnahme durch die Ein- und Ausgabe von Daten: ob Ticketsysteme, Entwicklungs- und Testumgebungen von Softwarefirmen wie SAP, Chat-Gruppen oder (geteilte) Excel-Dateien. Zudem stellen die Software und die dahinterliegenden Datenbanken den zentralen Wissensspeicher für die in der Softwaregestaltung arbeitenden Personen dar. Ticketsysteme lassen es zu, dass Anwendende oder andere Beteiligte aus unterschiedlichen Organisationen und Abteilungen Anforderungen aufnehmen, wodurch sie zugleich die Softwaregestaltung dokumentieren und sie für andere nachvollziehbar machen. Im Quellcode selbst ist die Umsetzung dokumentiert und manchmal existieren auch noch separate, softwarebasierte Dokumentationswerkzeuge wie Wikis. Zweitens ist die Kontrollwirkung der softwarebasierten Werkzeuge abhängig von der soziotechnischen Konstellation. In Hierarchien sorgt Software für Transparenz (über Anforderungen, deren Bearbeitungsstatus etc.) für die Führungskraft, in Marktbeziehungen für die Kundschaft und in reinen Netzwerkorganisationen zwischen den gleichgestellten Peers. Auch der zeitliche Takt in der Softwaregestaltung ist konstellationsabhängig: Die Reaktionszeiten werden entweder durch die zwischen IT-DL und EVU vereinbarten SLA, durch Aufwandsschätzungen von Fachleuten wie IT-Beratenden, durch die von der Regulierung vorgegebenen Fristen oder durch die Dringlichkeit von Ad-hoc-Fehlerbehebungen bestimmt. Letztendlich nutzen die Organisationen der Fallstudien die Software weniger zur direkten Kontrolle von Arbeit. So ist es den Arbeitenden möglich, sich auf komplizierte Probleme einzulassen, sich mit der Software und fachlichen Fragen tiefergehend zu beschäftigen und schwierigere Texte bzw. Anforderungen und Software zu schreiben.

Die für Netzwerkarbeit typische verteilte Teilnahme hängt von der Softwarearchitektur ab und es ist Teil des Arbeitsprozesses der Softwaregestaltung, den **softwaretechnischen Zuschnitt** der Software und damit einen Teil der Softwarearchitektur zu verwaltet. In der Fallstudie KOOP1 entscheidet das Anforderungsmanagement für einzelne Anforderungen, ob das IT-DL sie als Teil des Standards umsetzt. Bei INTERN2 priorisiert die Anforderungsrunde, was das EVU individuell programmiert. Dabei zeigen die Fallstudien, dass die Architektur meist zentral vorgegeben ist (z.B. durch eine Softwarefirma). Zudem müssen Organisationen fähig sein, Synergien bei der Softwaregestaltung zu erkennen und Abstimmungsprozesse zu etablieren, um darüber zu verhandeln, was Teil einer Standardsoftware wird. So drückt sich in der jeweiligen Softwarearchitektur in den Fallstudien der Netzwerkcharakter der Arbeit aus: Mal lässt die Architektur eine dezentrale Gestaltung zu, mal nur eine zentrale.

9.2.2.2. Arbeit der Softwaregestaltenden

Zur soziotechnischen Netzwerkarbeit gehört auch, wie die Softwaregestaltenden arbeiten, die zwischen Anwendung und Programmierung tätig sind (6.4.3 und 8.4). Die Fallstudien zeigen, dass hinter der Softwaregestaltung eine eigenständige Gruppe von Beschäftigten steht und für die Transformation ihrer Arbeitskraft das Beschäftigungssystem, ihre individuelle Kontrolle und die Wissensverteilung entscheidend sind.

Die Führungskräfte **kontrollieren** die Softwaregestaltenden vor allem anhand ihrer Ergebnisse und nur bei Auffälligkeiten. Führungskräfte kümmern sich um den Rahmen: Sie legen je nach Fall das Budget fest, stellen Mitarbeitende ein, initiieren Projekte oder klären Konflikte innerhalb (z.B. mit anderen Führungskräften oder Teams) oder zu anderen Organisationen (Softwarefirma, IT-DL, EVU). Die Arbeitsbelastung der Softwaregestaltenden ist sehr unterschiedlich. Manche Befragte beschreiben sie als verhandelbar. Viele sprechen davon, dass sie intrinsisch motiviert sind. Wenn formale Hierarchien vorhanden sind, stehen sie über den Anwendenden. Dabei prägt die Grundkoordination die Kontrolle ihrer Arbeit: Von ihr hängt ab, ob mehr das Management (Hierarchie), die Kundschaft (Markt) oder die Kollegenschaft (Netzwerk) kontrollieren.

Softwaregestaltende können in puncto **Beschäftigungssystem** für ein IT-DL oder eine Softwarefirma arbeiten und für mehrere EVU tätig sein. Sie haben ein geringeres Interesse an einer disziplinarischen Karriere. Wenn EVU anfangen, Software zu gestalten, gibt es teilweise neue zusätzliche Karrierewege für Softwaregestaltende (»Kompetenzkarriere«). Die herkömmlichen Karrieremuster als Aufstieg auf einer Leiter in einer Hierarchie mit disziplinarischer Verantwortung gibt es weiterhin. Im Vergleich zu Anwendenden haben sie mehr (Arbeits-)Marktmacht und eigene Arbeitsmärkte. Die Organisationen setzen sie flexibel ein – ob innerhalb einer Matrixorganisation oder in anderen Organisationen. Das machen sie vor allem dann, wenn sie spezifisches Methodenwissen haben (z.B. Scrum) oder tiefere Kenntnisse einer auch in anderen Branchen genutzten Softwareumgebung (z.B. SAP). Es zeigt sich, was bereits im 6. Kapitel die Forschung feststellt: Sie erhalten Einfluss und ihre Position in den Organisationen, weil sie für diese die Software-Technologie nutzbar machen (vgl. Armstrong 1985, Hohlmann 2007).

Bei der **Wissensverteilung** zeigen die Fallstudien, dass die Softwaregestaltenden über sehr unterschiedliche interdisziplinäre Wissensstände verfügen. Das Spektrum reicht von Softwaregestaltenden, die sich vor allem mit der Koordination der Softwaregestaltung beschäftigen und nur noch wenig inhaltliches Wissen benötigen, bis hin zu IT-Beratenden, die nicht nur Anforderungen aufnehmen, sondern auch gleich noch umsetzen. Allgemein kann bei der Softwaregestaltung von einer interdisziplinären Praxisgemeinschaft gesprochen werden. Die Praxisgemeinschaft zeigt sich zum einen dadurch, dass jemand, der davon ausgeschlossen ist, nicht mehr Softwaregestaltung betreiben kann und von anderen abhängt, weil ihm das Wissen fehlt. Zum anderen zeigt sie sich dadurch, dass sich die beteiligten Personen in kontinuierlichem Austausch befinden, um Software zu gestalten. So entstehen unterschiedliche Praxisbiografien und damit unterschiedliche Lernbiografien für jeden einzelnen Beschäftigten. Hohlmann (2007) hat von einem Gestaltungsnetzwerk gesprochen, welches das interdisziplinäre Wissen hat, um die SAP-Standardsoftware über das Implementierungsprojekt hinaus zu gestalten. In Anlehnung an Wengers *Community of Practice* (1999) kann diese als

Community of Practice and Software Objects bezeichnet werden. Denn für das, was die Softwaregestaltenden wissen, sind weniger Schulungen entscheidend und vielmehr die Praxis mit anderen und der Software. Teil der Wissensverteilung der Softwaregestaltung sind die verwendeten, softwarebasierten Werkzeuge und die Software selbst. In ihnen materialisiert sich das Wissen (z.B. Quellcode, Ticketsystem, E-Mails, Dokumentationen von Umsetzungen und Funktionalitäten). Nicht nur Personen sind Träger des Wissens, sondern auch die Software mit dem entsprechend hinterlegten Wissen. Ob aus digitalen Dokumenten oder der gestalteten Software selbst: Das Lernen ergibt sich situativ in einer Praxis (Learning by Doing ist für die Befragten zentral), in der Software(-Objekte) eine wichtige Rolle spielt(en). In vielen Fallstudien sind Führungskräfte nicht Teil der Softwaregestaltung und haben entsprechend wenig Wissen über sie oder Softwareentwicklung im Allgemeinen.

9.2.3. Beitrag zur Debatte über die Kontrolle von Wissensarbeit

Das oben dargestellte Konzept der soziotechnischen Netzwerkarbeit stellt eine Ergänzung zur Debatte über die Kontrolle von Wissensarbeit in zweifacher Hinsicht dar: Erstens, weil ein netzwerkförmiger Arbeitsprozess die zentrale Ebene der Kontrolle von Arbeit ist. Zweitens, weil die Kontrolle allgemeine Netzwerkcharakteristika aufweist und es netzwerktypische Unterschiede in der Kontrolle gibt und z.B. keine Unterschiede wie jene zwischen permissiver und direkter Kontrolle (vgl. Friedman: 1977). Rennstam spricht von Wissens- statt Verhaltenskontrolle (vgl. Rennstam 2012: 1072). Dem ist zuzustimmen. Wobei es bei der Softwaregestaltung nicht nur um eine Kontrolle des Wissens, sondern der Kommunikation über letztlich individuelle und konstellationsabhängige Anforderungen geht. Der Standardisierung und Formalisierung von Softwaregestaltung und ihrer Koordination über Märkte und Hierarchien sind daher Grenzen gesetzt.

Der Netzwerkcharakter zeichnet sich erstens dadurch aus, dass nicht eine der Ebenen wie Subjekt, Betrieb oder Organisationsnetzwerk zentral für die Transformation der Arbeitskraft ist, sondern die Ebenen zusammenwirken, um zwischen Anwendung und Programmierung Software zu gestalten. Ein Teil der arbeitssoziologischen Forschung rückt den Beitrag der Arbeitenden selbst in den Vordergrund, wenn es um die Kontrolle von Arbeit geht.

> »Vor dem Hintergrund erweiterter Selbstorganisationspotentiale sorgen die Beschäftigten selbst für die effiziente Transformation ihrer eigenen Arbeitskraft in reale Arbeitsleistung« (Marrs 2010: 342).

Allgemein fußen neue Managementkonzepte mehr auf Autonomiespielräumen und weniger auf Fremdkontrolle, wie es beim Taylorismus der Fall ist, der davon ausgeht, dass Beschäftigte nicht gern und von sich aus effizient arbeiten, weshalb direkter Zwang und Kontrolle notwendig sind. Der Extremfall dieser subjektbasierten Transformation der Arbeitskraft ist jener des Arbeitskraftunternehmers von Voß und Pongratz (vgl. Marrs 2010: 339ff.). Die im 6. Kapitel zitierten Studien zu Produktmanagenden in Softwarefirmen nehmen schwerpunktmäßig eine einzelne Rolle in den Blick und wie diese Software gestaltet (vgl. Bolte 2017a, Bolte 2017b, Weishaupt/Hösl 2017). Andere Forschende sehen

für Fragen der Kontrolle von IT-Arbeit den einzelnen Betrieb (vgl. Boes et al. 2018) oder bei ausgelagerter IT-Arbeit die organisationsübergreifende Ebene als entscheidend an (vgl. Flecker/Holtgrewe 2008, Mezihorak 2018).

Im Gegensatz dazu zeichnet die soziotechnische Netzwerkarbeit aus, dass Ablauf, organisationale und interpersonale Beziehungen, Software und Softwaregestaltende alle dazu beitragen, zwischen Anwendung und Programmierung Kooperation und Kommunikation herzustellen, situativ Beschäftigte im Netzwerk einzubinden und allen Beteiligten Spielräume für Wissensaneignung, -verarbeitung und Kommunikation zu geben. Doch auch wenn das alle Arbeitsprozesse der Softwaregestaltung gemeinsam haben, ist der Schwerpunkt der Kontrolle des Arbeitsprozesses der Softwaregestaltung je soziotechnischer Konstellation anders und mal arbeiten nur Beschäftigte eines Betriebs und mal mehrere Organisationen zusammen. Damit sieht die vorliegende Untersuchung die Transformation der Arbeitskraft als etwas an, das auf mehreren Ebenen basiert, und folgt damit Sydow und Windeler (2000), Apitzsch (2006) oder Kalkowski/Mickler (2015), für die Netzwerke ebenfalls aus mehreren Ebenen und Dimensionen bestehen. Wobei diese Autor:innen die Ebene der Software vernachlässigen. Mit dem Konzept der soziotechnischen Netzwerkarbeit schließt die vorliegende Forschungsarbeit Forschungslücken sowohl bei der Verbindung von Arbeitsprozess, organisatorischer Ebene und unternehmensübergreifenden Beziehungen (vgl. Sydow/Helfen 2020: 225) als auch bei »Zusammenhängen zwischen praktischer Kooperation, Netzwerkformen, Nutzung von IuK-Techniken, Wissenstransfer und Arbeit« (Schmiede 2006: 466).

Neben der Debatte, welche Ebene zentral bei der Kontrolle von (IT-)Wissensarbeit ist, gibt es zweitens eine Diskussion darüber, welche Kontrollunterschiede von Arbeit existieren. Die im 5. Kapitel zitierte Forschung zur Softwareentwicklung sieht folgende Unterscheidungen: zwischen direkter und permissiver Kontrolle à la Friedman (1977) (vgl. Feuerstein 2012); zwischen Industrialisierung der Kopfarbeit inkl. digitalem Fließband und autonom agierenden Scrum-Teams (vgl. Boes et al. 2018).

Diesen Unterscheidungen folgt die vorliegende Untersuchung nicht. Sie stimmt vielmehr Autor:innen zu, die eine Gleichsetzung von Wissens- und Industriearbeit ablehnen. Klar grenzen z.B. Andrews et al. (2005) die Arbeit in der Softwareentwicklung von Fließbandarbeit ab, weil weder eingesetzte Methoden zur durchgehenden Standardisierung beitragen, noch die Abfolge der einzelnen Arbeitsschritte fest getaktet ist (vgl. ebd.: 63f.). Programmierende haben ihre Zeit selbst in der Hand und nur die Deadline ist direkt vom Management kontrolliert (vgl. Barrett 2005: 89). Auch in anderen stark digitalisierten Arbeitsbereichen ist die Rede von digitaler Fließbandarbeit irreführend. Zum Beispiel sind Microtasks bei Crowdwork nicht zeitlich in enge Prozessketten eingebunden und zeitliche Verzögerungen von wenigen Minuten führen anders als am Fließband nicht zur Störung des Gesamtprozesses. Im Vergleich zum klassischen Taylorismus ist im Fall des Crowdworking eine erweiterte Autonomie bei zumeist eng begrenzten Handlungsspielräumen vorhanden (vgl. Menz/Nies/Sauer 2019: 193f.).

Die Forschung zur Subjektivierung von Arbeit am Beispiel des Produktmanagements hat bereits herausgearbeitet (siehe 6.4.3.1), dass Softwaregestaltung ganz andere Anforderungen an die Kontrolle von Arbeit stellt. Softwaregestaltung lässt sich nicht im Vorhinein genau finanziell bewerten und standardisieren wie z.B. die Zubereitung eines Burgers, eine Massage, die Massenproduktion von Seife oder Hochleistungschips.

IT-Beratende oder IT-Projektleitende bei ERP-Implementierungen müssen komplexe Aufgaben bearbeiten und vorher unbestimmbare Informationsmengen sammeln, deuten und integrieren. Es wird Selbstmanagement erwartet (vgl. Bläsche/Lappe 2006: 307). Wissensarbeit ist an sich unbestimmt (vgl. Schmiede 2015: 51) und verlangt oftmals situativ zu improvisieren, weshalb sie sich von formal-rational durchorganisierten Prozessen unterscheidet (vgl. Stark 2017: 18f.). Der Typ der permissiven Kontrolle von Friedman ist notwendig, weil Engagement, Kooperation, Kreativität und Qualifikation der Beschäftigten »nicht durch direkte Kontrolle mobilisiert werden« (Marrs 2010: 336) können, wie es für Wissensarbeit typisch ist. Das Management muss die konkrete Arbeitsausführung den Beschäftigten überlassen (vgl. ebd.). Mehrere Autoren erachten Situativität und lokale Praktiken als typisch für Softwareentwicklung (vgl. Friedman/Cornford 1989: 358, Schulz-Schaeffer 1996, Wohlin/Smite/Moe 2015). Dabei ist IT keine reine Steuerungs- und Kontrolltechnik für die Wissensarbeitenden, sondern eine Infrastruktur, die Mitwirkung verlangt. Deshalb werden tayloristische und systemische Rationalisierungen für Organisationen, die IT nutzen, dysfunktional (vgl. Rock/Ulrich/Witt 1990: 43). Einige Autoren betonen die Dysfunktionalität einer Kontrolle von Softwareentwicklung, die zu sehr auf Hierarchien oder Märkte setzt (vgl. Gregory et al. 2013, vgl. Felin/Zenger/Tomsik 2009: 557, vgl. Brusoni/Prencipe/Pavitt 2001: 610). Eine andere Studie zur Softwareentwicklung hat festgestellt, dass es besser sein kann, die Werkzeuge den Kommunikationswegen anzupassen und nicht, wie am Fließband, die Arbeitenden einer Maschine unterzuordnen.

> »One direct implication is that software tools and programs should be designed to take advantage of the naturally occurring patterns of communication and work which spontaneously come about as a result of large scale programming« (Waterson/Clegg/Axtell 1997: 97).

Bei der soziotechnischen Netzwerkarbeit geht es nicht um die Unterscheidung von direkter und permissiver Kontrolle, industrialisierter und nicht-industrialisierter Wissensarbeit. Es geht um eine netzwerkspezifische Form permissiver Kontrolle und ihre typischen Netzwerkcharakteristika.

Was unterscheidet die soziotechnische Netzwerkarbeit von der permissiven Kontrolle? Das zeigt sich nicht nur an jeder der oben genannten Kategorien zum Arbeitsprozess der Softwaregestaltung (zwischen dezentral und zentral) und der Arbeit der Softwaregestaltenden (zwischen Matrix- und reiner Netzwerkorganisation). Die soziotechnische Netzwerkarbeit beruht zudem darauf, dass die Kategorien in ihrem Zusammenspiel eine flexible, situative, verteilte und horizontale Kooperation und Kommunikation ermöglichen. Rollen, Ablauf, kommunikative Beziehungen, digitale Werkzeuge und die Arbeit der Softwaregestaltenden können situativ und flexibel kombiniert auftreten. So kann die Kommunikation über vielfältige Kommunikationskanäle erfolgen – digital oder persönlich. Mal gibt der Ablauf vor, wer mitarbeitet, mal entscheiden das die Softwaregestaltenden eigenständig. Entscheidend ist die (temporäre) Praxisgemeinschaft sämtlicher Beteiligter der Softwaregestaltung unabhängig von ihrer Team-, Abteilungs- oder Organisationszugehörigkeit. In ihr stimmen sich die Beschäftigten über Erwartungen ab und es ergeben sich die Rollen, der genaue Ablauf, das Wissen, die Beziehungen und die ver-

wendeten Werkzeuge. Wer nicht Teil dieser Praxisgemeinschaft ist, hat nicht nur keine Chance mitzugestalten, sondern auch keinen Zugriff auf das Wissen und es fehlen die Möglichkeiten zu lernen. Das Wissen, das in ihr entsteht, zeigt netzwerktypische Eigenschaften: Es ist soziotechnisch im Netzwerk auf Beschäftigte und Software verteilt. Dabei ist eine gemeinsame Wissensbasis nicht garantiert. Das Wissen ergänzt sich erst in der Zusammenarbeit und es existiert ein stetiger Austausch zwischen Anwendung und Programmierung (8.4.3.2). Zu dieser netzwerkförmigen Praxisgemeinschaft gehört ein Mindestmaß an horizontaler Transparenz, ein »nicht-hierarchisches Modell reziproker Sichtbarkeit« (Schmidt 2012: 197), damit sich die Beteiligten in der Praxis untereinander eigenständig abstimmen können. Diese Transparenz zeichnet aus, dass sie soziotechnisch ist: Sie ist softwarebasiert aufgrund von softwarebasierten Werkzeugen wie Ticketsystem oder Softwareentwicklungsumgebung, wo die Beteiligten einsehen können, wer was gerade bearbeitet und welchen Status eine Aufgabe hat. Sie ist aber nicht ausschließlich softwarebasiert. Meist gehören noch Teamtreffen von Angesicht zu Angesicht dazu, die den Rahmen zur Ausübung von »peer group pressure« geben (vgl. Boes et al. 2018: 187).

In den Fallstudien zeigen sich nicht nur die gerade genannten allgemeinen netzwerktypischen Eigenschaften. Es lassen sich auch konkrete Lösungen von Kooperationsherausforderungen in organisationalen Netzwerken erkennen. INTERN1 konnte einen für seine Zwecke angepassten Scrum-Prozess über althergebrachte Abteilungsgrenzen hinweg mit vielfältigen Methoden wie Workshops, Resonanzgruppen und Prototyping etablieren. INTERN2 musste erst interne Konflikte zwischen Fachabteilungen lösen, um einen abteilungsübergreifenden Ablauf einzurichten. Bei KOOP1 kooperieren mehrere EVU und ein IT-DL. Sie haben einen ausgeklügelten Ablauf etabliert, um zu verhandeln, was das IT-DL als gemeinsamen Standard und was die EVU individuell gestalten. Bei KOOP2 ist die Kooperation teilweise prekär. An Stelle des gemeinsamen IT-DL übernehmen einzelne EVU die Softwaregestaltung wieder selbst – mit der Folge, dass diese dann bei eigenen IT-Gestaltungsprojekten intern eine abteilungsübergreifende Kooperation hinbekommen müssen. Bei KOOP3 funktioniert die Kooperation für die IoT-Softwaregestaltung zwischen IT-DL, Softwarefirma und EVU so gut, dass weder Verträge noch Hierarchien bei der Kontrolle der Arbeit eine große Rolle spielen. Die Softwarefirma aus PAKET hat intern energiewirtschaftliche Fachleute, braucht aber trotzdem Input zu Anforderungen und das Feedback ausgewählter EVU für die Gestaltung der komplizierten, industriespezifischen Standard-ERP-Software. STARTUP begreift sich anders als althergebrachte EVU als Softwarefirma, ist von Beginn an auf die Softwaregestaltung ausgerichtet (Primat der Softwareentwicklung) und arbeitet ohne formale Führungskräfte in einer Mischung aus Scrum und Holokratie.

Ist ein Arbeitsprozess der Softwaregestaltung etabliert und wollen das Management oder die Softwaregestaltenden selbst den Prozess weiter optimieren, müssen sie dessen Eigenheiten berücksichtigen. Denn statt wie in einer (Standard-)Massenproduktion ist jede Anforderung individuell in ihrer Ausarbeitung und Umsetzung. Methoden wie Scrum betonen, dass es sich nicht um einen linearen, vorweg detailliert planbaren Prozess bis zur fertigen Software handelt, sondern um einen iterativen Prozess, im Laufe dessen sich z.B. das Wissen über einen Anwendungsbereich einer Software ebenso wie jenes über die Anwendenden ändert. Es muss möglich sein, immer wieder neues Wissen

in Anforderungen aufnehmen zu können und zu lernen. In den Fallstudien zeigt sich keine Prozesstaktung, sondern vielmehr eine situative Synchronisierung von Arbeitsschritten – ob durch Zyklen wie bei Scrum oder Deadlines wie bei Projekten. Wobei diese Synchronisierung des Handelns immer Interventionen, Feedback (z.B. durch Tests) und Priorisierung durch (bestimmte) Beteiligte zulässt. Zwar ist es Teil der Softwaregestaltung, dass die »Kooperations- und Kommunikationspraxis der Arbeitsteams […] aufgedeckt, analysiert und optimiert« (Boes et al. 2018: 26) wird, z.B. durch Scrum Master:in oder die Projektleitung. Ein digitales Fließband ist aber in keiner der Fallstudien zu beobachten. Es ist in den Fallstudien nicht das primäre Ziel, »selbst hochqualifizierte Wissensarbeit systematisch und rational zu organisieren, um sie – mittels Transparenz und Kontrolle sowie einer Kollektivierung von Wissen – plan- und wiederholbar zu machen« (Boes et al. 2018: 180). Es geht darum, Kommunikation und Kooperation zu verbessern. Das schließt keine Standardisierung und Formalisierung aus. Allerdings nehmen sie dann eine andere Form an, z.B. als Standardisierung der verwendeten Softwarewerkzeuge oder Formalisierung des Scrum-Ablaufs und der Rollen. Dabei sind der Standardisierung und Formalisierung klare Grenzen gesetzt, weil die Softwaregestaltung stark von der Subjektivität der Beschäftigten abhängig bleibt und situatives Handeln zwischen Anwendung und Programmierung möglich bleiben muss. Andere Optimierungen sind zielführender. Wobei nur eine befragte Person erwähnt hat, dass ihr EVU Workshops allein mit dem Ziel durchführt, die Kommunikation zwischen den Beschäftigten zu verbessern.

Statt den Begriff der Industrialisierung auf Softwaregestaltung anzuwenden, sind andere Begriffe angebracht, um die netzwerkspezifischen Unterschiede deutlich zu machen und nicht zu suggerieren, Softwaregestaltung sei industrialisierbar. Es steht außer Frage, dass Organisationen die Möglichkeiten von Software nutzen, um zu automatisieren und Beschäftigte zu steuern. Aber für die Softwaregestaltung trifft es deshalb nicht zu, dass »insbesondere in hochqualifizierten Feldern [die] sehr ausgeprägten – typischen Freiheitsgrade der Angestellten« (Boes et al. 2018: 37) verschwinden. Die Freiheitsgrade sind gerade die Ressource, welche die Organisationen nutzen wollen und müssen. Alle Kategorien der soziotechnischen Netzwerkarbeit zeichnen sich dadurch aus, dass es nicht um die direkte Kontrolle des Verhaltens geht.

Thompson/Laaser verwenden den Begriff der qualitativen Intensivierung, um zu beschreiben, wenn Organisationen zusätzliche Fertigkeiten der Beschäftigten wie deren implizites Wissen und Gefühle nutzen (vgl. Thompson/Laaser 2021: 145). Für die vorliegende Arbeit geht es bei qualitativer Intensivierung u.a. neben Beziehungsfähigkeit, Beziehungspflege, intrinsische Motivation, sich aktiv auf Software einzulassen und Lernbereitschaft auch um ein Mehr an interdisziplinärem Wissen, thematisch unterschiedliche Projekte gleichzeitig zu machen, gesamte Prozesse zu verstehen bzw. in digitalen Prozessen denken zu müssen und mehrere Kunden und deren IT-Landschaft zu kennen (8.4.3.2). Andere Autorinnen sprechen von einer Psychologisierung der Arbeitswelt, bei der emotionale und kommunikative Kompetenzen wichtiger werden (vgl. Illouz 2008), und Firmen erwarten, dass Mitarbeitende ihre Gefühle instrumentell für die Dienstleistungsarbeit einsetzen (vgl. Hochschild 1990). Dabei geht es nicht nur darum, die subjektiven Potenziale überhaupt zu nutzen. Es geht auch um eine bestimmte Art und Weise, unter welchen Bedingungen und wie eine Form der Optimierung von Arbeit ausse-

hen und konzeptualisiert werden kann, bei der diese Arbeit in einem Netzwerk aus Subjekt, Software, Ablauf und Organisation(en/-seinheiten) stattfindet. So fällt z.B. auf, dass die Arbeitsbelastung der Befragten sehr unterschiedlich ist. Ist das womöglich typisch für die Arbeit in Netzwerken: Wer sich gut verkaufen, im Netzwerk bewegen und selbst Grenzen setzen kann, ist weniger belastet?

Es geht bei der Softwaregestaltung nicht darum, dass sich die oder der Einzelne einem objektiven, weitgehend standardisierten und formalisierten Prozess unterwirft. Statt entweder Netzwerkarbeit oder keine Netzwerkarbeit geht es bei der Softwaregestaltung darum: Ab welchem Grad an Standardisierung und Formalisierung liegt keine soziotechnische Netzwerkarbeit mehr vor und die Kontrolle der Arbeit über Hierarchien und Märkte wird aus Sicht der Softwaregestaltung dysfunktional und die Transformation der Arbeitskraft misslingt?

9.3. Zweiter Debattenbeitrag: Softwaregestaltung als Arbeit an der digitalen Transformation

9.3.1. Teil der digitalen Transformation: soziotechnische Arbeitsgestaltung der Softwareanwendung durch die Softwaregestaltung

Die Softwaregestaltung betrifft nicht nur die Softwaregestaltenden selbst (ob sie in einer Matrix- oder einer Netzwerkorganisation arbeiten). Sie hat je nach soziotechnischer Konstellation sowohl Folgen für die Software als auch für die anwendende Organisation. Anders als z.B. bei der Einführung eines ERP-Systems geht es nicht um die Folgen einer fertigen Standardsoftware und deren Anpassung (6.5.4). Es geht um die Folgen des Arbeitsprozesses der Softwaregestaltung. Wie Organisationen seine Möglichkeiten ausschöpfen, hängt von seinem Verhältnis zur Softwareanwendung ab. Besteht das Verhältnis in der reinen Zulieferung einer Software, dann sind beide Arbeitsprozesse getrennt voneinander (z.B. in einer Softwarefirma und einem EVU). Beim Primat der Softwareentwicklung bildet der Arbeitsprozess der Softwaregestaltung als Teil der Softwareentwicklung den Kern der anwendenden Organisation. Der Arbeitsprozess der Softwareanwendung organisiert sich um die stetig weiterentwickelte Software herum. Die vorliegende Studie befasst sich mit den Folgen für die Softwareanwendung, die sich aus der Softwaregestaltung ergeben, indem sie das Verhältnis von Softwaregestaltung zur Softwareanwendung betrachtet. Sie bezeichnet dieses Verhältnis mit dem Begriff der soziotechnischen Arbeitsgestaltung der Softwareanwendung.

Die Untersuchung nutzt zwei Kategorien, um in den Fallstudien die soziotechnische Arbeitsgestaltung der Softwareanwendung zu analysieren: den **Einfluss** beider Arbeitsprozesse aufeinander und die **Konflikte** zwischen ihnen. Beide Kategorien führt der nächste Punkt 9.3.2 näher aus.

Dabei zeigen die Fallstudien, dass Betriebsräte innerhalb der EVU die soziotechnische Arbeitsgestaltung mitbestimmen, indem sie intervenieren, aber nicht inhaltlich gestalten. Sie spielen bei Fragen der Partizipation insofern eine Rolle, als sie Rahmenbedingungen mitgestalten und weniger die Software selbst. So prägen sie die Partizipation z.B., wenn ein Betriebsrat einfordert, dass die Softwaregestaltung Anwendende via

Workshops einbezieht. Das Management fällt strategische Entscheidungen, gibt Ziele vor, stellt Ressourcen zur Verfügung und bringt teilweise eigene Anforderungen ein.

Bevor das Schlusskapitel den Beitrag der Softwaregestaltung zur Debatte über die digitale Transformation diskutiert, sollen vier Idealtypen verdeutlichen, welchen Unterschied die Softwaregestaltung machen kann.

9.3.2. Zwischen unabhängiger und abhängiger soziotechnischer Arbeitsgestaltung

Die vier im Folgenden vorgestellten Idealtypen verdeutlichen die Unterschiede der soziotechnischen Arbeitsgestaltung, die davon abhängen, wie Organisationen eines der Kernprobleme der Softwaregestaltung lösen: jenes der softwaretechnischen Gestaltungsmöglichkeiten. Richten sie sich organisatorisch nur auf die Softwareanwendung aus oder (auch) auf die Softwaregestaltung? Gestalten sie eine Individual- oder eine Standardsoftware? Bei den Typen INTERN und PROPRIETÄRER STANDARD sind die anwendenden Organisationen unabhängig in ihrer Arbeitsgestaltung, weil sie die Softwaregestaltung selbst in der Hand haben. Bei den Typen STANDARDPAKET und WERKVERTRAG sind sie hingegen abhängig von anderen Organisationen wie Softwarefirmen oder IT-DL (8.6.1.3.). Hier sind nur zwei Typen ausführlich mit sämtlichen Kategorien und Unterkategorien dargestellt, um die Unterschiede zwischen abhängiger und unabhängiger soziotechnischer Arbeitsgestaltung der Softwareanwendung zu verdeutlichen.

Beim Typ **STANDARDPAKET** entwickelt eine Organisation (wie z.B. in der Fallstudie PAKET) eine industriespezifische Standardlösung, die sie verkauft. Die anwendenden EVU sind von dieser Organisation (z.B. einer Softwarefirma) **abhängig**, was die soziotechnische Arbeitsgestaltung anbelangt. Warum? Das liegt am Einfluss, der sich anhand von vier Kategorien beschreiben lässt: Aufgrund des **Kontrollverhältnisses** der Arbeitsprozesse von Softwaregestaltung und -anwendung zueinander können die EVU nur die Anwendung der industriespezifischen Standardlösung kontrollieren, nicht aber die Softwaregestaltung des Standards. Sie können den Standard nur in den vorgegebenen Möglichkeiten anpassen (z.B. durch Einstellungen). Zudem **partizipieren** vorwiegend Fachexpert:innen und gestalten z.B. über Arbeitskreise oder einzelne Projekte mit. So können die EVU nur den Arbeitsprozess der Softwareanwendung **reorganisieren**. Zuletzt hat die Softwarefirma des Standardpakets das letzte Wort bei Entscheidungen über die **Ziele** der Softwaregestaltung. Das betrifft zum Beispiel Entscheidungen darüber, was in den Standard kommt und ob sie individuellen Wünschen einzelner EVU entgegenkommt. Die Abhängigkeit der EVU bei der soziotechnischen Arbeitsgestaltung zeigt sich auch an den **Konflikten**. Es sind externe Konflikte mit der Softwarefirma und die EVU können diese daher nicht unabhängig von ihr lösen, z.B. wenn es um die inhaltliche Gestaltung der Standardsoftware geht.

Beim Typ **INTERN** entwickeln EVU eine industriespezifische Software, was ihnen eine **unabhängige** Arbeitsgestaltung der Anwendung ermöglicht (wie z.B. die Fallstudien INTERN1 und INTERN2). Die EVU haben **Einfluss** auf beide Arbeitsprozesse von Softwaregestaltung und Softwareanwendung. Sie können die **Ziele** der Softwaregestaltung selbst bestimmen und auf die eigenen Bedarfe ausrichten. Zu einer solchen Ausrichtung kann gehören, dass sämtliche Anwendenden unterschiedlicher Fachbereiche

partizipieren. Das **Kontrollverhältnis** ist so, dass die Softwaregestaltung die Softwareanwendung für ihre Zwecke kontrollieren kann. Die Softwaregestaltung entscheidet, ob sie Führungskräfte der Anwendung mitgestalten lässt und welche Anforderungen die Programmierenden umsetzen. Die beiden Arbeitsprozesse von Softwaregestaltung und -anwendung kann das EVU im Sinne der Softwaregestaltung **reorganisieren** und z.B. Team- und Abteilungssilos auflösen und ein integriertes Prozessteam für die Softwaregestaltung schaffen. Die **Konflikte** sind rein intern in der Organisation, z.B. zwischen Fachbereichen oder mit dem Betriebsrat. Deshalb kann das EVU diese eigenständig lösen.

Beim Typ **PROPRIETÄRER STANDARD** schöpft eine Organisation die Möglichkeiten der Softwaregestaltung maximal aus. Sie gestaltet für die eigenen Zwecke eine individuelle Software, wobei die Softwareanwendung abhängig von der Softwaregestaltung und der entwickelten Software organisiert ist. Zusätzlich verdienen Organisationen dieses Typs mit dem Verkauf der Software auch noch Geld, indem sie andere Organisationen als Standardsoftware anwenden. Dieser Typ trifft auf einen Teil der in der Fallstudie STARTUP gestalteten Software zu.

Beim Typ **WERKVERTRAG** wird eine individuelle, industriespezifische Software durch eine Softwarefirma oder einen IT-DL im Auftrag eines EVU gestaltet. Dem Typ entsprechen KOOP1 und KOOP2 zum Teil, und zwar dann, wenn das IT-DL für ein EVU eine individuelle Software gestaltet. Wobei dies dann im Rahmen der bestehenden Kooperation geschieht und die Interviews zu diesem Fall darauf schließen lassen, dass das EVU dann zumindest mitgestaltet und nicht nur eine Beauftragung macht, wie es für einen reinen Werkvertrag typisch wäre.

9.3.3. Beitrag zur Debatte über die digitale Transformation

Indem Softwaregestaltung die Arbeit der Softwareanwendung prägt, ist sie ein Teil des soziotechnischen Wandels der Arbeitswelt. Dieser wird derzeit unter dem Begriff der digitalen Transformation diskutiert, hinter dem sich laut Pfeiffer und Schrape (2023) aus arbeits- und industriesoziologischer Perspektive allerdings keine allgemeine Theorie verbirgt. Sie nennen drei Ansätze, die versuchen, die digitale Transformation theoretisch zu fassen. Zweien davon ist die vorliegende Untersuchung in ihrer grundlegenden Ausrichtung gefolgt, indem sie wie Apitzsch et al. (2021) die Arbeit an der Digitalisierung in konkreten Arbeitswelten erforscht und wie Hirsch-Kreinsen (2020) von einer inkrementellen Digitalisierung ausgeht (6.5.3). Warum das Schlusskapitel den dritten von Pfeiffer und Schrape genannten Ansatz hier nicht weiter diskutiert, erläutert 9.3.3.2 weiter unten. Unabhängig vom Ansatz benennen Pfeiffer und Schrape als Forschungslücke, wie die konkrete Gestaltung der digitalen Transformation abläuft (2023: 137). Die vorhergehenden Kapitel haben sich damit befasst. Die folgenden Seiten erläutern, was die vorliegende Studie von den beiden oben genannten Ansätzen unterscheidet und welche Gemeinsamkeiten bei der Analyse der Gestaltung und des Verlaufs der digitalen Transformation bestehen.

Dabei sei vorweg darauf hingewiesen, dass anders als bei Hirsch-Kreinsen (2020) und Apitzsch et al. (2021) die vorliegende Untersuchung über soziotechnischen Wandel bzw. die digitale Transformation durch Softwaregestaltung gleichzeitig eine Unter-

suchung über Softwareentwicklung ist. Bei industriespezifischer Software ist die Technikgestaltung nicht mehr von der Gestaltung von Arbeit und Organisation zu trennen, weil die Folgen der Technikgestaltung für die anwendende Organisation so gravierend sind und davon abhängen, welche Möglichkeiten Organisationen bei der Softwaregestaltung nutzen. Einerseits lässt sich ein Teil des Wandels der Energiewirtschaft als soziotechnischer Wandel durch Softwareentwicklung beschreiben (z.B. durch ihre Akteur:innen, Methoden und Wissensbestände). Indem Softwareentwicklung ein Teil von (bisher) Nicht-IT-Branchen und -Firmen wird, prägt sie deren Organisation und Arbeitsweise. Andererseits zeigen die Fallstudien wie auch das Konzept der soziotechnischen Netzwerkarbeit, dass industriespezifische Softwaregestaltung und damit industriespezifische Softwareentwicklung ein soziologisches Problem sind. Die Analyse der Softwaregestaltung in der Energiewirtschaft und der Zusammenhänge zwischen soziotechnischer Konstellation, Arbeitsprozess der Softwaregestaltung, Arbeit der Softwaregestaltenden und soziotechnischer Arbeitsgestaltung der Softwareanwendung ist zugleich ein Beitrag zum besseren Verständnis von industriespezifischer Softwareentwicklung.

9.3.3.1. Gestaltung – zwischen dezentral und zentral, zwischen Standard- und Individualsoftware

Bei der Gestaltung der digitalen Transformation rückt Hirsch-Kreinsen die betriebliche Ebene in den Vordergrund und dass die Betriebe ganz unterschiedliche Wege einschlagen können, was er anhand von drei Typen zeigt (weitreichende Digitalisierer, selektive Digitalisierer, skeptische Unternehmen) (vgl. Hirsch-Kreinsen 2020: 43). Wie auch Apitzsch et al. vertritt er keinen Technikdeterminismus und sieht Betriebe als ein soziotechnisches System aus Technik, Mensch und Organisation an (vgl. ebd.: 88). Für Apitzsch et al. (2021) hängt die Gestaltung von mehreren Ebenen ab (Betrieb, Arbeitsprozess, Subjekt, organisationales Feld, Arbeitsbeziehungen, Recht). Womit sie anders als Hirsch-Kreinsen die überbetriebliche Ebene in den Fokus rücken. Sie konstatieren, dass Betriebe Gestaltungsspielräume von IT nicht nutzen, weil »prozessferne Akteure (zentrale Planungsbereiche oder externe Technikanbieter) eine dominante Rolle spielen« (ebd.: 27). Die Partizipation betrifft »oftmals […] die reine Implementation einer bereits vorgegebenen Technik und hängt jedenfalls stark von den vorhandenen betrieblichen Partizipationskulturen ab« (ebd.).

Die vorliegende Untersuchung betont, dass auch die Entwicklung von Software kein technikdeterministischer, sondern ein soziotechnischer Gestaltungsprozess ist. Zudem ist in den Fallstudien der Betrieb eine wichtige Ebene der Gestaltung und es bestehen große Unterschiede zwischen den EVU, wie sie die Möglichkeiten der Softwaregestaltung nutzen. Wie die Fallstudien zeigen, kann ein dezentraler Arbeitsprozess der Softwaregestaltung in einem EVU auch nur aus wenigen Beschäftigten bestehen, die für einen Anwendungsbereich eine Software gestalten. Damit ergeben sich vielfältige Gestaltungsmöglichkeiten innerhalb eines Betriebs. Aber wie bei Apitzsch et al. (2021) spielt bei der Mehrzahl der Fallstudien bei der Softwaregestaltung die überbetriebliche Ebene eine Rolle. So gestalten in der Fallstudie KOOP1 mehrere EVU kooperativ einen Standard, und auch individuelle Programmierungen für einzelne EVU erledigt zum Teil das IT-DL. Solch ein zentralisierter Arbeitsprozess der Softwaregestaltung verlangt allerdings Kompromisse, denen sich die einzelnen Betriebe fügen müssen.

Apitzsch et al. (2021) ist insofern recht zu geben, als dass die EVU oftmals eine vorgegebene Technik in Form einer Standardsoftware einsetzen und häufig externe technikanbietende Unternehmen wie Softwarefirmen über die Gestaltung entscheiden. Allerdings erlaubt es der Fokus auf den Arbeitsprozess der Softwaregestaltung, die vielfältigen Formen der Softwaregestaltung und damit der Gestaltungsmöglichkeiten in den Blick zu nehmen. So zeigen die Fallstudien die typischen Muster einer Gestaltung durch soziotechnische Netzwerkarbeit: Gleichzeitigkeit und Wechsel zwischen Zentralisierung und Dezentralisierung, zwischen Individual- und Standardsoftware, zwischen einer rein anwendenden Organisation und einer (mit)gestaltenden Organisation. Zudem ist die Partizipation an der Softwaregestaltung in einem Netzwerk verteilt, bei dem Anwendende, Betriebsrat und Management unterschiedlich stark beteiligt sind.

Es wirken nur ausgewählte EVU an der Gestaltung der industriespezifischen ERP-Software der Fallstudie PAKET mit. Die Vielzahl der anwendenden EVU hat sehr geringen Einfluss auf die Softwaregestaltung. Es zeigen sich Unterschiede, wie EVU die Einstellungsmöglichkeiten am Standard nutzen und sich die anwendenden Organisationen an der Standardlösung ausrichten. In den Fallstudien, die SAP einsetzen, zeigt sich beides: Anwendung eines Standards und dessen individuelle Gestaltung und damit sowohl eine Abhängigkeit und Unabhängigkeit bei der Arbeitsgestaltung der Softwareanwendung. In den Fällen INTERN1 und INTERN2 nehmen die EVU weitreichende Erweiterungen oder Anpassungen des SAP-Standards selbst vor. Auch in den EVU der Fallstudien KOOP1 und KOOP2 können einzelne Teams eigenständig Software gestalten. Vor allem KOOP1 zeigt die Dynamik innerhalb von EVU, dezentral Rollen und Abläufe für die Softwaregestaltung auf- und abzubauen, Key User:innen als permanent Softwaregestaltende und/oder flexible Softwaregestaltende wie IT-Beratende einzusetzen. Auch wenn die vorliegende Untersuchung nicht sämtliche Arbeitsprozesse der Softwaregestaltung der EVU ausleuchten konnte, entsteht doch vor allem bei jenen Fallstudien, für die viele Interviews vorliegen (KOOP1, KOOP2), der Eindruck, dass die EVU an allen Ecken und Enden Software im größeren und kleineren Umfang gestalten. Das liegt auch daran, wie der Fall KOOP3 zeigt, dass kleinere Entwicklungen nicht aufwendig sind. In der Fallstudie erlaubt es die modulare IoT-Softwarelösung, dass sowohl EVU wie IT-DL selbst gestaltete und programmierte Module anschließen. Ein EVU von KOOP2 etabliert eine eigene, kleine Entwicklungsplattform, um kleinere Anwendungen dezentral in unterschiedlichen Fachbereichen entwickeln zu können.

Neben diesen typischen Mustern zwischen Zentralisierung und Dezentralisierung und Individual- und Standardsoftware zeigen sich große Unterschiede, ob EVU rein anwendende Organisationen sind oder auch mitgestalten. In den Fallstudien sind es die größeren EVU, die eher Ressourcen dafür haben, um selbst Software und damit die eigene Arbeit zu gestalten. EVU und IT-DL aus vier Fallstudien (INTERN1, INTERN2, KOOP1, KOOP2) nutzen die Entwicklungsplattform von SAP, die ermöglicht, dass Unternehmen die ERP-Standardsoftware individuell anpassen und erweitern. Kleine EVU fokussieren sich auf die reine Anwendung. Sie sind wie bei PAKET abhängig von Softwarefirmen, die keine Entwicklungsplattform zur Verfügung stellen, wodurch die EVU nur Einstellungen vornehmen können – wofür sie teilweise gar nicht die Kapazitäten haben. Dabei kann die Gestaltung einer industriespezifischen Software und die Abhängigkeit von der Softwarelieferfirma so weit gehen, dass eine industriespezifische

Anwendungsplattform auf Basis einer Standardsoftware entsteht, die die EVU nutzen. Zwei Befragte sprechen davon, dass das IT-DL bzw. das Softwareunternehmen Stadtwerke »spielen« (so ein Befragter) können und dies auch tun. Sie trennt von einem EVU nur noch, dass sie Netze besitzen oder Verträge mit Endkunden haben. Die Software, um die notwendigen industriespezifischen Prozesse abzuwickeln, haben sie. Ein Beispiel dafür ist die Standard-ERP-Software, die das IT-DL von KOOP1 für mehrere EVU betreibt und gestaltet. Das IT-DL übernimmt alle Anpassungen durch Einstellungen an der Standardsoftware oder die Programmierung von Erweiterungen. Ein anderes Beispiel ist die Zusammenarbeit zwischen einer Softwarefirma und einem EVU in der PAKET-Fallstudie. Das EVU betreibt den von der Softwarefirma entwickelten industriespezifischen Standard mit entsprechenden Standardeinstellungen auf einem Server. In beiden Fällen bieten IT-DL bzw. Softwarefirma an, einzelne Prozesse via BPO für EVU zu übernehmen, d.h. nicht nur die (fertig eingestellte) industriespezifische Software direkt nutzbar zur Verfügung zu stellen, sondern auch ihren Betrieb zu verantworten und die Anwendenden zur Verfügung zu stellen, die die Software bedienen.

Der Fokus auf den Arbeitsprozess der Softwaregestaltung lässt es zu, in jeder Fallstudie zu untersuchen, inwiefern Anwendende in ihn eingebunden sind. Eine direkte Partizipation der Anwendenden findet in den Fallstudien allerdings nur punktuell und meist vermittelt z.B. über Key User:innen oder Anforderungsmanagende statt. Oftmals schreiben fachliche Expert:innen Anforderungen, welche die Software selbst gar nicht anwenden. So bewegt sich die Softwaregestaltung in den Fallstudien zwischen einer repräsentativen Technokratie und einer kapitalistischen Technokratie (Näheres zu diesen Partizipationstypen unter 8.5.3.3). Vielmehr gehören die Softwaregestaltenden zur Gruppe der Höherqualifizierten, die die Arbeit der Anwendenden gestalten und in den betrieblichen Hierarchien höhergestellt sind. Die Mehrzahl der Anwendenden sind das Objekt der Softwaregestaltung und müssen mit ihren Ergebnissen leben. So zeigen die Fallstudien, dass meistens die Grundstruktur der Technikgestaltung bleibt: Einige wenige gestalten die Arbeit vieler. Das zeigt sich vor allem dann, wenn die Softwaregestaltung außerhalb der anwendenden Organisation stattfindet. Ein Extremfall ist eine Cloud-Anwendung, auf deren Gestaltung der befragte Anwender und sein EVU gar keinen Einfluss haben (EVU von KOOP2). In einem anderen Fall ist der Anwender fast ein Jahr aufgrund von Burnout ausgefallen (EVU von PAKET), weil über einen längeren Zeitraum eine Kombination aus Termindruck durch die Regulierung und stetigen Updates der extern entwickelten Standardsoftware bestand. In beiden Fällen fungieren Anwendende als Puffer. Sie sind zwar abhängig von einer funktionierenden Software, haben aber ihre Gestaltung nicht in der Hand und müssen fehlerhafte oder nicht fertige Software durch ihre Arbeit kompensieren. Eine im 6. Kapitel (6.5.4) zitierte Studie kommt zu dem Schluss, dass nach einer ERP-Implementierung die Anwendenden mit einer nicht passenden Software zurechtkommen mussten:

> »In short, users were left to pick up the pieces after a so called ›successful‹ implementation.« (Lyytinen/Newman 2015: 90)

Ob dies bei Standardsoftware oder Cloud-Lösungen häufiger vorkommt und unter welchen Bedingungen Anwendende als eine Art Puffer fungieren müssen, wäre noch weiter

zu untersuchen. Aber selbst bei den Fallstudien, bei denen die anwendende Organisation eine eigene individuelle Software gestaltet, ist eine direkte Partizipation der Anwendenden und eine Umsetzung ihrer Anforderungen nicht garantiert. STARTUP gestaltet zwar unabhängig die eigene Software, bezieht die Anwendenden aber nur ein, um eine fertige Umsetzung zu testen oder Feedback zu ihr zu geben.

Ob die interne Gestaltung von Software mit den Partizipationskulturen zusammenhängt, wie Apitzsch et al. (2021) schreiben, kann die vorliegende Untersuchung nur dahingehend stützen, dass der Betriebsrat zwar keinen direkten Input zur Softwaregestaltung liefert, aber bei INTERN1 explizit die Beteiligung der Anwendenden angefordert hat. Sonst interveniert er vor allem, um eine individuelle Leistungskontrolle zu verhindern (8.5.3.2). Software zu gestalten, um die »Potenziale einer qualifikationsorientierten Gestaltung der Arbeit bestmöglich auszuschöpfen« (Hirsch-Kreinsen 2020: 88), kam in keiner der Fallstudien vor. Es wäre zu untersuchen, wie so ein Prozess der Softwaregestaltung aussehen und unter welchen Bedingungen ihn ein EVU etablieren würde.

Grundsätzlich ist Hirsch-Kreinsen (2020) zuzustimmen, wenn er von »widersprüchlichen Perspektiven für das untere und mittlere Management« (77) schreibt und davon, dass es neue Aufgaben für das Management gibt, weil IT- und Produktionskompetenz in manchen Managementpositionen verschmelzen (vgl. ebd.: 78). Der Blick auf die soziotechnische Arbeitsgestaltung durch die Softwaregestaltung zeigt, dass nicht das Management die Technik- und Arbeitsgestaltung und damit die Rationalisierung in den Organisationen allein betreibt (8.6.4). Die Führungskräfte – falls vorhanden – sind Rahmengeber und nur teilweise direkt involviert. Sie haben die Kontrolle über finanzielle und personale Ressourcen, fällen strategische Entscheidungen, setzen Ziele und kontrollieren Ergebnisse. Sie entscheiden über betriebsinterne Karrieren – egal ob von Personen der Softwareanwendung, -gestaltung und -programmierung. Sonst unterscheidet sich von Fall zu Fall, welche Rolle das Management und bestimmte Führungsebenen spielen: ob sie ein Hindernis für die Softwaregestaltung sind oder diese selbst vorantreiben und entsprechendes Wissen dazu haben. Wenn die Softwaregestaltung mehrere Abteilungen und Teams betrifft, hängen die Softwaregestaltenden in einer Matrixorganisation davon ab, dass die Führungskräfte die Softwaregestaltung unterstützen. Wenn die Softwaregestaltenden hierarchisch nicht höhergestellt sind, hängt es von der Kooperationsbereitschaft anderer Führungskräfte ab, wie viel Einfluss der Arbeitsprozess der Softwaregestaltung auf die Softwareanwendung innerhalb eines EVU hat. Die Anforderungsrunde von INTERN2 hat z.B. erst funktioniert, als die Fachbereiche kooperationswillig waren. Basiert eine Firma auf einer Netzwerkorganisation mit flachen Hierarchien, sind Teile des Managements überflüssig. Lagert ein EVU die Softwaregestaltung komplett aus, muss sich das Management zwar nicht mehr um die Kontrolle dieser Arbeit kümmern. Es ist dafür aber von externen Softwarefirmen abhängig. Es ist offen, inwiefern Softwaregestaltende Teil des Managements werden. Wenn nicht nur die informationsvermittelnde Funktion des mittleren Managements durch Softwarelösungen wie ERP-Systeme entfällt (Hohlmann 2007: 365f.), sondern auch die der disziplinarischen Führung bei Netzwerkarbeit und Teile der Arbeitsgestaltung zur (externen) Softwaregestaltung wandern: Welche Hierarchien und welches Management brauchen anwendende Organisationen dann überhaupt? Zudem wäre noch zu vertiefen, aus welchen Gründen sich das Management für welche Strategie entscheidet und inwiefern und welche EVU bei der

Softwaregestaltung abhängig sind. Denn der Markt für industriespezifische Software ist groß (7.2.1.1). Die EVU können auch einzelne, industriespezifische Module von unterschiedlichen Softwareherstellern beziehen und so ist deren Marktmacht beschränkt. Andererseits wird es gerade jenen EVU schwerfallen, das softwarezuliefernde Unternehmen zu wechseln, die viel in die individuelle Anpassung und Erweiterung von z.B. SAP investiert haben.

9.3.3.2. Verlauf – die Möglichkeiten der Softwaregestaltung ausreizen

Was den Verlauf der digitalen Transformation anbelangt, grenzen sich Hirsch-Kreinsen (2020) und Apitzsch et al. (2021) von Ansätzen ab, die allgemein einen disruptiven Wandel beobachten. Dabei befasst sich Hirsch-Kreinsen mit industrieller Arbeit, bei der er eine inkrementelle Optimierung gegebener Prozesse und eine Pfadabhängigkeit ausmacht. Die Betriebe setzen bei eingespielten Arbeitspraktiken und vorhandenen Qualifikationen an, um in kleinen Schritten die Produktion durch den Einsatz von digitaler Technik zu verbessern (vgl. Hirsch-Kreinsen 2020: 40ff.). Auch Apitzsch et al. machen eine Pfadabhängigkeit aus, wobei sie branchenunabhängig eine strukturierte Vielfalt, Ungleichheiten, Widersprüche und eine plurale Digitalisierung wahrnehmen (vgl. Apitzsch et al. 2021: 20f.).

Auch in den Fallstudien der vorliegenden Untersuchung zeigen sich ein inkrementeller Wandel und Pfadabhängigkeiten. Anders als Hirsch-Kreinsen (2020) und Apitzsch et al. (2021) fokussiert sie aber den konkreten Arbeitsprozess der Softwaregestaltung, der den soziotechnischen Wandel vorantreibt. So wird eine besondere Form einer technikentwicklungsbezogenen Rationalisierung sichtbar und dass der Verlauf der digitalen Transformation auch davon abhängt, inwieweit Organisationen die Möglichkeiten der Softwaregestaltung ausreizen. Damit erklärt das Zusammenspiel von Arbeit, Organisation, Software und Softwaregestaltung, wie die digitale Transformation verläuft. Gestalten EVU nicht nur eine Software (mit), sondern ändern auch ihre Organisation, um die Softwaregestaltung zu optimieren? Wo liegen die Grenzen des Einsatzes von Software in einem Anwendungsbereich und inwieweit kann der Verlauf der Softwaregestaltung disruptive Veränderungen erklären? Setzen sich EVUs mit Individualsoftware in der Branche durch oder Softwareunternehmen bzw. IT-DL mit einer Standardanwendungsplattform? Im Folgenden werden diese Fragen zusammenfassend beantwortet.

Softwaregestaltung ist ein inkrementeller Wandel in Form einer **technikentwicklungsbezogenen Rationalisierung** von Software und ihres Anwendungsbereichs (8.6.3). Organisationen der Branche setzen das Mittel der Softwaregestaltung und -programmierung für die eigenen Zwecke ein. Besonderes Kennzeichen dieses Rationalisierungstyps ist der Fokus auf die Arbeitsteilung zwischen Softwareanwendung, -gestaltung und -programmierung, stetiges Abgleichen branchenspezifischer Bedarfe mit softwaretechnischen Möglichkeiten und die Dynamiken zwischen zentraler und dezentraler Softwaregestaltung. Organisationen prüfen ausgehend von ihrem Status quo, ob in einem Anwendungsbereich ein Standard oder eine individuelle Gestaltung zielführender ist, ob sie sich auf die Kontrolle der Anwendung einer Software konzentriert oder auch die Kontrolle des Arbeitsprozesses der Gestaltung übernimmt. Stetig ist abzuwägen zwischen niedrigen Kosten und ausgelagerter Softwaregestaltung für einen (automa-

tisierten) Standardsoftwareprozess – mit den entsprechenden Nachteilen, dass so z.B. nur begrenzt eine individuelle Prozessgestaltung möglich ist. Gleichzeitig ist eine Standardsoftware in Bereichen, in denen die Software keine Wettbewerbsvorteile bringt, zielführender und vor allem für jene EVU die bessere Wahl, die sich mit der internen Kontrolle der Softwaregestaltung schwertun. Zu einer individuellen Softwaregestaltung in der eigenen Organisation müssen EVU fähig sein und die entsprechenden Methoden und Beschäftigten organisieren können: sei es IT-Projekte durchzuführen, Scrum-Teams aufzubauen und Product Owner:innen oder Anforderungsmanagende einzustellen und mit Programmierenden zusammenzuarbeiten. Wie die Fallstudien zeigen, kann ein EVU unterschiedliche Entscheidungen für die vielzähligen Anwendungsbereiche treffen und jeweils mal eine industriespezifische Standardsoftware einsetzen und mal eine individuell gestalten.

Neben diesem kontinuierlichen Prozess, die Möglichkeiten der Softwaregestaltung zu prüfen, zeigen die Fallstudien, dass der Verlauf der digitalen Transformation davon geprägt ist, dass die EVU die Möglichkeiten der Softwaregestaltung unterschiedlich ausschöpfen. Allgemein verändern sich die EVU in den Fallstudien iterativ von ihrem Status quo aus. Wenige EVU ändern sowohl die Software als auch ihre Organisation und z.B. die Teamgrenzen, um die prozessübergreifende, interdisziplinäre Zusammenarbeit für die Softwaregestaltung zu verbessern. Die anwendenden Organisationen behalten ihre alten Strukturen aus Abteilungen und Hierarchien bei und etablieren die Softwaregestaltung quer dazu, so dass eine Matrixorganisation entsteht. Teils nutzen die Organisationen neuste digitale Werkzeuge, teilweise aber unterschiedliche Ticketsysteme je Abteilung oder nur Excel und E-Mails für die Softwaregestaltung. Organisatorisch hat nur INTERN1 die Anwendung reorganisiert und die Hierarchieebene der Meister:innen abgeschafft – und dass nicht für den Arbeitsprozess der Softwaregestaltung, sondern aufgrund der neu entwickelten Software. INTERN2 denkt über eine Reorganisation hin zu einer Prozessorganisation nach, die keine Abteilungs- und Teamgrenzen mehr kennt und einen Vorteil für den Arbeitsprozess der Softwaregestaltung bietet. Wie der Wandel von INTERN2 aussehen könnte, wenn z.B. Team-Silos aufgelöst sind und es nur noch Führungskräfte auf der Ebene des digitalen Gesamtprozesses gibt, zeigt das Beispiel einer auf Softwaregestaltung ausgerichteten softwaretechnischen Prozessorganisation (8.5.3). Bei EVU2 von KOOP2 fällt auf, wie sehr die alten Strukturen bestehen bleiben und die Arbeitsprozesse der Softwaregestaltung sich in diese einfügen müssen. Einige EVU von PAKET richten sich organisatorisch weniger an Abteilungen und Teams aus und mehr an übergreifenden Prozessen, z.B. indem sie übergreifende Projekte durchführen oder übergreifend die ERP-Software einstellen. Diese EVU haben aber, was die Standardsoftware anbelangt, wenige Einflussmöglichkeiten auf die Softwaregestaltung. Unternehmen wie STARTUP, die den Primat der Softwareentwicklung leben und sich von Anfang an als softwarebasierte Unternehmung begreifen, haben bei der Softwaregestaltung einen Vorteil gegenüber all jenen, die noch an althergebrachten Strukturen festhalten.

Die Firmen reizen die Möglichkeiten der Verwendung von Software je nach Anwendungsbereich unterschiedlich aus. Denn neben dem Verhältnis der beiden Arbeitsprozesse zueinander ist der Anteil der Datenverarbeitung im Anwendungsbereich entscheidend dafür, was die Softwaregestaltung überhaupt leisten kann und wie weit die digitale

Transformation geht. Kann die Arbeit weitgehend in Software abgebildet werden, dann ist eine weitgehende Automatisierung möglich und die Anwendenden bearbeiten vorwiegend (noch) nicht automatisierbare Restfälle. Ist die eigentliche Arbeit im Anwendungsbereich nicht digitalisierbar, geht es um automatisierte Steuerung von Arbeitskräften, deren Arbeit Software nicht erledigen kann. Dann werden die Anwendenden Teil eines digitalen Prozesses, der ihre Arbeit steuert, aber nicht ersetzt. In den Fallstudien liegt der Schwerpunkt auf Anwendungen, in denen nur Daten verarbeitet werden oder die Arbeit gesteuert wird, also nicht wie bei Hirsch-Kreinsen (2020) auf dem Einsatz in der Industrie zur Steuerung von Maschinen. Die Grenzen der Softwaregestaltung in einem Anwendungsbereich liegen in den Fallstudien darin, dass Organisationen häufig Software nur für einen Teil eines Prozesses gestalten. Andere Teile des Prozesses sind in der Verantwortung anderer Teams oder Firmen. Somit hängt die Durchschlagskraft der Softwaregestaltung auch davon ab, ob ein gesamter Arbeitsprozess der Anwendung unter ihrer Kontrolle ist oder nicht. Womit wir wieder beim Verhältnis der beiden Prozesse zueinander wären.

Kann der Arbeitsprozess der Softwaregestaltung auch einen disruptiven Verlauf der digitalen Transformation erklären? Für das Unternehmen aus der Fallstudie STARTUP ist Softwaregestaltung im Unterschied zu den anderen der Kern von Arbeit und Organisation. Es ist ein Musterbeispiel dafür, die Möglichkeiten individueller Softwaregestaltung auszureizen: Die Organisation basiert auf dem Primat der Softwareentwicklung. Die erbrachte Leistung, mit der das Unternehmen Geld verdient, ist weitestgehend in Software abgebildet. Zusätzlich bietet STARTUP die selbst entwickelte Software auch noch anderen an, verdient damit Geld und ist somit ein Beispiel für den oben beschriebenen Typ PROPRIETÄRER STANDARD. Trotzdem passt der Begriff disruptiv für STARTUP nicht. Warum? STARTUP bedient mit dem Emissionshandel für E-Mobilität nur eine Nische, von der aus eine Disruption der Energiebranche nicht möglich ist. Es gibt auch keine etablierte Konkurrenz, die durch die digitale Abbildung des Geschäfts zu verdrängen wäre.

Solche Nischen der Regulierung mit Software zu bedienen, ist typisch für die Branche (siehe Kapitel 7). In der Energiewirtschaft konnte sich in den Kerngeschäftsfeldern wie Netze, Lieferung, Handel und Erzeugung noch kein EVU allein durch eine individuelle Software (und deren Verkauf) durchsetzen (auch wenn einige EVU mit IT-Dienstleistungen Geld verdienen). Die meisten untersuchten EVU nutzen eine Mischung aus Standardsoftwarepaketen und individuell gestalteter Software. Viele EVU müssen die Möglichkeiten individueller Softwaregestaltung nicht nutzen, um Gewinne zu erwirtschaften und am Markt zu bestehen. Ob der Anteil an Standardsoftware in den EVU größer oder kleiner wird, ist eine offene Frage. Damit kann die vorliegende Untersuchung für die von Bessen (2022) beschriebene Strategie des »Competing on Complexity« (4.1) kein exemplarisches Unternehmen in der Energiewirtschaft finden. Die IT-Landschaften der EVU werden zwar umfangreicher. Es gibt allerdings keine Hinweise dafür, dass sie sich gerade dadurch entscheidend am Markt durchsetzen. So erhofft sich das EVU in Fallstudie INTERN1 dank individueller Software die Instandhaltung immer besser erledigen und sich so von konkurrierenden Firmen absetzen zu können, wenn es um die Konzessionsvergabe von Netzen geht. Ein EVU von KOOP1 versucht mit einem individuellen Portal für seine Kundschaft zu punkten. Was solch eine Softwaregestaltung wirklich an

zusätzlichen Marktanteilen oder Umsätzen bringt, müsste eine weitere Untersuchung erforschen.

Eine andere Form von starkem soziotechnischem Wandel wäre es, wenn sich eine Softwarefirma mit einer industriespezifischen Standardlösung durchsetzt, auf der EVU oder das Softwareunternehmen selbst sämtliche digitalen Kernprozesse der Branche abwickeln würden: eine Anwendungsplattform, auf der die große Mehrheit der Anwendenden der Branche arbeitet. Das wäre allerdings vor allem eine starke Veränderung für all jene Beschäftigten, die derzeit in den EVU oder IT-DL Software gestalten und programmieren, weil diese ihre Arbeit verlieren würden. Solch eine neue Softwarefirma, die mit ihrer Standardlösung den Markt erobert, zeichnet sich aktuell nicht ab. Zahlen dazu, wie viele Anwendende auf den aktuell existierenden Anwendungsplattformen der diversen IT-DL arbeiten, die auf den Standard-ERP-Systemen wie von SAP, SIV oder Schleupen basieren, oder wie viele Anwendende Standard-Cloud-Angebote nutzen, gibt es jedoch nicht (7.2.1). Das gilt auch für spezielle Standardsoftwarelösungen und Plattformen wie für den Energiehandel. Es ist auch nicht zu unterschätzen, dass Softwarefirmen intern nicht all das industriespezifische Wissen haben, um eine Standardsoftware zu entwickeln. Sie sind von der Kooperation einzelner EVU abhängig. Es sind große Investitionen notwendig, bevor eine Industriestandardlösung vorhanden ist. Neue anbietende Unternehmen wie Powercloud fangen deshalb erst damit an, für einzelne Geschäftsfelder oder Anwendungsbereiche Lösungen anzubieten.

Letztendlich hängt ein disruptiver Verlauf der digitalen Transformation noch von anderen Dingen wie der Softwaregestaltung ab. Hier kommt der dritte Ansatz ins Spiel, den Pfeiffer und Schrape (2023) anführen und dem die vorliegende Untersuchung nicht gefolgt ist, weil er weniger zur Fragestellung passt: Butollo et al. (2021) sehen Produktions- und Geschäftsmodelle als wichtig an, um die digitale Transformation zu untersuchen. Wenn z.B. ein Stromlieferant den Markt erobern will, indem sie eine Discounter-Strategie fährt, bietet es sich für sie möglicherweise eher an, auf eine Standardlösung zu setzen. Sie muss z.B. die Software nicht individuell gestalten, weil sie keine individuellen Produkte anbietet. Geschäftsmodelle in die Analyse einzubeziehen, könnte womöglich auch erklären, warum Stadtwerke, die seit über zwei Jahrzehnten eine eigene industriespezifische Standardsoftware haben, trotzdem nicht den Markt erobert haben: Weil es möglicherweise nicht Teil ihrer Strategie war?[1] Es ist einerseits naheliegend, dass die Möglichkeiten einer digitalen Disruption beschränkt sind, weil Teil des Produktionsmodells der Energiewirtschaft eine integrierte Netz-Infrastruktur, ein physikalisches Gesamtsystem (Produktion und Konsum von Strom müssen sich zu jedem Zeitpunkt die Waage halten) und eine starke Regulierung sind. Andererseits müsste das aber noch genauer untersucht werden. Eine offene Forschungsfrage ist darüber hinaus, inwiefern sich kommunale, staatliche und private Unternehmen in der digitalen Transformation unterscheiden.

1 In einigen Bundesländern sind den wirtschaftlichen Aktivitäten von kommunalen Unternehmen Grenzen gesetzt (7.1.2.3). Inwiefern dies in diesem Fall und in welchem Umfang bei der Digitalisierung eine Rolle spielt, wäre zu untersuchen.

9.3.3.3. Besonderheiten der digitalen Transformation in der Energiewirtschaft

Für die digitale Transformation der Energiewirtschaft ist die Regulierung ein besonderer Treiber und es gibt Besonderheiten in der Dynamik zwischen Individual- und Standardsoftware. Die zunehmende Digitalisierung der Branche zeigen zudem Zielkonflikte in ihrer Governance auf, wie der folgende Abschnitt ausführt.

Das Besondere an der Energiewirtschaft ist, dass ein Teil der digitalen Transformation weniger darauf zurückzuführen ist, dass Unternehmen mit Softwaregestaltung Märkte erobern oder die Energiewende vorantreiben können, sondern vielmehr darauf, dass die Energiewirtschaft mithilfe von Software zum Markt wird (7. Kapitel). Die von der staatlichen Regulierung verfolgte Markt-Governance seit 1999 für Strom und 2007 für Gas basiert auf Software. Das gilt für den Wechsel des energieliefernden Unternehmens durch Privatkund:innen per Webseite, die diversen Transparenzvorschriften, wie sie auf der Stromrechnung sichtbar sind (u.a. der Herkunftsnachweis für Ökostrom), den Datenaustausch zwischen den Tausenden von energiewirtschaftlichen Organisationen, den Energiehandel oder die Vermarktung von dezentralen Erzeugungsanlagen erneuerbarer Energie via virtuelle Kraftwerke. Die Regulierung basiert auf Softwareentwicklung und weil sie sich die letzten Jahrzehnte stetig geändert hat, ist auch stetige Softwaregestaltung notwendig.

Dabei zeigt sich eine besondere Dynamik aus individueller und Standardsoftware in der Energiewirtschaft. Das liegt wieder an der Regulierung. Für Software, die auf Regulierung beruhende Prozesse umsetzt, bietet sich eine Standardgestaltung an, weil diese Prozesse für alle Unternehmen gleich sind. Das ist ein Vorteil für die Zulieferindustrie von Software, die an der Gestaltung von Standardsoftware interessiert ist. Auf individuelle Softwaregestaltung setzen EVU eher im nichtregulierten Bereich des Energievertriebs, um sich von der Konkurrenz abzugrenzen. Daneben ist für die Dynamik aus Individual- und Standardsoftware in der Energiewirtschaft typisch, dass sie sich in organisationsübergreifenden, regionalen Kooperationen (KOOP1, KOOP2, KOOP3) für die Softwaregestaltung niederschlägt. In diesen verhandeln mehre Organisationen, was sie als gemeinsamen Standard und was jede für sich individuell umsetzt. Sowohl die Branchenanalyse im 7. Kapitel wie auch die Fallstudien legen zwar nahe, dass größere und mittlere EVU mehr Geld für Software ausgeben und stärker in die Softwaregestaltung involviert sind. Doch wäre der Zusammenhang zwischen EVU-Größe und Softwaregestaltung noch näher zu untersuchen. Dies könnte zum Beispiel geschehen, indem an erforscht, wann kleinere EVU nur fertige industriespezifische Standardpakete nutzen und wann und wie sie selbst Software gestalten.

Software und ihre Gestaltung ist Teil der Industrie-Governance der Energiewirtschaft geworden, weil sie eine IT-basierte Branche geworden ist. Indem die Grenzen zwischen IT- und Energiewirtschaft verschwimmen (z.B. indem EVU IT-DL werden, Softwarefirmen energiewirtschaftliche Geschäftsprozesse abwickeln oder digitale Startups entstehen) und die Markt-Governance so sehr auf digitalen Prozessen basiert, wird Software Teil der Governance-Strukturen der Branchen. Es geht nicht mehr nur darum, eine Energieinfrastruktur zu betreiben, sondern auch eine digitale Infrastruktur. Datenschutz, IT-Sicherheit, digitale Geschäftsfelder, die Macht einzelner Softwareunternehmen und IT-DL sowie der IT-Fachkräftemangel treiben die EVU um. Die enorme Bedeutung von Software schlägt sich auf den vier Ebenen der Industrie-Governance

nieder: Erstens prüfen Unternehmen ihre Strategien beim Softwareeinsatz – ob für das gesamte Unternehmen, einzelne Wertschöpfungsstufen oder Anwendungsbereiche. Zweitens sind Produkte wie Stromtarife der EVU digital verfügbar und lassen sich digital verkaufen. Drittens ist der Datenaustausch entlang der Wertschöpfungskette von Erzeugung, Handel, Netzen, Vertrieb softwarebasiert und es gibt unzählige Kooperationen zwischen EVU zur Entwicklung von Software. Viertens schlägt sich die gesteigerte Bedeutung von softwaretechnischer Interdisziplinarität und Wissensarbeit in einer Akademisierung der Beschäftigten nieder. Sie arbeiten zunehmend an automatisierten, softwarebasierten Prozessen. Die in der Branche weitverbreiteten Betriebsräte sind gefragt, bei der Softwaregestaltung und dem Softwareeinsatz die Interessen der Beschäftigten zu vertreten.

Bei der Debatte über Veränderung der Governance einer Industrie durch die Liberalisierung spielt die IT nur am Rande eine Rolle (vgl. Mayntz 2009). Doch entstehen durch die IT Zielkonflikte in der Branche. Sie resultieren daraus, dass Software Strukturen ermöglicht, erhält und schafft, die deutlich bürokratischer und damit teurer sind als jene vor der Liberalisierung. Lohnt sich der ganze digitale Aufwand, nur um wettbewerbliche Strukturen und einen europäischen Strommarkt zu etablieren, um selbst kommunale Versorgungsunternehmen dort einzugliedern? Denn die Energiepreise sind seit Beginn der Liberalisierung stetig gestiegen[2], viele Konsument:innen wechseln das Gas-/Stromlieferunternehmen selten oder nie[3] und die reduzierten CO_2-Emissionen lassen sich mehr auf andere, CO_2-arme Produktionsanlagen und weniger auf eine zunehmend digitale Organisation der Branche zurückführen. Für wen bietet die viele Arbeit an industriespezifisch gestalteter Software eigentlich welche Vorteile?

Wer nun womit Geld verdient (ob mit Software oder Energie), sich wie durchsetzt (dank Softwaregestaltung und/oder Softwareanwendung) und wie nachhaltig diese Strukturen sind: Für die einzelnen Betriebe stellt sich die Frage, wo die Reise der technikentwicklungsbezogenen Rationalisierung durch Softwaregestaltung in der Branche hingeht und ob jene EVU, die führend in der Softwaregestaltung sind, in Zukunft auch führend in der Branche sein werden. Zu einer soziotechnischen Organisation, welche die Möglichkeiten der Softwaregestaltung (ob individuell oder Standard) für ihre Zwecke einsetzt, gibt es aktuell in der Energiewirtschaft keine Alternative.

9.3.3.4. Folgen für die Arbeit in den EVU

Neben der soziotechnischen Arbeitsgestaltung durch Softwaregestaltung, die bedeuten kann, dass Anwendende den Bedarfen des Arbeitsprozesses der Softwaregestaltung untergeordnet sind, gibt es noch weitere Folgen für die Arbeit in den anwendenden EVU. Hirsch-Kreinsen beschreibt drei Entwicklungsszenarien der Digitalisierung von Arbeit: Substitution (Arbeit ersetzen), Upgrading (Qualifikationsniveaus steigen) und

2 Haushaltskunden: https://de.statista.com/statistik/daten/studie/914784/umfrage/entwicklung-der-strompreise-in-deutschland-verivox-verbraucherpreisindex/, abgerufen am 29.02.2024. Industriekunden: https://de.statista.com/statistik/daten/studie/252029/umfrage/industriestrompreise-inkl-stromsteuer-in-deutschland/, abgerufen am 29.02.2024.

3 https://de.statista.com/statistik/daten/studie/155532/umfrage/versorgerwechsel-der-haushalte-in-der-stromversorgung-seit-2005/, abgerufen am 29.02.2024.

Polarisierung (zunehmend Gering- und Hochqualifizierte, weniger Mittelqualifizierte) (vgl. Hirsch-Kreinsen 2020: 50ff.). Theoretisch können Softwaregestaltende zu allen drei Szenarien beitragen. Sie automatisieren Arbeit durch Softwaregestaltung und erhöhen das Qualifikationsniveau und den Anteil Hoch- und Mittelqualifizierter. Das ist in den Fallstudien allerdings nur zum Teil der Fall. So ersetzt die individuell gestaltete Software bei INTERN1 z.B. die Arbeit der Meister:innen in der Instandhaltung (Substitution). Anwendende wie Monteur:innen benötigen allerdings weiterhin ihre bisherigen Qualifikationen. In Anwendungsbereichen mit einem hohen Anteil an Datenverarbeitung und Automatisierung führt die Softwaregestaltung dazu, dass die Beschäftigten nur noch die komplizierteren, nicht automatisierbaren Restfälle bearbeiten und mehr interdisziplinär tätig sind, wofür exemplarisch die Rolle der Key User:innen steht (Upgrading). Ob ein zunehmender Bedarf an höherqualifizierten Softwaregestaltenden dazu führt, dass es weniger Beschäftigte mit mittlerer Qualifikation gibt (Polarisierung), zeigt sich nicht eindeutig. So erledigen in der Fallstudie STARTUP nur zwei bis drei Minijob-Beschäftigte einfache Arbeit, indem sie automatisierte Prozesse prüfen und ergänzen. Die restlichen ca. 18–22 Beschäftigten sind in der Mehrzahl mittel bis hoch qualifiziert und nahezu allesamt Akademiker:innen. Rechnet man jene Anwendenden, welche komplizierte Restfälle bearbeiten und internen Support für die Softwaregestaltung komplexer Softwarepakete wie ERP-Systeme leisten (Key User:innen), zu den Mittelqualifizierten, dann würden in solchen EVU wenige Geringqualifizierte übrigbleiben und die Mehrzahl wären mittel bis hoch qualifizierte Anwendende, Gestaltende und Programmierende.

In den Fallstudien fällt zudem auf, dass die Anwendenden der EVU vom Wandel der Software durch den Betrieb, für den sie arbeiten, geschützt sind. Selbst wenn Betriebe Anwendungsplattformen einsetzen, werden deren Anwendende nicht gleich zu Gig-Arbeitenden. Eindeutige Folgen der Softwaregestaltung für sämtliche Anwendende wie z.B. mehr Routinearbeit, eine stärkere soziale Isolation oder eine Dequalifizierung haben die Fallstudien nicht gezeigt. Für eine fundierte Aussage dazu wären quantitative Daten notwendig.

9.4. Methodische Grenzen und weiterführende Fragestellungen

Was hat das alles mit Max Weber zu tun, der am Anfang der Untersuchung zitiert wurde? Das wäre en détail noch herauszuarbeiten. Doch weder durch den Begriffsapparat von Weber noch von jenem z.B. von Giddens' Strukturationstheorie wollte sich die vorliegende Arbeit einschränken und überfordern lassen. Es war das primäre Ziel, eine eigene soziotechnische Begriffsbildung zu betreiben, welche den Besonderheiten der Softwaregestaltung gerecht wird. Mit welchen bestehenden theoretischen Ansätzen die hier entwickelten Konzepte am besten kompatibel sind, wäre eine noch zu leistende Aufgabe.

Das Methoden-Kapitel hat bereits weitere Grenzen der vorliegenden Untersuchung genannt. Darunter waren auch die allgemeinen Grenzen qualitativer Untersuchungen. Um die Grenze der Verallgemeinerung zu überwinden, wäre die Erforschung anderer Branchen naheliegend. Ziegler hat ähnliche Phänomene wie die in der vorliegenden Arbeit in einem großen Industriekonzern und dessen Bemühungen, im IoT-Bereich ein Geschäftsfeld aufzubauen, untersucht. Auch dort gibt es crossfunktionale Teams, wel-

che die Linienorganisation ersetzen (vgl. Ziegler: 264f.). Auch dort ist die Herausforderung, agile Methoden und standardisierte Prozesse in einer Organisation zu vereinen (vgl. ebd.: 266). Und auch dort wird reorganisiert, um »domänenübergreifende[…] Organisationseinheiten« (ebd.: 282) aufzubauen. Lässt sich das hier vorliegende Analyseschema also verallgemeinern auf sämtliche Formen und Orte der Softwaregestaltung? Wie und wer gestaltet Software für die Versicherungswirtschaft oder für Banken und gibt es dort auch vielfältige Kooperationen? Gestalten die softwareanwendenden Organisationen in anderen Branchen mehr oder weniger individuell? Sind sie mehr oder weniger abhängig in der soziotechnischen Arbeitsgestaltung der Softwareanwendung? Wie sieht eine größere Organisation in der Logistik, dem Online-Handel oder der Verwaltung aus, in der von Anfang an der Primat der Softwareentwicklung gilt? Wie unterscheidet sich z.B. die industriespezifische Softwareentwicklung zwischen Amazon und Zalando? Zudem wirkt der Sprung vom Arbeitsprozess der Softwaregestaltung zur industriespezifischen Anwendungsplattform noch groß. Die gesamte Organisation hinter der Anwendungsplattform hat z.B. die Fallstudie PAKET nicht untersucht. Es gibt sicherlich noch mehr Faktoren als die Softwaregestaltung, die maßgeblich dafür sind, dass eine Anwendungsplattform entsteht.

Zudem konnte die Frage nicht ausführlich beantwortet werden, welchen Unterschied es für die Tätigkeit der einzelnen Anwendenden macht, wenn sie mit einer individuell gestalteten Software arbeiten. Dafür müssten deutlich mehr Anwendende befragt werden und es wäre bspw. die Untersuchung unterschiedlicher Formen der Softwaregestaltung für den gleichen Anwendungsbereich notwendig. Es gibt wahrscheinlich auch Unterschiede zwischen Beschäftigtengruppen, was das IT-Budget für Softwaregestaltung angeht. Können Beschäftigte aus dem Energiebörsenhandel mehr Änderungen mit dem Hinweis einfordern, dass sie dadurch bessere Ergebnisse erzielen? Fällt dies Beschäftigten aus dem Kundenservice schwerer, weil sie ihre Tätigkeiten formalisierter und standardisierter erledigen müssen? Auch könnten Fragen von Partizipation vertieft werden. Es gibt eine seit langem geführte Diskussion dazu (vgl. Becker/Brinkmann 2017). Unterschiede zu Partizipationstypen wie jenem des »democratic Taylorism« (Adler 1995) könnten herausgearbeitet werden.

Es ist noch offen, wie sich das softwaretechnische, interdisziplinäre Wissen nun genau zusammensetzt: Der Unterscheidung von Hohlmann (2007) von Organisations-, Prozess- und Technologiewissen wurde in dieser Arbeit nicht gefolgt und anders als Hohlmann spricht diese Arbeit nicht von Integrationswissen, sondern von interdisziplinärem Wissen. Wie sieht aber dann die Zusammensetzung aus? Welches Wissen ist für wen in welcher Organisation nun zentral für die Softwaregestaltung: über den Anwendungsbereich, über ein einzelnes EVU, die Regulierung, das ERP-Paket und sein Customizing, über Programmierung, Methoden und Kompetenzen der Softwaregestaltung, spezifisches Wissen einzelner Anwendender? Wo spielt welches implizite Wissen eine Rolle, in welcher Phase der Softwareentwicklung und wo ist explizites Wissen über Regulierung oder Geschäftsprozesse wichtiger? Wie wichtig sind betriebliche Qualifikationen? Einerseits sind betriebsspezifische Qualifikationen nützlich (langfristige Zusammenarbeit mit externen Entwickelnden, langfristige Zusammenarbeit mit IT-DL, lange Betriebszugehörigkeit von Anwendenden, Programmierenden

oder Gestaltenden). Andererseits wechseln die EVU trotzdem den IT-DL und es gibt Personalfluktuation in den Organisationen.

Zudem bietet die vorliegende Forschungsarbeit vielfältige Anknüpfungspunkte zu anderen Forschungsgebieten wie zur Labour Process Theory. Dort gibt es zwar Forschungsarbeiten zur Softwareentwicklung, aber keine zum Arbeitsprozess der Softwaregestaltung. In der Organisationssoziologie sieht Carlile (2004) zur Überwindung von Wissensgrenzen ein gemeinsames Verständnis, eine gemeinsame Sprache, gute Beziehungen und gemeinsame Interessen als wichtig an. Welchen Beitrag kann die vorliegende Untersuchung zu diesem Forschungsfeld der Wissensgrenzen leisten? Es gibt Anknüpfungspunkte zum Thema Innovationen in Organisationen, zum Requirements Engineering bzw. Anforderungsmanagement in der Informatik oder zu neuen beruflichen Identitäten und Professionalisierungen in der Berufsforschung: Qualifikation, Karrierewege, Arbeitsweise, Gehalt und Tätigkeiten der Softwaregestaltenden. Eine tiefergehende Auseinandersetzung mit der Techniksoziologie erscheint notwendig, ob mit den Arbeiten von Pollock et al. (2007) zur »generification work« bei Standardsoftwarepaketen oder zur digitalen Transformation u.a. durch Plattformen (vgl. Dolata/Schrape 2023).

Es fehlt an Forschung zu Wertschöpfungsketten von Software in der Soziologie oder der Wirtschaftsgeografie. Was ist der Unterschied zu anderen Wertschöpfungsketten? Dafür sind andere Konzepte notwendig wie jene der Global Production Networks (Henderson et al. 2002, Coe et al. 2008) oder Global Value Chains (vgl. Gereffi et al. 2005). Denn Software ist nichts, was wie in der Automobilwirtschaft die Fahrzeughersteller (OEM) weiterverarbeiten oder montieren. Softwaregestaltung entscheidet über die Organisations- und Arbeitsgestaltung. Der Wechsel der zuliefernden Softwarefirma hat entsprechend andere Folgen, als wenn ein Unternehmen ein Teil einer Maschine von einem anderen Zulieferer bezieht. Die Untersuchung konnte auch nicht der Frage nachgehen, warum EVU sich für welche Möglichkeiten zwischen Standard- und Individualsoftwaregestaltung entscheiden. Spielt dabei das organisationale Feld der Energiewirtschaft (vgl. Bleicher 2006) eine Rolle, in dem eine Form der Isomorphie wirkt (vgl. DiMaggio/Powell 1983)?

Es wäre theoretisch zu klären, wie sich die Begriffe von Macht, Herrschaft, Koordination, Steuerung, Governance, Kontrolle und Transformation der Arbeitskraft analytisch trennscharf auseinanderhalten lassen. Dazu gehört, Ansätze wie die Unterscheidung zwischen technischer, bürokratischer und direkter Kontrolle (vgl. Edwards 1979), Input-, Verhaltens-, Ergebnis-, Clan- und Selbstkontrolle (vgl. Wiener et al. 2016), normative Kontrolle durch Professionen und Firmenkultur (vgl. Kunda 1992, Fleming/Sturdy 2011) und neuere Ansätze des algorithmischen Managements zu reflektieren und voneinander theoretisch abzugrenzen. Inwiefern sind die Ergebnisse dieser Arbeit anschlussfähig an Arbeiten von Vincent August zum Thema Kybernetik und Herrschaft durch Netzwerke (vgl. August 2019, August 2021)? Ist eine neue Theorie der Macht und Herrschaft notwendig, wenn Technik, Organisation und Arbeit so eng verzahnt sind, sich wechselseitig verändern und auf mehrere Organisationen verteilt sind?

Literatur

A. T. Kearney, BDEW, & IMP³rove. (2018). *Digital@EVU – Wo steht die deutsche Energiewirtschaft?* https://www.bdew.de/media/documents/201802_Paper-Digital-EVU.pdf. Zugegriffen: 24. Mai 2023.

A. T. Kearney, BDEW, & IMP³rove. (2019). *Digital@EVU – Wo steht die deutsche Energiewirtschaft?* https://www.bdew.de/media/documents/Pub_20190401_Studie-Digital-EVU_WDcoFda.pdf. Zugegriffen: 24. Mai 2023.

Adler, P. (1995). ›Democratic Taylorism‹: The Toyota production system at NUMMI. In S. Babson (Hg.), *Lean Work: Empowerment and Exploitation in the Global Auto Industry* (S. 207–219). Detroit, MI: Wayne State University Press.

Adler, P. S. & Borys, B. (1996). Two types of bureaucracy: Enabling and coercive. *Administrative Science Quarterly*, *41*(1), 61–89.

Adler, P. S. & Heckscher, C. (2006). Towards collaborative community. In P. S. Adler & C. Heckscher (Hg.), *The Firm as a Collaborative Community: Reconstructing Trust in the Knowledge Economy* (S. 11–105). Oxford: Oxford University Press.

Ahrne, G. & Brunsson, N. (2011). Organization outside organizations: The significance of partial organization. *Organization*, *18*(1), 83–104.

Alsanoosy, T., Spichkova, M. & Harland, J. (2020). Cultural influence on requirements engineering activities: A systematic literature review and analysis. *Requirements Engineering*, *25*, 339–362.

Alvarez, R. (2002). Confessions of an information worker: a critical analysis of information requirements discourse. *Information and Organization*, *12*(2), 85–107.

Ambrosius, G. (2012). Geschichte der Stadtwerke. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 35–51). Baden-Baden: Nomos.

Andelfinger, U. (1997). *Diskursive Anforderungsanalyse: Ein Beitrag zum Reduktionsproblem bei Systementwicklungen in der Informatik*. Frankfurt a.M.: Peter Lang.

Andreessen, M. (2011, August 20). Why Software Is Eating the World. *Wall Street Journal*. https://online.wsj.com/article/SB10001424053111903480904576512250915629460.html. Zugegriffen: 17. Oktober 2022.

Andrews, C. K., Lair, C. D. & Landry, B. (2005). The labor process in software startups: Production on a virtual assembly line? In R. Barrett (Hg.), *Management Labour Pro-*

cess and Software Development: Reality Bytes (Bd. 13, S. 45–75). London; New York, NY: Routledge.

Apitzsch, B. (2006). *Unternehmensnetzwerke und soziale Einbettung: Begriffliche Bestimmungen, Funktionen und Entstehungsbedingungen*. Duisburg: Duisburger Beiträge zur soziologischen Forschung. https://www.ssoar.info/ssoar/handle/document/11455. Zugegriffen: 24. Mai 2023.

Apitzsch, B. (2010). *Flexible Beschäftigung, neue Abhängigkeiten. Projektarbeitsmärkte und ihre Auswirkungen auf Lebensverläufe*. Frankfurt a.M.: Campus.

Apitzsch, B., Buss, K.-P., Kuhlmann, M., Weißmann, M. & Wolf, H. (2021). Arbeit in und an Digitalisierungen. Ein Resümee als Einführung. In K.-P. Buss, M. Kuhlmann, M. Weißmann, H. Wolf, & B. Apitzsch (Hg.), *Digitalisierung und Arbeit: Triebkräfte – Arbeitsfolgen – Regulierung* (S. 9–39). Frankfurt; New York: Campus Verlag.

Armstrong, P. (1985). Changing management control strategies: The role of competition between accountancy and their organisational professions. *Accounting, Organizations and Society*, *10*(2), 129–148.

August, V. (2019). Von ›Unregierbarkeit‹ zu Governance: Neoliberale, teleologische und technologische Staatskritik. In A. Cavuldak (Hg.), *Die Grammatik der Demokratie. Das Staatsverständnis von Peter Graf Kielmannsegg* (S. 287–312). Baden-Baden: Nomos.

August, V. (2021). *Technologisches Regieren: Der Aufstieg des Netzwerk-Denkens in der Krise der Moderne. Foucault, Luhmann und die Kybernetik*. Bielefeld: transcript.

Baethge, M. (1996). Zwischen Computer und Kunden – Rationalisierung und neue Arbeitskonzepte in den Dienstleistungen. In H.-J. Braczyk, H.-D. Ganter, & R. Seltz (Hg.), *Neue Organisationsformen in Dienstleistung und Verwaltung* (S. 15–28). Stuttgart: Kohlhammer.

Bähr, J. & Erker, P. (2017). *NetzWerke: Die Geschichte der Stadtwerke München*. München: Piper.

Barrett, M. & Oborn, E. (2010). Boundary object use in cross-cultural software development teams. *Human Relations*, *63*(8), 1199–1221.

Barrett, R. (2005). Managing the software development labour process: Direct control, time and technical autonomy. In R. Barrett (Hg.), *Management Labour Process and Software Development: Reality Bytes* (S. 76–99). London; New York, NY: Routledge.

Baudach, T. (2018). *Organisation von Wissensarbeit: Entwicklung eines gedanklichen Bezugsrahmens vor dem Hintergrund von Zukunftsannahmen sowie theoretischen Bezügen zur Arbeits- und Organisationsforschung*. Baden-Baden: Nomos.

Baukrowitz, A. (2006). Einführung. In A. Baukrowitz, T. Berker, S. Pfeiffer, R. Schmiede, & M. Will (Hg.), *Informatisierung der Arbeit – Gesellschaft im Umbruch* (S. 81–87). Berlin: Edition Sigma.

Baukrowitz, A., Berker, T., Boes, A., Pfeiffer, S., Schmiede, R. & Will, M. (Hg.). (2006). *Informatisierung der Arbeit – Gesellschaft im Umbruch*. Berlin: Edition Sigma.

Baukrowitz, A., Boes, A. & Eckhardt, B. (1994). *Software als Arbeit gestalten: konzeptionelle Neuorientierung der Aus- und Weiterbildung von Computerspezialisten*. Opladen: Westdeutscher Verlag.

BDEW (Hg.). (2016). Die digitale Energiewirtschaft. Agenda für Unternehmen und Politik. https://www.bdew.de/media/documents/Pub_20160501_Digitale-Energiewirtschaft.pdf. Zugegriffen: 24. Mai 2023.

BDEW (Hg.). (2019). Zahl der Unternehmen in den einzelnen Marktbereichen. https://www.bdew.de/media/documents/Marktteilnehmer_Energie_aktuell_online_o_halbjaehrlich_Ki_05112019.pdf. Zugegriffen: 24. Mai 2023.

Bechky, B. A. (2006). Gaffers, gofers, and grips: Role-based coordination in temporary organizations. *Organization Science*, *17*(1), 3–21.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., et al. (2001). The Agile Manifesto. *Manifesto for Agile Software Development*. http://agilemanifesto.org/. Zugegriffen: 20. Juli 2022.

Becker, P. (2010). *Aufstieg und Krise der deutschen Stromkonzerne. Zugleich ein Beitrag zur Entwicklung des Energierechts*. Bochum: Ponte Press.

Becker, K. & Brinkmann, U. (2017). Partizipation. In H. Hirsch-Kreinsen & H. Minssen (Hg.), *Lexikon der Arbeits- und Industriesoziologie* (2. Auflage, S. 254–258). Baden-Baden: Nomos.

Berlo, K., Herr, C., Wagner, O. & Companie, M. (2018). *Explorative Untersuchung zu Erfolgspotentialen bei neugegründeten Stadtwerken: Eine Sondierungsstudie zur kommunalen Energieversorgung. Ergebnisse einer Befragung bei neugegründeten Stadtwerken im Energiebereich* (No. 16). Wuppertal: Wuppertal Institut für Klima, Umwelt, Energie.

Bessen, J. (2022). *The New Goliaths: How Corporations Use Software to Dominate Industries, Kill Innovation, and Undermine Regulation*. New Haven, CT; London: Yale University Press.

Bjørn-Andersen, N. & Raymond, B. (2014). The impact of IT over five decades – Towards the ambient organization. *Applied ergonomics*, *45*(2), 188–197.

Bläsche, A. & Lappe, L. (2006). Veränderung der Arbeitsprozesse bei der Anpassung und Implementierung betriebswirtschaftlicher Standardlösungen. In A. Baukrowitz, T. Berker, A. Boes, S. Pfeiffer, R. Schmiede, & M. Will-Zocholl (Hg.), *Informatisierung der Arbeit – Gesellschaft im Umbruch* (S. 302–308). Baden-Baden: Nomos.

Bleicher, A. (2006). *Die Institutionalisierung eines organisationalen Feldes – das Beispiel der Elektrizitätswirtschaft*. Universität Hannover, Hannover. Abgerufen von https://opus4.kobv.de/opus4-btu/frontdoor/deliver/index/docId/322/file/Bleicher_Diss.pdf. Zugegriffen: 24. Mai 2023.

Boehm, B. (2006). A view of 20th and 21st century software engineering. In *Proceedings of the 28th international conference on Software engineering* (S. 12–29). Association for Computing Machinery. https://doi.org/10.1145/1134285.1134288. Zugegriffen: 24. Mai 2023

Boes, A. & Kämpf, T. (2017). Informations- und Wissensarbeit. In H. Hirsch-Kreinsen & H. Minssen (Hg.), *Lexikon der Arbeits-und Industriesoziologie* (2. Auflage, S. 184–187). Baden-Baden: Nomos.

Boes, A., Kämpf, T., Gül, K., Langes, B., Lühr, T., Marrs, K. & Ziegler, A. (2016). Digitalisierung und »Wissensarbeit«: Der Informationsraum als Fundament der Arbeitswelt der Zukunft. *Aus Politik und Zeitgeschichte*, *66*(18), 32–39.

Boes, A., Kämpf, T., Langes, B. & Lühr, T. (2018). *»Lean« und »agil« im Büro: Neue Organisationskonzepte in der digitalen Transformation und ihre Folgen für die Angestellten*. Bielefeld: transcript.

Boes, A., Kämpf, T. & Ziegler, A. (2020). Arbeit im Informationsraum – Informatisierung als Perspektive für ein soziologisches Verständnis der digitalen Transformation. In S. Maasen & J.-H. Passoth (Hg.), *Soziologie des Digitalen – Digitale Soziologie?* (Bd. Sonderband 23, S. 305–325). Baden-Baden: Nomos.

Boes, A. & Pfeiffer, S. (2006). Informatisierung der Arbeit – Gesellschaft im Umbruch. Eine Einführung. In A. Baukrowitz, T. Berker, S. Pfeiffer, R. Schmiede, & M. Will (Hg.), *Informatisierung der Arbeit – Gesellschaft im Umbruch*. Berlin: Edition Sigma.

Bogner, A. & Menz, W. (2002). Expertenwissen und Forschungspraxis: Die modernisierungstheoretische und die methodische Debatte um die Experten. In A. Bogner, B. Littig, & W. Menz (Hg.), *Das Experteninterview. Theorie, Methode, Anwendung* (S. 7–29). Wiesbaden: VS Verlag für Sozialwissenschaften.

Böhle, F. (2010). Arbeit als Handeln. In F. Böhle, G. G. Voß, & G. Wachtler (Hg.), *Handbuch Arbeitssoziologie* (S. 151–176). Wiesbaden: VS Verlag für Sozialwissenschaften.

Bolici, F., Howison, J. & Crowston, K. (2009). Coordination without discussion? Socio-technical congruence and stigmergy in free and open source software projects. Gehalten auf der International Conference on Software Engineering, Vancouver BC. https://crowston.syr.edu/sites/crowston.syr.edu/files/Coordination%20without%20discussion%3F%20Socio-technical%20congruence.pdf. Zugegriffen: 24. Mai 2023.

Bolici, F., Howison, J. & Crowston, K. (2016). Stigmergic coordination in FLOSS development teams: Integrating explicit and implicit mechanisms. *Cognitive Systems Research*, *38*, 14–22.

Boltanski, L. & Chiapello, È. (2003). *Der neue Geist des Kapitalismus*. Konstanz: UVK Verlagsgesellschaft mbH.

Bolte, A. (2017a). Tätigkeit und Arbeitsprozess. In F. Böhle (Hg.), *Arbeit als Subjektivierendes Handeln. Handlungsfähigkeit bei Unwägbarkeiten und Ungewissheit* (S. 473–484). Wiesbaden: Springer VS.

Bolte, A. (2017b). Unwägbarkeiten. In F. Böhle (Hg.), *Arbeit als Subjektivierendes Handeln. Handlungsfähigkeit bei Unwägbarkeiten und Ungewissheit* (S. 487–492). Wiesbaden: Springer VS.

Bolte, A. & Porschen, S. (2007). *Die Organisation des Informellen: Modelle zur Organisation von Kooperation im Arbeitsalltag*. Wiesbaden: Springer VS.

Bontrup, H.-J. & Marquardt, R.-M. (2010). *Kritisches Handbuch der deutschen Elektrizitätswirtschaft: Branchenentwicklung, Unternehmensstrategien, Arbeitsbeziehungen*. Berlin: Edition Sigma.

Boonstra, A. (2006). Interpreting an ERP-implementation project from a stakeholder perspective. *International Journal of Project Management*, *24*(1), 38–52.

Boonstra, A. & de Vries, J. (2015). Information system conflicts: Causes and types. *International Journal of Information Systems and Project Management*, *3*(4), 5–20.

Boudreau, M.-C. & Robey, D. (2005). Enacting integrated information technology: A human agency perspective. *Organization Science*, *16*(1), 3–18.

Bradach, J. L. & Eccles, R. G. (1989). Price, authority, and trust: From ideal types to plural forms. *Annual Review of Sociology*, *15*(1), 97–118.

Bräunig, D. (2012). Entflechtung von Stadtwerken als Konsequenz des europäischen Energiebinnenmarktes. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 419–437). Baden-Baden: Nomos.

Brede, H. (2012). Führung und Marketing von Stadtwerken. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 305–319). Baden-Baden: Nomos.

Breuer, F., Dieris, B. & Lettau, A. (2009). *Reflexive Grounded Theory: Eine Einführung für die Forschungspraxis*. Wiesbaden: VS Verlag für Sozialwissenschaften. Zugegriffen: 24. Mai 2023.

Brinkmann, U. & Dörre, K. (2006). Wissensarbeit und Kontrolle im neuen Marktregime. In A. Baukrowitz, T. Berker, S. Pfeiffer, R. Schmiede, & M. Will (Hg.), *Informatisierung der Arbeit – Gesellschaft im Umbruch* (S. 132–144). Berlin: Edition Sigma.

Brödner, P. (2014). Durch »Information« desinformiert? Zur Kritik des Paradigmas der Informationsverarbeitung. *AIS-Studien*, *7*(1), 42–59.

Brown, J. S. & Duguid, P. (2001). Knowledge and organization: A social-practice perspective. *Organization Science*, *12*(2), 198–213.

Brusoni, S., Prencipe, A. & Pavitt, K. (2001). Knowledge specialization, organizational coupling, and the boundaries of the firm: Why do firms know more than they make? *Administrative Science Quarterly*, *46*(4), 597–621.

Bundesstelle für Energieeffizienz (BfEE) (Hg.). (2019). Empirische Untersuchung des Marktes für Energiedienstleistungen, Energieaudits und andere Energieeffizienzmaßnahmen im Jahr 2018. Endbericht 2019 – BfEE 17/2017, Eschborn. https://www.bfee-online.de/SharedDocs/Downloads/BfEE/DE/Energiedienstleistungen/marktерhebung2019.pdf;jsessionid=55257773B7302115604EF5B4241D21A1.intranet662?__blob=publicationFile&v=2. Zugegriffen: 24. Mai 2023.

Butollo, F., Feuerstein, P. & Krzywdzinski, M. (2021). Was zeichnet die digitale Transformation der Arbeitswelt aus? Ein Deutungsangebot jenseits von Großtheorien und disparater Empirie. *AIS-Studien*, *14*(2), 27–44.

Carlile, P. R. (2004). Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization Science*, *15*(5), 555–568.

Carmel, E. & Sawyer, S. (1998). Packaged software development teams: What makes them different? *Information Technology & People*, *11*, 7–19.

Carugati, A., Fernández, W., Mola, L. & Rossignoli, C. (2018). My choice, your problem? Mandating IT use in large organisational networks. *Information Systems Journal*, *28*(1), 6–47.

Cascio, W. F. & Montealegre, R. (2016). How technology is changing work and organizations. *Annual Review of Organizational Psychology and Organizational Behavior*, *3*, 349–375.

Castell, M. (2001). *Das Informationszeitalter: Der Aufstieg der Netzwerkgesellschaft (Bd. 1)*. Leske und Budrich: Opladen.

Chan, Y. E. & Reich, B. H. (2007). IT alignment: What have we learned? *Journal of Information Technology*, *22*(4), 297–315.

Chandler, A. D. (1990). *Scale and Scope. The Dynamics of Industrial Capitalism*. Cambridge, Mass.: Belknap.

Charmaz, K. (2014). *Constructing Grounded Theory* (2nd Edition.). London: Sage.

Coe, N. M., Dicken, P. & Hess, M. (2008). Global production networks: Realizing the potential. *Journal of Economic Geography*, *8*(3), 271–295.

Colfer, L. J. & Baldwin, C. Y. (2016). The mirroring hypothesis: theory, evidence, and exceptions. *Industrial and Corporate Change*, *25*(5), 709–738.

Collins, C. S. & Stockton, C. M. (2018). The central role of theory in qualitative research. *International Journal of Qualitative Methods*, *17*(1).

Conrad, L. (2017). *Organisation im soziotechnischen Gemenge: Mediale Umschichtungen durch die Einführung von SAP*. Bielefeld: transcript.

Conway, M. E. (1968). How do committees invent. *Datamation*, *14*(4), 28–31.

Corvera Charaf, M., Rosenkranz, C. & Holten, R. (2013). The emergence of shared understanding: applying functional pragmatics to study the requirements development process. *Information Systems Journal*, *23*(2), 115–135.

Cullmann, A., Nieswand, M., Seifert, S. & Stiel, C. (2016). Trend zur (Re-)Kommunalisierung in der Energieversorgung: Ein Mythos? *DIW-Wochenbericht*, *83*(20), 441–447.

Crowston, K. & Myers, M. D. (2004). Information technology and the transformation of industries: Three research perspectives. *The Journal of Strategic Information Systems*, *13*(1), 5–28.

D'Adderio, L. (2003). Configuring software, reconfiguring memories: the influence of integrated systems on the reproduction of knowledge and routines. *Industrial and Corporate Change*, *12*(2), 321–350.

Darr, A. (2019). Automatons, sales-floor control and the constitution of authority. *Human Relations*, *72*(5), 889–909.

Degele, N. (2000). *Informiertes Wissen: eine Wissenssoziologie der computerisierten Gesellschaft*. Frankfurt a.M.; New York: Campus.

Dery, K., Grant, D., Harley, B. & Wright, C. (2006). Work, organisation and enterprise resource planning systems: an alternative research agenda. *New Technology, Work and Employment*, *21*(3), 199–214.

Deterding, N. M. & Waters, M. C. (2021). Flexible coding of in-depth interviews: A twenty-first-century approach. *Sociological Methods & Research*, *50*(2), 708–739.

DGS, & BDS (Hg.). (2017, Juni 10). Ethik-Kodex der Deutschen Gesellschaft für Soziologie (DGS) und des Berufsverbandes Deutscher Soziologinnen und Soziologen (BDS). https://soziologie.de/fileadmin/user_upload/dokumente/Ethik-Kodex_2017-06-10.pdf. Zugegriffen: 4. Mai 2023

DiMaggio, P. J. & Powell, W. W. (1983). The iron cage revisited: Institutional isomorphism and collective rationality in organizational fields. *American Sociological Review*, 147–160.

Dolata, U. & Schrape, J.-F. (2023). Politische Ökonomie und Regulierung digitaler Plattformen. In T. Carstensen, S. Schaupp, & S. Sevignani (Hg.), *Theorien des digitalen Kapitalismus. Arbeit und Ökonomie, Politik und Subjekt*. Berlin: Suhrkamp.

Doleski, O. D. (2016). *Utility 4.0: Transformation vom Versorgungs- zum digitalen Energiedienstleistungsunternehmen*. Wiesbaden: Springer Vieweg.

Dörre, K. (2001). Das deutsche Produktionsmodell unter dem Druck des Shareholder Value. *KZfSS Kölner Zeitschrift für Soziologie und Sozialpsychologie*, *53*(4), 675–704.

Edeling, T., Stölting, E. & Wagner, D. (2004). *Öffentliche Unternehmen zwischen Privatwirtschaft und öffentlicher Verwaltung. Eine empirische Studie im Feld kommunaler Versorgungsunternehmen*. Opladen: VS Verlag für Sozialwissenschaften.

Edwards, R. (1981). *Herrschaft im modernen Produktionsprozess*. Frankfurt a.M.; New York: Campus.

Elbanna, A. R. (2007). Implementing an integrated system in a socially dis-integrated enterprise: A critical view of ERP enabled integration. *Information Technology & People*, *20*(2), 121–139.

Ellguth, P. & Kohaut, S. (2017). Tarifbindung und betriebliche Interessenvertretung: Ergebnisse aus dem IAB-Betriebspanel 2016. *WSI-Mitteilungen*, *70*(4), 278–286.

Esposito, E. (1993). Der Computer als Medium und Maschine. *Zeitschrift für Soziologie*, *22*(5), 338–354.

Faraj, S. & Sproull, L. (2000). Coordinating expertise in software development teams. *Management Science*, *46*(12), 1554–1568.

Felin, T., Zenger, T. R. & Tomsik, J. (2009). The knowledge economy: emerging organizational forms, missing microfoundations, and key considerations for managing human capital. *Human Resource Management: Published in Cooperation with the School of Business Administration, The University of Michigan and in alliance with the Society of Human Resources Management*, *48*(4), 555–570.

Feuerstein, P. (2012). *Viele Wege führen nach Indien: Reorganisation von Arbeit im Zuge der Internationalisierung der IT-Industrie*. Göttingen: Universitätsverlag Göttingen.

Fischbach, R. (2016). Weshalb sind Softwareprojekte schwierig? In F. Fuchs-Kittowski & W. Kriesel (Hg.), *Informatik und Gesellschaft. Festschrift zum 80. Geburtstag von Klaus Fuchs-Kittowski* (S. 393–402). Berlin: Peter Lang.

Flecker, J., Haidinger, B. & Schönauer, A. (2013). Divide and serve: The labour process in service value chains and networks. *Competition & Change*, *17*(1), 6–23.

Flecker, J. & Hermann, C. (2011). The liberalization of public services: Company reactions and consequences for employment and working conditions. *Economic and Industrial Democracy*, *32*(3), 523–544.

Flecker, J. & Holtgrewe, U. (2008). Überbetriebliche Arbeitsteilung: Auslagerung von Unternehmensfunktionen und die Folgen für Arbeit und Beschäftigung. *Wirtschaft und Gesellschaft*, *34*(3), 307–336.

Flecker, J. & Meil, P. (2010). Organisational restructuring and emerging service value chains: Implications for work and employment. *Work, Employment & Society*, *24*(4), 680–698.

Fleming, P. & Sturdy, A. (2011). ›Being yourself‹ in the electronic sweatshop: New forms of normative control. *Human Relations*, *64*(2), 177–200.

Floyd, C. (1992). Software Development as Reality Construction. In C. Floyd, H. Züllighoven, R. Budde, & R. Keil-Slawik (Hg.), *Software Development and Reality Construction* (S. 86–100). Berlin, Heidelberg: Springer Berlin Heidelberg.

Foerderer, J., Kude, T., Schuetz, S. W. & Heinzl, A. (2019). Knowledge boundaries in enterprise software platform development: Antecedents and consequences for platform governance. *Information Systems Journal*, *29*(1), 119–144.

Ford, R. C. & Randolph, W. A. (1992). Cross-functional structures: A review and integration of matrix organization and project management. *Journal of Management*, *18*(2), 267–294.

Frederick, J. & Zierau, T. (2011). *SAP for Utilities: Das umfassende Handbuch für Energieversorger*. Bonn: Galileo Press.

Friedman, A. (1977). Responsible autonomy versus direct control over the labour process. *Capital & Class*, *1*(1), 43–57.

Friedman, A. L. & Cornford, D. S. (1993). *Computer Systems Development: History, Organization and Implementation* (2. Aufage.). Chichester: Wiley.

Friese, S. (2016). Grounded Theory computergestützt und umgesetzt mit ATLAS. ti. In C. Equit & C. Hohage (Hg.), *Handbuch Grounded Theory: Von der Methodologie zur Forschungspraxis* (S. 483–507). Weinheim: Beltz Juventa.

Fromme, H. (2022, April 22). Allianz scheitert mit IT-Strategie. *SZ*, S. 16. München.

Funken, C. (2001). *Modellierung der Welt. Wissenssoziologische Studien zur Software-Entwicklung*. Opladen: Leske + Budrich.

Funken, C., Stoll, A. & Hörlin, S. (2011). *Die Projektdarsteller: Karriere als Inszenierung: Paradoxien und Geschlechterfallen in der Wissensökonomie*. Wiesbaden: VS Verlag für Sozialwissenschaften.

Gerber, C. & Krzywdzinski, M. (2019). Entgrenzung in der digitalen Onlinearbeit am Beispiel von Crowdwork. In H. Hanau & W. Matiaske (Hg.), *Entgrenzung von Arbeitsverhältnissen: Arbeitsrechtliche und sozialwissenschaftliche Perspektiven* (S. 25–48). Baden-Baden: Nomos.

Gereffi, G., Humphrey, J. & Sturgeon, T. (2005). The governance of global value chains. *Review of International Political Economy*, *12*(1), 78–104.

Gerst, D. (2006). *Von der direkten Kontrolle zur indirekten Steuerung: eine empirische Untersuchung der Arbeitsfolgen teilautonomer Gruppenarbeit*. München: Hampp.

Giddens, A. (1988). *Die Konstitution der Gesellschaft*. Frankfurt a.M.; New York: Campus.

Gläser, J. & Laudel, G. (2006). *Experteninterviews und qualitative Inhaltsanalyse als Instrument rekonstruierender Sozialforschung* (2., durchges. Auflage). Wiesbaden: VS Verlag für Sozialwissenschaften.

Gloger, B. (2009). *Scrum. Produkte zuverlässig und schnell entwickeln*. München: Carl Hanser.

Gottschalk, W. (2012). Strukturen und Organisation von Stadtwerken. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 53–73). Baden-Baden: Nomos.

Grant, D., Hall, R., Wailes, N. & Wright, C. (2006). The false promise of technological determinism: The case of enterprise resource planning systems. *New Technology, Work and Employment*, *21*(1), 2–15.

Gregory, R. W., Beck, R. & Keil, M. (2013). Control balancing in information systems development offshoring projects. *MIS Quarterly*, *73*(4), 1211–1232.

Griese, H. M. (2002). Rolle. In G. Endruweit & G. Trommsdorff (Hg.), *Wörterbuch der Soziologie* (2., völlig neubearb. u. erw. Auflage, S. 458–462). Stuttgart: Lucius und Lucius.

Grimshaw, D., Cooke, F.-L., Grugulis, I. & Vincent, S. (2002). New technology and changing organisational forms: implications for managerial control and skills. *New Technology, Work and Employment*, *17*(3), 186–203.

Guillemette, M. G. & Paré, G. (2012). Toward a new theory of the contribution of the IT function in organizations. *MIS Quarterly*, *36*(2), 529–551.

Hägler, M. (2022, Mai 13). Volkswagen ringt mit der Software. *SZ*, S. 14. München.

Hall, P. A. & Soskice, D. (2001). *Varieties of capitalism: the institutional foundations of comparative advantage*. Oxford: Oxford University Press.

Heckscher, C. (2015). From bureaucracy to networks. *S. Edgell, H. Gottfried, & E. Granter, The SAGE Handbook of the Sociology of Work and Employment*, 245–258.

Heidenreich, M., Kirch, B. & Mattes, J. (2008). Die organisatorische Einbettung von Informationstechnologien in einem globalen Entwicklungsprojekt. In C. Funken & I. Schulz-Schaeffer (Hg.), *Digitalisierung der Arbeitswelt. Zur Neuordnung formaler und informeller Prozesse in Unternehmen* (S. 193–219). Wiesbaden: VS Verlag für Sozialwissenschaften.

Heidling, E. (2018). Projektarbeit. In F. Böhle, G. G. Voß, & G. Wachtler (Hg.), *Handbuch Arbeitssoziologie. Band 2: Akteure und Institutionen* (S. 207–236). Wiesbaden: Springer VS.

Helfen, M. & Wirth, C. (2020). *Management von Arbeit in pluralen Netzwerkorganisationen: Trends, Deutungen und Handlungsoptionen*. Forschungsförderung Working Paper, Düsseldorf. https://www.boeckler.de/fpdf/HBS-007692/p_fofoe_WP_185_2020.pdf. Zugegriffen: 11. Juli 2023.

Hellermann, J. (2000). *Örtliche Daseinsvorsorge und gemeindliche Selbstverwaltung. Zum kommunalen Betätigungs- und Gestaltungsspielraum unter den Bedingungen europäischer und staatlicher Privatisierungs- und Deregulierungspolitik* (Bd. 54). Tübingen: Mohr Siebeck.

Henderson, J., Dicken, P., Hess, M., Coe, N. & Yeung, H. W.-C. (2002). Global production networks and the analysis of economic development. *Review of International Political Economy*, *9*(3), 436–464.

Herrigel, G. (2010). *Manufacturing possibilities: creative action and industrial recomposition in the United States, Germany, and Japan*. Oxford [u.a.]: Oxford University Press.

Herrmann, B. J. (2012). Kommunale Strom- und Gaswirtschaft im Zeitalter der Anreizregulierung. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 285–304). Baden-Baden: Nomos.

Hirschfeld, K. (2000). Auf verschlungenen Pfaden zum High-Tech-Produkt. Das Digitalfunkkonsortium – eine strategische Allianz und ihre Folgen. In F. Naschold, C. Dörrenbächer, H.-R. Meißner, & L. Renneke (Hg.), *Kooperieren über Grenzen* (S. 232–288). Heidelberg: Physica.

Hirschheim, R. & Klein, H. K. (2012). A glorious and not-so-short history of the information systems field. *Journal of the Association for Information Systems*, *13*(4), 5.

Hirsch-Kreinsen, H. (1995). Dezentralisierung: Unternehmen zwischen Stabilität und Desintegration. *Zeitschrift für Soziologie*, *24*(6), 422–435.

Hirsch-Kreinsen, H. (2002). Unternehmensnetzwerke – revisited. *Zeitschrift für Soziologie*, *31*(2), 106–124.

Hirsch-Kreinsen, H. (2020). *Digitale Transformation von Arbeit: Entwicklungstrends und Gestaltungsansätze*. Stuttgart: Kohlhammer.

Hohlmann, B. (2007). *Organisation SAP – soziale Auswirkungen technischer Systeme*. Aachen: Shaker.

HolacracyOne. (2023). Holacracy Constitution. Version 5.0. *Holacracy*. https://www.holacracy.org/constitution/5. Zugegriffen: 17. Mai 2023.

Holtgrewe, U. (2014). New new technologies: The future and the present of work in information and communication technology: The future and present of work in ICT. *New Technology, Work and Employment*, *29*(1), 9–24.

Höpner, M. (2003). *Wer beherrscht die Unternehmen? Shareholder Value, Managerherrschaft und Mitbestimmung in Deutschland*. Frankfurt a.M.; New York: Campus.

Howcroft, D. & Richardson, H. (2012). The back office goes global: Exploring connections and contradictions in shared service centres. *Work, Employment & Society*, *26*(1), 111–127.

Hvatum, L. & Kelly, A. (2005). What do I think about Conway's Law now? Conclusions of a EuroPLoP 2005 focus group. Gehalten auf der European Conference on Pattern Languages of Programs, Kloster Irsee. https://www.allankelly.net/static/presentations/EuroPLoP2005/ConwaysLawFocusGroup.pdf. Zugegriffen: 24. Mai 2023.

Illouz, E. (2008). Saving the Modern Soul: Therapy, Emotions, and the Culture of Self-Help. Berkeley u.a.: University of California Press.

Jacobsen, H., Blazejewski, F. & Graf, P. (2017). Der verdeckte Transformationsprozess der Energieversorger – Kollisionen von Rechtfertigungsordnungen. In S. Giacovelli (Hg.), *Die Energiewende aus wirtschaftssoziologischer Sicht* (S. 93–117). Wiesbaden: Springer VS.

Joerges, B., Czamiawska, B., Handelshögskolan, Göteborgs universitet, Gothenburg University, Gothenburg Research Institute (GRI), & School of Business, E., and Law. (1998). The Question of Technology, or How Organizations Inscribe the World. *Organization Studies*, *19*(3), 363–385.

Jürgens, U. (2007). Industrielle Entwicklungsdynamik und neue Formen der Governance. In U. Jürgens (Hg.), *Arbeitspolitik im Wandel. Entwicklung und Perspektiven der Arbeitspolitik*. (S. 117–152). Berlin: Edition Sigma.

Kalkowski, P. & Mickler, O. (2005). *Projektorganisation in der IT- und Medienbranche. Herausforderungen an Management, Mitarbeiter und Interessenvertretung* (Bd. 141). Düsseldorf: Edition der Hans-Böckler-Stiftung.

Kalkowski, P. & Mickler, O. (2015). *Kooperative Produktentwicklung: Fallstudien aus der Automobilindustrie, dem Maschinenbau und der IT-Industrie*. Baden-Baden: Edition Sigma.

Kaminski, A. (2012). Wie entsteht Software? Übersetzungen zwischen vertrautem Kontext und formalem System: Die heiße Zone des Requirements Engineerings. In C. Schilcher & M. Will-Zocholl (Hg.), *Arbeitswelten in Bewegung* (S. 85–123). Wiesbaden: VS Verlag für Sozialwissenschaften.

Kaniadakis, A. (2012). ERP implementation as a broad socio-economic phenomenon: The agora of techno-organisational change. *Information Technology & People*, *25*(3), 259–280.

Kelle, U. & Kluge, S. (2010). *Vom Einzelfall zum Typus: Fallvergleich und Fallkontrastierung in der qualitativen Sozialforschung* (2., überarb. Auflage). Wiesbaden: VS Verlag für Sozialwissenschaften.

Kern, H. & Schumann, M. (1984). *Das Ende der Arbeitsteilung? Rationalisierung in der industriellen Produktion: Bestandsaufnahme, Trendbestimmung*. München: Beck.

Kleemann, F. & Matuschek, I. (2008). Informalisierung als Komplement der Informatisierung von Arbeit. In C. Funken & I. Schulz-Schaeffer (Hg.), *Digitalisierung der Arbeitswelt. Zur Neuordnung formaler und informeller Prozesse in Unternehmen* (S. 43–67). Wiesbaden: VS Verlag für Sozialwissenschaften.

Klemm, M. & Liebold, R. (2017). Qualitative Interviews in der Organisationsforschung. In S. Liebig, W. Matiaske, & S. Rosenbohm (Hg.), *Handbuch Empirische Organisationsforschung* (S. 299–324). Wiesbaden: Springer Gabler.

Klischewski, R. (2009). Anarchie – ein Leitbild für die Informatik. In D. Krause & E. J. Simon (Hg.), *Im Widerspruch. Arno Rolf zum 65* (S. 75–88). Hamburg: Universität Hamburg. http://edoc.sub.uni-hamburg.de/informatik/volltexte/2009/130/. Zugegriffen: 25. April 2023.

Knoblauch, H. (1996). Arbeit als Interaktion: Informationsgesellschaft, Post-Fordismus und Kommunikationsarbeit. *Soziale Welt*, *47*(3), 344–362.

Ko, D.-G. & Kirsch, L. J. (2017). The hybrid IT project manager: One foot each in the IT and business domains. *International Journal of Project Management*, *35*(3), 307–319.

Kocyba, H. (1999). Wissensbasierte Selbststeuerung: Die Wissensgesellschaft als arbeitspolitisches Kontrollszenario. In W. Konrad & W. Schumm (Hg.), *Wissen und Arbeit. Neue Konturen von Wissensarbeit* (S. 92–119). Wiesbaden: VS Verlag für Sozialwissenschaften.

Kolloch, M. & Golker, O. (2016). Staatliche Regulierung und Digitalisierung als Antezedenzien für Innovationen in der Energiewirtschaft am Beispiel von REMIT. *Zeitschrift für Energiewirtschaft*, *40*(1), 41–54.

Kopplin, I. (2022, Juni 28). Siemens kauft Brightly Software. *FAZ*, S. 19. Frankfurt a.M.

Kotlarsky, J., van Fenema, P. C. & Willcocks, L. P. (2008). Developing a knowledge-based perspective on coordination: The case of global software projects. In J. Kotlarsky, I. Ilan, & P. C. van Fenema (Hg.), *Knowledge Processes in Globally Distributed Contexts. Technology, Work and Globalization* (S. 74–105). London: Palgrave Macmillan.

Kruse, J. (2015). *Qualitative Interviewforschung. Ein integrativer Ansatz* (2. überarbeitete und ergänzte Auflage). Weinheim; Basel: Beltz Juventa.

Kunda, G. (1992). *Engineering culture. Control and commitment in a high-tech corporation*. Philadelphia, PA: Temple University Press.

Lamoreaux, N. R., Raff, D. M. & Temin, P. (2003). Beyond markets and hierarchies: Toward a new synthesis of American business history. *The American Historical Review*, *108*(2), 404–433.

Lee, M. Y. & Edmondson, A. C. (2017). Self-managing organizations: Exploring the limits of less-hierarchical organizing. *Research in Organizational Behavior*, *37*, 35–58.

Lehman, M. M. (1980). Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, *68*(9), 1060–1076.

Lenk, K. (2016). Gedanken zur Gestaltung technikdurchtränkter Arbeitsorganisation. In F. Fuchs-Kittowski & W. Kriesel (Hg.), *Informatik und Gesellschaft. Festschrift zum 80. Geburtstag von Klaus Fuchs-Kittowski* (Bd. 80, S. 351–360). Berlin: Peter Lang.

Levina, N. & Vaast, E. (2005). The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems. *MIS Quarterly*, *29*(2), 335.

Ley, T. (2010). *Einführung in die Methode der objektiv-hermeneutischen Sequenzanalyse*. Frankfurt a.M.: Verlag für Polizeiwissenschaft.

Light, B. & Wagner, E. (2006). Integration in ERP environments: rhetoric, realities and organisational possibilities. *New Technology, Work and Employment*, *21*(3), 215–228.

Longen, J. (2015). *Technikeinsatz und Verlagerungsprozesse in Unternehmensnetzwerken: Die Organisation von Callcenter-Dienstleistungen in Deutschland*. Wiesbaden: Springer VS.

Lütjen, H., Tietze, F. & Nuske, T. (2014). *Innovationskooperationen von Stadtwerken: Eine empirische Untersuchung von Treibern und Barrieren*. Norderstedt: Books on Demand.

Lyytinen, K. & Newman, M. (2015). A tale of two coalitions – marginalising the users while successfully implementing an enterprise resource planning system. *Information Systems Journal*, *25*(2), 71–101.

Mahr, B. (2009). Die Informatik und die Logik der Modelle. *Informatik-Spektrum*, *32*(3), 228–249.

Mann, F. C. & Williams, L. K. (1960). Observations on the dynamics of a change to electronic data-processing equipment. *Administrative Science Quarterly*, *5*(2), 217–256.

Marquardt, R.-M. & Bontrup, H.-J. (2010). Beschäftigungsbedingungen und Unternehmenskultur in der Elektrizitätswirtschaft. *WSI-Mitteilungen*, *63*(6), 291–298.

Marrs, K. (2010). Herrschaft und Kontrolle in der Arbeit. In F. Böhle, G. G. Voß, & G. Wachtler (Hg.), *Handbuch Arbeitssoziologie* (S. 331–356). Wiesbaden: VS Verlag für Sozialwissenschaften.

Martin, J. L. (2017). *Thinking Through Methods: A Social Science Primer*. Chicago, IL: University of Chicago Press.

Masak, D. (2006). *IT-Alignment. IT-Architektur und Organisation*. Heidelberg: Springer.

Massimo, F. (2022, Mai 23). How Amazon Invented »Plat-Fordism«. https://jacobin.com/2022/05/amazon-fordism-capitalism-logistics-vertical-horizontal-integration. Zugegriffen: 12. April 2023.

Mayntz, R. (2009). The Changing Governance of Large Technical Infrastructure Systems (2008). In R. Mayntz (Hg.), *Über Governance: Institutionen und Prozesse politischer Regelung* (S. 121–150). Frankfurt a.M.; New York: Campus.

Menz, W., Nies, S. & Sauer, D. (2019). Digitale Kontrolle und Vermarktlichung: Beschäftigtenautonomie im Kontext betrieblicher Strategien der Digitalisierung. *PROKLA. Zeitschrift für kritische Sozialwissenschaft*, *49*(195), 181–200.

Merkens, H. (2012). Auswahlverfahren, Sampling, Fallkonstruktion. In U. Flick, E. von Kardorff, & I. Steinke (Hg.), *Qualitative Forschung: ein Handbuch* (9. Auflage, S. 286–299). Reinbek: Rowohlt Taschenbuch.

Meuser, M. & Nagel, U. (2002). ExpertInneninterviews – vielfach erprobt, wenig bedacht. In A. Bogner, B. Littig, & W. Menz (Hg.), *Das Experteninterview. Theorie, Methode, Anwendung* (S. 71–93). Wiesbaden: VS Verlag für Sozialwissenschaften.

Meves, A.-K. (2021, Juni 24). RWE: Kommunale Aktionäre formieren sich neu. https://www.derneuekaemmerer.de/beteiligungen/stadtwerke/rwe-kommunale-aktionaere-formieren-sich-neu-18734. Zugegriffen: 7. Juni 2023.

Mezihorak, P. (2018). Competition for control over the labour process as a driver of relocation of activities to a shared services centre. *Human Relations*, *71*(6), 822–844.

Mingers, J. & Willcocks, L. (2014). An integrative semiotic framework for information systems: The social, personal and material worlds. *Information and Organization*, *24*(1), 48–70.

Minssen, H. (1999). *Von der Hierarchie zum Diskurs? Die Zumutungen der Selbstregulierung*. München: Hampp.

Minssen, H. (2011). *Arbeit in der modernen Gesellschaft. Eine Einführung*. Wiesbaden: VS Verlag für Sozialwissenschaften.

Miozzo, M. & Grimshaw, D. (2005). Modularity and innovation in knowledge-intensive business services: IT outsourcing in Germany and the UK. *Research Policy*, *34*(9), 1419–1439.

Mola, L., Russo, I., Giangreco, A. & Rossignoli, C. (2017). Who knows what? Reconfiguring the governance and the capabilities of the supply chain between physical and digital processes in the fashion industry. *Production Planning & Control*, *28*(16), 1284–1297.

Monstadt, J. (2004). Die Modernisierung der Stromversorgung. *Regionale Energie- und Klimapolitik im Liberalisierungs- und Privatisierungsprozess. Wiesbaden*.

Monteiro de Carvalho, M. (2013). An investigation of the role of communication in IT projects. *International Journal of Operations & Production Management*, *34*(1), 36–64.

Mormann, H. (2016). *Das Projekt SAP: Zur Organisationssoziologie betriebswirtschaftlicher Standardsoftware*. Bielefeld: transcript.

Morozov, E. (2019, Februar 4). Capitalism's New Clothes. Shoshana Zuboff's new book on »surveillance capitalism« emphasizes the former at the expense of the latter. *The Baffler*. https://thebaffler.com/latest/capitalisms-new-clothes-morozov. Zugegriffen: 11. Juli 2023.

Müller, N. (2010). *Reglementierte Kreativität: Arbeitsteilung und Eigentum im computerisierten Kapitalismus*. Berlin: Edition Sigma.

Müller, S. (2021, Juli 1). Stromnetz Berlin ist wieder im Eigentum des Landes Berlin. *Vattenfall Newsroom*. https://group.vattenfall.com/de/newsroom/pressemitteilungen/2021/stromnetz-berlin-ist-wieder-im-eigentum-des-landes-berlin. Zugegriffen: 7. Juni 2023.

Mutch, A. (2010). Technology, organization, and structure – A morphogenetic approach. *Organization Science*, *21*(2), 507–520.

Nicolini, D., Mengis, J. & Swan, J. (2012). Understanding the role of objects in cross-disciplinary collaboration. *Organization Science*, *23*(3), 612–629.

Ortmann, G., Windeler, A., Becker, A. & Schulz, H.-J. (1990). *Computer und Macht in Organisationen. Mikropolitische Analysen*. Wiesbaden: Springer Fachmedien.

Overbeck, H. (2017). Dienstleistungsarbeit. In H. Hirsch-Kreinsen & H. Minssen (Hg.), *Lexikon der Arbeits- und Industriesoziologie* (2. Auflage, S. 103–106). Baden-Baden: Nomos.

Päffgen, J. & Sperling, C. (2019, Juni 24). EPEX: Datenpanne, Decoupling, Desaster? https://www.next-kraftwerke.de/energie-blog/epex-decoupling. Zugegriffen: 23. Juni 2023.

Peled, A. (2001). Outsourcing and political power: Bureaucrats, consultants, vendors and public information technology. *Public Personnel Management*, *30*(4), 495–514.

Peter, L. (1993). »Jeder irgendwie für sich allein«? Probleme und Chancen sozialer Interaktion am Arbeitsplatz. *Zeitschrift für Soziologie*, *22*(6), 416–432.

Pfeiffer, S. (2021). *Digitalisierung als Distributivkraft: Über das Neue am digitalen Kapitalismus*. Bielefeld: transcript.

Pfeiffer, S., Sauer, S. & Ritter, T. (2014). Agile Methoden als Werkzeug des Belastungsmanagements? Eine arbeitsvermögensbasierte Perspektive. *Arbeit*, *23*(2), 119–132.

Pfeiffer, S. & Schrape, J.-F. (2023). Digitale Transformation von Arbeit. In H. Hirsch-Kreinsen, S. Pfeiffer, M. Will-Zocholl, & R. Bohn (Hg.), *Lexikon der Arbeits- und Industriesoziologie* (3., aktualisierte und erweiterte Auflage, S. 134–138). Baden-Baden: Nomos.

Pohlmann, M. (2017). Management. In H. Hirsch-Kreinsen & H. Minssen (Hg.), *Lexikon der Arbeits- und Industriesoziologie* (2. Auflage, S. 207–211). Baden-Baden: Nomos.

Pollock, N. & Williams, R. (2009). *Software and Organisations: The Biography of the Enterprise-Wide System or How SAP Conquered the World*. London; New York, NY: Routledge.

Pollock, N., Williams, R. & D'Adderio, L. (2007). Global software and its provenance: Generification work in the production of organizational software packages. *Social Studies of Science*, *37*(2), 254–280.

Pongratz, H. J. & Voß, G. G. (1997). Fremdorganisierte Selbstorganisation. Eine soziologische Diskussion aktueller Managementkonzepte. *German Journal of Human Resource Management*, *11*(1), 30–53.

Ponte, D., Rossi, A. & Zamarian, M. (2009). Cooperative design efforts for the development of complex IT-artefacts. *Information Technology & People*, *22*(4), 317–334.

Powell, W. W. (1990). Neither market nor hierarchy – Network forms of organization. *Research in Organizational Behavior*, *12*, 295–336.

Puranam, P., Alexy, O. & Reitzig, M. (2014). What's »new« about new forms of organizing? *Academy of Management Review*, *39*(2), 162–180.

Püttner, G. (2012). Stadtwerke zwischen Daseinsvorsorge und Wettbewerb. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 139–153). Baden-Baden: Nomos.

Ralph, P. (2015). The sensemaking-coevolution-implementation theory of software design. *Science of Computer Programming*, *101*, 21–41.

Ramioul, M. & De Vroom, B. (2009). *Global value chain restructuring and the use of knowledge and skills*. HIVA-KU Leuven; Leuven.

Reich, B. H. & Benbasat, I. (2000). Factors that influence the social dimension of alignment between business and information technology objectives. *MIS Quarterly*, *24*(1), 81–113.

Remer, S. (2008). *Soziale Strukturen und Informationstechnologie Die organisatorische Bedeutung von »Service Oriented Architectures«*. Technische Universität. Abgerufen von http://tuprints.ulb.tu-darmstadt.de/1191/2/Soziale_Strukturen_und_Informationstechnologie.pdf. Zugegriffen: 27. Februar 2024.

Rennstam, J. (2012). Object-control: A study of technologically dense knowledge work. *Organization Studies*, *33*(8), 1071–1090.

Robertson, B. J. (2007). Organization at the leading edge: Introducing Holacracy™. *Integral Leadership Review*, *7*(3), 1–13.

Robertson, P. L. & Verona, G. (2006). Post-Chandlerian Firms: Technological Change and Firm Boundaries. *Australian Economic History Review*, *46*(1), 70–94.

Robey, D. & Sahay, S. (1996). Transforming work through information technology: A comparative case study of geographic information systems in county government. *Information Systems Research*, *7*(1), 93–110.

Rock, R., Ulrich, P. & Witt, F. H. (1990). *Dienstleistungsrationalisierung im Umbruch*. Opladen: Westdeutscher Verlag.

Rödl und Partner. (2017). *Energiewirtschaft: Digitalisierung der Geschäftsprozesse und IT im Unternehmen transformieren – Potenziale nachhaltig nutzen*. Nürnberg/Köln: Rödl & Partner.

Rohracher, H. (2007). Die Wechselwirkung technischen und institutionellen Wandels in der Transformation von Energiesystemen. In U. Dolata & R. Werle (Hg.), *Gesellschaft und die Macht der Technik* (S. 133–151). Frankfurt a.M.; New York: Campus.

Ross, A. & Chiasson, M. (2011). Habermas and information systems research: New directions. *Information and Organization*, *21*(3), 123–141.

Roth, I. (2018, Mai). Digitalisierung in der Energiewirtschaft. Technologische Trends und ihre Auswirkungen auf Arbeit und Qualifizierung. Working Paper Förschungsförderung, Nummer 73. https://www.boeckler.de/pdf/p_fofoe_WP_073_2018.pdf. Zugegriffen: 19. September 2018.

Rüegg-Stürm, J. & Young, M. (2001). Die Bedeutung neuer netzwerkartiger Führungs- und Organisationsformen für die Dynamisierung von Unternehmungen. *Die Unternehmung*, *55*(3), 187–213.

Sack, D. (2018). Zwischen europäischer Liberalisierung und Energiewende – Der Wandel der Governanceregime im Energiesektor (1990–2016). In L. Holstenkamp & J. Radtke (Hg.), *Handbuch Energiewende und Partizipation* (S. 81–99). Wiesbaden: Springer VS.

Sahaym, A., Steensma, H. K. & Schilling, M. A. (2007). The influence of information technology on the use of loosely coupled organizational forms: An industry-level analysis. *Organization Science*, *18*(5), 865–880.

Sander, C. (2009). *Kooperationen kommunaler Energieversorger: eine empirische Bestandsaufnahme* (No. 78). Münster: Westfälischen Wilhelms-Universität Münster.

Sarshar, K., Loos, P. & Weber, M. (2006). Einsatz der Informationsmodellierung bei der Einführung betrieblicher Standardsoftware. *Wirtschaftsinformatik*, *48*(2), 120–127.

Sauer, D. (2017). Systemische Rationalisierung/Wertschöpfungsketten. In H. Hirsch-Kreinsen & H. Minssen (Hg.), *Lexikon der Arbeits- und Industriesoziologie* (2. Auflage, S. 285–289). Baden-Baden: Nomos.

Sauer, D. (2018). Vermarktlichung und Vernetzung der Unternehmens- und Betriebsorganisation. In F. Böhle, G. G. Voß, & G. Wachtler (Hg.), *Handbuch Arbeitssoziologie. Band 2: Akteure und Institutionen* (S. 177–206). Wiesbaden: Springer VS.

Schäfer, M. (2014). *Kommunalwirtschaft: eine gesellschaftspolitische und volkswirtschaftliche Analyse*. Wiesbaden: Springer Gabler.

Schaupp, S. (2021). *Technopolitik von unten. Algorithmische Arbeitssteuerung und kybernetische Proletarisierung*. Berlin: Matthes & Seitz.

Schimank, U. & Hamp, A. (2010). *Handeln und Strukturen: Einführung in die akteurtheoretische Soziologie* (4., völlig überarbeitete Auflage). Weinheim: Juventa Verlag.

Schiek, D. (2022). Schriftliche Online-Interviews in der qualitativen Sozialforschung: zur methodologischen Begründung einer neuen Forschungspraxis. *Forum Qualitative Sozialforschung*, *23*(1).

Schilcher, C. & Diekmann, J. (2012). Arbeit, Informatisierung und die neue Rolle des Wissens. In C. Schilcher & M. Will-Zocholl (Hg.), *Arbeitswelten in Bewegung* (S. 25–57). Wiesbaden: VS Verlag für Sozialwissenschaften.

Schlosser, F., Beimborn, D., Weitzel, T. & Wagner, H.-T. (2015). Achieving social alignment between business and IT – An empirical evaluation of the efficacy of IT governance mechanisms. *Journal of Information Technology*, *30*(2), 119–135.

Schmiede, R. (2006). Wissen und Arbeit im »Informational Capitalism«. In A. Baukrowitz, T. Berker, S. Pfeiffer, R. Schmiede, & M. Will (Hg.), *Informatisierung der Arbeit – Gesellschaft im Umbruch* (S. 457–492). Berlin: Edition Sigma.

Schmiede, R. (2015). Homo faber digitalis? Zur Dialektik von technischem Fortschritt und Arbeitsorganisation. *Mittelweg 36*, *24*(6), 37–58.

Schmiede, R. (2017). Informationsgesellschaft. In H. Hirsch-Kreinsen & H. Minssen (Hg.), *Lexikon der Arbeits-und Industriesoziologie* (2. Auflage, S. 187–190). Baden-Baden: Nomos.

Schmierl, K. & Pfeiffer, S. (2005). Lego-Logik der kapitalistischen »Netzwerkökonomie« – Theoretische Spekulationen zum Wandel von Betrieb und Technik. *Die Organisation der Arbeit. München/Mering: Hampp*, 43–66.

Schöneich, M. (2012). Strukturwandel der Stadtwerke. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 73–93). Baden-Baden: Nomos.

Schulz, M. & Ruddat, M. (2012). »Let's talk about sex!« Über die Eignung von Telefoninterviews in der qualitativen Sozialforschung. *Forum Qualitative Sozialforschung*, *13*(3).

Schulz-Schaeffer, I. (1996). Software-Entwicklung zwischen Ingenieur- und Designwissenschaft: Überzeugungskraft und nützliche Widersprüchlichkeit von Software-Engineering und Software-Gestaltung. In H. D. Hellige (Hg.), *Technikleitbilder auf dem Prüfstand: Das Leitbild-Assessment aus informatik- und computerhistorischer Sicht* (S. 115–140). Berlin: Edition Sigma.

Schulz-Schaeffer, I. (1999). Technik und die Dualität von Ressourcen und Routinen: zur sozialen Bedeutung gegenständlicher Technik. *Zeitschrift für Soziologie*, *28*(6), 409–428.

Schulz-Schaeffer, I. & Bottel, M. (2018). Die Herstellung transnational mobiler Arbeitstätigkeiten in der Softwareentwicklung. In S. Quack, I. Schulz-Schaeffer, K. Shire, & A. Weiß (Hg.), *Transnationalisierung der Arbeit* (S. 99–127). Wiesbaden: Springer VS.

Schulz-Schaeffer, I. & Funken, C. (2008). Das Verhältnis von Formalisierung und Informalität betrieblicher Arbeits- und Kommunikationsprozesse und die Rolle der Informationstechnik. In C. Funken & I. Schulz-Schaeffer (Hg.), *Digitalisierung der Arbeitswelt. Zur Neuordnung formaler und informeller Prozesse in Unternehmen* (S. 11–39). Wiesbaden: VS Verlag für Sozialwissenschaften.

Schwarz, G. M. & Brock, D. M. (1998). Waving hello or waving good-bye? Organizational change in the information age. *The International Journal of Organizational Analysis*, *6*(1), 65–90.

Schwintowski, H.-P. (2012). Public Corporate Governance öffentlicher Unternehmen für Stadtwerke. In D. Bräunig & W. Gottschalk (Hg.), *Stadtwerke. Grundlagen, Rahmenbedingungen, Führung und Betrieb* (S. 319–343). Baden-Baden: Nomos.

Seeliger, A., Michalik, S., Roller, J. & Kühnhenrich, D. (2019). Bürokratiekosten der Energiewende. *WISTA – Wirtschaft und Statistik*, *71*(6), 59–72.

Sesay, A. & Ramirez, R. (2016). Theorizing the IT Governance role in IT sourcing research. *AMCIS 2016 Proceedings*. 15.
https://aisel.aisnet.org/amcis2016/SCU/Presentations/15. Zugegriffen: 24. Mai 2023.

Shaikh, M. & Henfridsson, O. (2017). Governing open source software through coordination processes. *Information and Organization*, *27*(2), 116–135.

Siegele, L. & Zepelin, J. (2009). *Matrix der Welt: SAP und der neue globale Kapitalismus*. Frankfurt a.M.; New York: Campus.

Silva, L. & Backhouse, J. (1997). Becoming part of the furniture: The institutionalization of information systems. In A. S. Lee, Liebenau J. & DeGross J. I. (Hg.), *Information Systems and Qualitative Research* (S. 389–414). Boston (MA): Springer.

Silva, L. O. (2005). Theoretical Approaches for Researching Power and Information Systems: The Benefit of a Machiavellian View. In D. Howcroft & E. M. Trauth (Hg.), *Handbook of Critical Information Systems Research: Theory and Application* (S. 47–69). Cheltenham: Edward Elgar.

Song, M., Berends, H., Van der Bij, H. & Weggeman, M. (2007). The effect of IT and colocation on knowledge dissemination. *Journal of Product Innovation Management*, *24*(1), 52–68.

Sonnenholzner, J. (2020). Horch, was kommt von draußen rein. Outsourcing – Nur ein verschwindend geringer Teil der IT entsteht in den Stadtwerken selbst. Ohne Offenheit für externes Wissen geht es nicht mehr. Was jedoch ist das richtige Maß? *ZfK – Zeitung für kommunale Wirtschaft*, (1), 13.

Sperling, C. (2019, Februar 11). Wer die Netzfrequenz stört. https://www.next-kraftwerke.de/energie-blog/stromnetzfrequenz. Zugegriffen: 28. April 2023.

Srikanth, K. & Puranam, P. (2014). The firm as a coordination system: Evidence from software services offshoring. *Organization Science*, *25*(4), 1253–1271.

Srnicek, N. (2017). *Platform Capitalism*. Cambridge; Malden, MA: Polity Press.

Staab, P. (2019). *Digitaler Kapitalismus: Markt und Herrschaft in der Ökonomie der Unknappheit*. Berlin: Suhrkamp.

Stadtwerke München GmbH (Hg.). (2023, April). Geschäftsbericht 2022. https://www.swm.de/dam/doc/swm/swm-geschaeftsbericht.pdf. Zugegriffen: 20. April 2023.

Statistisches Bundesamt (Hg.). (2011, Juli 13). 2020 Beschäftigung, Umsatz, Investitionen und Kostenstruktur der Rechtlichen Einheiten in der Energieversorgung, Wasserversorgung, Abwasser- und Abfallentsorgung, Beseitigung von Umweltverschmutzungen. https://www.destatis.de/DE/Themen/Branchen-Unternehmen/Energie/Beschaeftigte-Umsatz-Investitionen/Publikationen/Downloads-Beschaeftigte/beschaeftigung-umsatz-kostenstruktur-2040610207004.pdf?__blob=publicationFile. Zugegriffen: 2. Mai 2023.

Statistisches Bundesamt (Hg.). (2022, Juni 3). 2009 Beschäftigung, Umsatz, Investitionen und Kostenstruktur der Rechtlichen Einheiten in der Energieversorgung, Wasserversorgung, Abwasser- und Abfallentsorgung, Beseitigung von Umweltverschmutzungen. Zugegriffen: 22. Juli 2019.

Steinke, I. (2012). Gütekriterien qualitativer Forschung. In U. Flick, E. von Kardorff, & I. Steinke (Hg.), *Qualitative Forschung: Ein Handbuch* (9. Auflage, S. 319–331). Reinbek: Rowohlt Taschenbuch.

Strulik, T. (2017). Wissensgesellschaft. In H. Hirsch-Kreinsen & H. Minssen (Hg.), *Lexikon der Arbeits- und Industriesoziologie* (2. Auflage, S. 322–325). Baden-Baden: Nomos.

Sutton, R. I. & Staw, B. M. (1995). What theory is not. *Administrative Science Quarterly*, *40*(3), 371–384.

Svejvig, P. & Jensen, T. B. (2013). Making sense of enterprise systems in institutions: A case study of the re-Implementation of an accounting system. *Scandinavian Journal of Information Systems*, *25*(1), 3–36.

Swan, J. & Scarbrough, H. (2005). The politics of networked innovation. *Human Relations*, *58*(7), 913–943.

Sydow, J. & Helfen, M. (2020). Work and Employment in Fluid Organizational Forms. In B. J. Hoffman, M. K. Shoss, & L. A. Wegman (Hg.), *The Cambridge Handbook of the Changing Nature of Work* (S. 214–236). Cambridge: Cambridge University Press.

Sydow, J. & Windeler, A. (2000). Steuerung von und in Netzwerken – Perspektiven, Konzepte, vor allem aber offene Fragen. In J. Sydow (Hg.), *Steuerung von Netzwerken. Konzepte und Praktiken* (S. 1–24). Opladen: Westdeutscher Verlag.

Symon, G. (2000). Information and communication technologies and the network organization: A critical analysis. *Journal of Occupational and Organizational Psychology*, *73*(4), 389–414.

Thompson, P. & Laaser, K. (2021). Beyond technological determinism: Revitalising labour process analyses of technology, capital and labour. *Work in the Global Economy*, *1*(1–2), 139–159.

Thornberg, R. (2012). Informed grounded theory. *Scandinavian Journal of Educational Research*, *56*(3), 243–259.

Tiwana, A. & Kim, S. K. (2016). Concurrent IT sourcing: Mechanisms and contingent advantages. *Journal of Management Information Systems*, *33*(1), 101–138.

Trianel GmbH (Hg.). (2022). Geschäfts- und Nachhaltigkeitsbericht 2021. https://www.trianel.com/fileadmin/user_upload/unternehmen/zahlen_und_fakten/trianel-geschaefts-und-nachhaltigkeits-bericht-2021.pdf. Zugegriffen: 28. April 2023.

Tsamenyi, M., Cullen, J. & González, J. M. G. (2006). Changes in accounting and financial information system in a Spanish electricity company: A new institutional theory analysis. *Management Accounting Research*, *17*(4), 409–432.

Upadhya, C. (2009). Controlling offshore knowledge workers: Power and agency in India's software outsourcing industry. *New Technology, Work and Employment*, *24*(1), 2–18.

Uzzi, B. (1997). Social structure and competition in interfirm networks: The paradox of embeddedness. *Administrative science quarterly*, 35–67.

Valorinta, M. (2011). IT alignment and the boundaries of the IT function. *Journal of Information Technology*, *26*(1), 46–59.

van Fenema, P. C., Keers, B. & Zijm, H. (2014). Interorganizational Shared Services: Creating Value Across Organizational Boundaries. In T. Bondarouk (Hg.), *Shared Services as a New Organizational Form* (S. 175–217). Bingley: Emerald Group Publishing Limited.

Vitols, S. (2002). Shareholder value, management culture and production regimes in the transformation of the German chemical-pharmaceutical industry. *Competition and Change*, *6*(3), 309–325.

Vogel, O., Arnold, I., Chughtai, A., Ihler, E., Kehrer, T., Mehlig, U. & Zdun, U. (2009). *Software-Architektur. Grundlagen – Konzepte – Praxis* (2. Auflage). Heidelberg: Spektrum Akademischer Verlag.

Volkoff, O., Strong, D. M. & Elmes, M. B. (2007). Technological embeddedness and organizational change. *Organization Science*, *18*(5), 832–848.

von Jouanne-Diedrich, H., Zarnekow, R. & Brenner, W. (2005). Industrialisierung des IT-Sourcings. *HMD-Praxis der Wirtschaftsinformatik*, *245*, 18–27.

von Petersdorff, W. (2013, November 17). Lichtblick macht den Strom schlau. *FAZ*, S. 28. Frankfurt a.M.

Vormbusch, U. (2002). *Diskussion und Disziplin. Gruppenarbeit als kommunikative und kalkulative Praxis*. Frankfurt a.M.; New York: Campus.

Voß, J.-P. & Bauknecht, D. (2007). Der Einfluss von Technik auf Governance-Innovationen: Regulierung zur gemeinsamen Netznutzung in Infrastruktursystemen. In U. Dolata & R. Werle (Hg.), *Gesellschaft und die Macht der Technik. Sozioökonomischer und institutioneller Wandel durch Technisierung* (S. 109–131). Frankfurt a.M.; New York: Campus.

Walker, E.-M. (2016). »Dadurch wird unsere Arbeit weiter nach vorne verlagert in der Prozesskette.« Organisationale Anerkennungsphänomene bei der Einführung eines digitalen Warenwirtschaftssystems. *AIS Studien*, *9*(1).

Waterson, P. E., Clegg, C. W. & Axtell, C. M. (1997). The dynamics of work organization, knowledge and technology during software development. *International Journal of Human-Computer Studies*, *46*(1), 79–101.

Weber, M. (1980). *Wirtschaft und Gesellschaft. Grundriß der verstehenden Soziologie* (5. rev. Auflage). Tübingen: Mohr.

Weber, M. (1988). *Gesammelte Politische Schriften*. (J. Winckelmann, Hg.). Tübingen: Mohr.

Weishaupt, S. & Hösl, G. (2017). Subjektivierendes Arbeitshandeln beim Produktmanagement. In F. Böhle (Hg.), *Arbeit als Subjektivierendes Handeln. Handlungsfähigkeit bei Unwägbarkeiten und Ungewissheit* (S. 493–506). Wiesbaden: Springer VS.

Weizenbaum, J. (1978). *Die Macht der Computer und die Ohnmacht der Vernunft* (14. Auflage). Frankfurt a.M.: Suhrkamp.

Wenger, E. (1999). *Communities of Practice: Learning, Meaning, and Identity*. Cambridge, MA: Cambridge University Press.

Weyer, J. (2010). Netzwerke in der Techniksoziologie. Karriere und aktueller Stellenwert eines Begriffs. In C. Stegbauer & R. Häußling (Hg.), *Handbuch Netzwerkforschung* (S. 847–855). Wiesbaden: VS Verlag für Sozialwissenschaften.

Wiener, M., Mähring, M., Remus, U. & Saunders, C. (2016). Control configuration and control enactment in information systems projects. *MIS Quarterly*, *40*(3), 741–774.

Wiesche, M., Jurisch, M. C., Yetton, P. W. & Krcmar, H. (2017). Grounded theory methodology in information systems research. *MIS Quarterly*, *41*(3), 685–701.

Wiesenthal, H. (2000). Markt, Organisation und Gemeinschaft als »zweitbeste« Verfahren sozialer Koordination. In R. Werle & U. Schimank (Hg.), *Gesellschaftliche Komplexität und kollektive Handlungsfähigkeit* (S. 44–73). Frankfurt a.M.; New York: Campus.

Wilkesmann, U. (2005). Die Organisation von Wissensarbeit. *Berliner Journal für Soziologie*, *15*(1), 55–72.

Williams, P. (2002). The competent boundary spanner. *Public Administration*, *80*(1), 103–124.

Williamson, O. E. (1985). *The Economic Institutions of Capitalism: Firms, Markets, Relational Contracting*. New York: Free Press.

Willke, H. (1998). Organisierte Wissensarbeit. *Zeitschrift für Soziologie*, *27*(3), 161–177.

Windeler, A. & Wirth, C. (2010). Betriebliche und überbetriebliche Organisation: Netzwerke und Arbeit. In F. Böhle, G. G. Voß, & G. Wachtler (Hg.), *Handbuch Arbeitssoziologie* (S. 569–596). Wiesbaden: VS Verlag für Sozialwissenschaften.

Windolf, P. (Hg.). (2005). *Finanzmarkt-Kapitalismus. Analysen zum Wandel von Produktionsregimen*. Wiesbaden: VS Verlag für Sozialwissenschaften.

Wohlin, C., Šmite, D. & Moe, N. B. (2015). A general theory of software engineering: Balancing human, social and organizational capitals. *Journal of Systems and Software*, *109*, 229–242.

Wolff, B., Fuchs-Kittowski, K., Klischewski, R., Möller, A. & Rolf, A. (1999). Organisationstheorie als Fenster zur Wirklichkeit. In J. Becker, W. König, R. Schütte, O. Wendt & S. Zelewski (Hg.), *Wirtschaftsinformatik und Wissenschaftstheorie. Bestandsaufnahme und Perspektiven* (S. 289–327). Wiesbaden: Gabler.

Wood, A. J., Graham, M., Lehdonvirta, V. & Hjorth, I. (2019). Good gig, bad gig: autonomy and algorithmic control in the global gig economy. *Work, Employment and Society*, *33*(1), 56–75.

Wu, Q., Zhang, H., Li, Z. & Liu, K. (2019). Labor control in the gig economy: Evidence from Uber in China. *Journal of Industrial Relations*, *61*(4), 574–596.

Zammuto, R. F., Griffith, T. L., Majchrzak, A., Dougherty, D. J. & Faraj, S. (2007). Information Technology and the Changing Fabric of Organization. *Organization Science*, *18*(5), 749–762.

ZDNet Staff. (2004, Juli 20). Understanding software as a commodity. What does it mean to commoditise software? And is the definition of what a commodity is actually clear? https://www.zdnet.com/article/understanding-software-as-a-commodity/. Zugegriffen: 12. April 2023.

ZfK. (2019a, Februar 25). Vattenfall und EnBW kooperieren bei White-Label-Lösung. https://www.zfk.de/unternehmen/nachrichten/vattenfall-und-enbw-kooperieren-bei-white-label-loesung. Zugegriffen: 12. April 2023.

ZfK. (2019b, Juni 3). EnBW und Powercloud bieten Komplettlösung für Lieferanten. https://www.zfk.de/digitalisierung/it/enbw-und-powercloud-bieten-komplettloesung-fuer-lieferanten. Zugegriffen: 12. April 2023.

ZfK. (2022a, Februar 1). Plattform für Mieterstrom, Elektromobilität, Heizkostenverordnung und Co. https://www.zfk.de/digitalisierung/it/plattform-fuer-hkvo-mieterstrom-elektromobilitaet-und-co. Zugegriffen: 12. April 2023.

ZfK. (2022b, Juli 2). Chancen und Herausforderungen von Smart-City-Plattformen. https://www.zfk.de/digitalisierung/smart-city-energy/chancen-und-herausforderungen-von-smart-city-plattformen. Zugegriffen: 12. April 2023.

Zhu, K. X. & Zhou, Z. Z. (2012). Research note – Lock-in strategy in software competition: Open-source software vs. proprietary software. *Information Systems Research*, *23*(2), 536–545.

Ziegler, A. (2020). *Der Aufstieg des Internet der Dinge: Wie sich Industrieunternehmen zu Tech-Unternehmen entwickeln*. Frankfurt a.M.; New York: Campus.

Zuboff, S. (1988). *The Age of the Smart Machine*. New York, NY: Basic Books.

Zuboff, S. (2018). *Das Zeitalter des Überwachungskapitalismus*. Frankfurt a.M.; New York: Campus Verlag.

Anhang

Übersicht Interviews

Tabelle 31: Übersicht Interviews

Fall	Position	Organisation	Datum	Dauer	Geschlecht
INTERN1	Teamleiter IT	EVU	14.07.2020	01:55:11	m
INTERN1	Software-Architekt	EVU	24.11.2021	01:16:24	m
INTERN1	Betriebsrat	EVU	01.12.2021	00:56:24	m
INTERN1	Programmierer	EVU	14.12.2021	01:12:12	m
INTERN1	Monteur	EVU	25.02.2022	00:44:59	m
INTERN1	Anforderungsmanagerin	EVU	16.05.2022	00:56:54	w
INTERN2	IT-Manager Fachbereich	EVU	02.07.2020	01:51:53	m
INTERN2	Programmierer	EVU	15.07.2021	00:54:47	m
INTERN2	Product Owner	EVU	15.07.2021	01:01:59	m
INTERN2	Anforderungsmanagerin	EVU	28.07.2021	00:58:53	w
INTERN2	Anwender	EVU	03.08.2021	01:05:36	m
INTERN2	Scrum Master	EVU	15.10.2021	01:10:49	m
INTERN2	Betriebsrat	EVU	18.11.2021	00:56:54	m
INTERN2	Manager Abteilung	EVU	03.12.2021	00:54:00	m
KOOP1	Digitalisierungsmanager	IT-DL	17.03.2020	01:19:57	m
KOOP1	Anforderungsmanager	IT-DL	25.03.2020	01:35:47	m
KOOP1	Key Account Manager	IT-DL	21.04.2020	01:30:17	m
KOOP1	Manager	IT-DL	28.04.2020	01:28:09	m
KOOP1	Prozessmanagerin	EVU1	19.05.2021	00:58:20	w
KOOP1	Betriebsrat	EVU3	29.07.2021	00:54:39	m

KOOP1	Programmierer1	IT-DL	16.09.2021	00:38:09	m
KOOP1	Programmierer2	IT-DL	14.10.2021	01:00:25	m
KOOP1	Digitalisierungsmanager	EVU2	19.10.2021	00:38:54	m
KOOP1	Betriebsrat	EVU2	03.11.2021	01:00:06	m
KOOP1	Anwenderin	EVU1	18.11.2021	00:47:14	w
KOOP1	Anforderungsmanager	EVU2	18.11.2021	01:25:47	m
KOOP1	Betriebsrat	EVU4	23.11.2021	01:07:51	m
KOOP1	Applikationsbetreuer	EVU2	30.11.2021	01:11:13	m
KOOP1	Prozessmanager	EVU3	02.12.2021 07.12.2021	00:58:14 00:32:29	m
KOOP1	Anwenderin	EVU3	16.12.2021	00:58:23	w
KOOP1	Teamleiter	EVU2	20.05.2022	00:51:38	m
KOOP2	Sachbearbeiter	EVU1	21.04.2020	00:50:39	m

Fall	Position	Organisation	Datum	Dauer	Geschlecht
KOOP2	Teamleiter Beratung/Entwicklung	IT-DL	15.06.2020	02:38:46	m
KOOP2	IT-Berater	IT-DL	23.07.2020	01:27:53	m
KOOP2	Abteilungsleiter	IT-DL	07.08.2020	01:27:09	m
KOOP2	Programmierer	IT-DL	15.09.2021	00:52:20	m
KOOP2	Change Managerin	EVU2	16.09.2021	00:53:07	w
KOOP2	IT-Projektleiter	EVU2	29.10.2021	00:50:29	m
KOOP2	Manager Digitalisierung	EVU3	21.01.2022	00:51:37	m
KOOP2	IT-Koordinator	EVU2	27.01.2022	00:56:26	m
KOOP2	Sachbearbeiterin	EVU2	11.02.2022	00:55:45	w
KOOP2	Betriebsrätin	EVU2	16.02.2022	00:58:43	w
KOOP2	Teamleiterin	EVU2	01.03.2022	01:04:29	w
KOOP3	Teamleiter	EVU1	30.03.2020	01:10:19	m
KOOP3	Programmierer	Softwarefirma	22.04.2021 29.04.2021	00:28:57 00:20:04	m
KOOP3	Teamleiter und Product Owner	IT-DL	13.10.2021	00:56:51	m
KOOP3	Gruppenleiter	EVU3	22.10.2021	00:46:45	m
PAKET	Lösungsarchitekt	Softwarefirma	01.04.2020	00:46:38	m
PAKET	Partnermanager	Softwarefirma	01.04.2020	00:39:54	m
PAKET	Führungskraft	Softwarefirma	08.04.2020	00:40:17	m
PAKET	IT-Leiter	EVU1	18.05.2020	01:14:06	m

PAKET	IT-Leiter	EVU2	04.03.2021	01:31:21	m
PAKET	Teamleiter	EVU2	06.05.2021	01:22:11	m
PAKET	Anwendungsbetreuerin	EVU3	12.10.2021	01:01:08	w
PAKET	Gruppenleiter	EVU4	29.10.2021	00:52:00	m
PAKET	Teamleiterin	EVU5	29.11.2021	01:11:23	w
PAKET	Programmierer	Softwarefirma	09.12.2021	01:04:24	m
PAKET	Gruppenleiter	EVU3	23.02.2022	00:45:12	m
PAKET	Betriebsrätin	EVU6	15.07.2022	00:56:40	w
START-UP	Programmierer1	Start-up	23.01.2023	00:36:04	m
START-UP	Programmierer2	Start-up	17.02.2023	00:35:51	m
START-UP	Product Ownerin	Start-up	04.10.2022	per E-Mail	w

Leitfaden für Interviews

Der Leitfaden stellt eine Sammlung von Fragen dar, aus denen je nach Interview dann die entsprechenden ausgewählt oder bei speziellen Interviews zusätzliche formuliert wurden. Am Kopf des Leitfadens hatte ich immer die folgende Gedächtnisstütze für mich angebracht:

Zur Erinnerung: Was vor dem Interview gesagt werden muss

- *Ziel der Untersuchung, Rolle, die das Interview für die Erreichung des Ziels spielt*
- *Meine Hintergrund als IT-Berater*
- *Hinweis: Daten sind geschützt und Anonymität der Untersuchung ist gesichert*

Zur Erinnerung: Bitte beachten

- *Erfahrungen, Wissen, Hintergrund, Meinung: hypothetische/erzählanregende/detaillierende Fragen*
- *sozialer Prozess soll rekonstruiert werden können*
- *offen = Frage hat keinen Einfluss auf die Antwort*
- *Pausen zulassen*
- *Nicht Verstandenes klären, Details erfragen (Teile der Antwort wiederholen/präzisieren; »Könnten Sie das bitte nochmal erklären?«)*

Einstieg/Kernfragen

- Was ist Ihre Position im Unternehmen und wie sind Sie dazu gekommen?
- Beschreiben Sie bitte, was Ihre Tätigkeiten sind!
- Wie sieht ein gewöhnlicher Arbeitstag bei Ihnen aus?
- Mit welcher Software arbeiten Sie?
- Haben Sie auch schon einmal mit einer anderen Software gearbeitet?
- Was hat sich, seitdem Sie in der Firma arbeiten, Wesentliches geändert, was die Software anbelangt?
- Was hat sich sonst Wesentliches verändert?
- Inwiefern waren Sie in Projekten beteiligt die letzten Jahre?
- Wie verhält sich der Arbeitsanteil von Projekt- zur Linienarbeit?
- Mit welchen anderen Abteilungen Ihrer Firma müssen Sie sich für Ihre Arbeit abstimmen?
- Inwiefern arbeiten Sie mit Ihrer IT-Abteilung zusammen?
- Welche Aufgaben übernimmt Ihre interne IT?
- Inwiefern arbeiten Sie mit externen IT-DL zusammen?
- Sind Sie Teil eines Teams?
- Wie viele Mitarbeiter hat dieses Team?
- Wie verortet sich das Team in der Organisation?
- Welche Qualifikationsprofile sind vorhanden?
- Wie hoch ist der Automatisierungsgrad in Ihrem Arbeitsbereich?
- Welche Folgen hatte das für Ihre Arbeit?
- Wie könnte aus Ihrer Sicht der Automatisierungsgrad erhöht werden?
- Inwiefern arbeiten Sie mit dem Softwareanbieter zusammen?
- Sind Sie an irgendwelchen Treffen bezüglich der Software beteiligt?
- Inwiefern können Sie Anforderungen aufnehmen bzw. Änderungsvorschläge für die Software machen?
- Inwiefern arbeiten Sie mit einem Ticketsystem?
- Wie viele Tickets haben Sie abgearbeitet?
- Inwiefern nehmen Sie selber Einstellungen an der Software vor?
- Wie stark verwenden Sie den Standard der Software?
- Welche Rolle spielen individuelles Customizing oder Programmierung?
- Was wurde individuell programmiert?
- Inwiefern sind Sie an Softwaretests beteiligt?
- Welche Rolle spielen Key User beim Einsatz der Software?
- Welche Rolle spielen Berechtigungen für Ihre Arbeit?
- Inwiefern ist Ihr Zugriff beschränkt?
- Inwiefern haben Sie einen Überblick über den gesamten Prozess?
- Inwiefern spielen Arbeitsanweisungen eine Rolle in Ihrer Arbeit?
- Inwiefern spielen Dokumentationen zur Software eine Rolle?
- Was könnte aus Ihrer Sicht helfen, damit die Arbeit in Ihrem Bereich effizienter wird? (bspw. Anwender schulen, automatisieren, Usability steigern)

- Was könnte aus Ihrer Sicht helfen, damit der teamübergreifende Prozess effizienter wird?
- Inwiefern könnte die Software verbessert werden?
- Inwiefern könnte die Zusammenarbeit mit den Kollegen verbessert werden?
- Inwiefern könnte die Zusammenarbeit mit der IT verbessert werden?
- Inwiefern könnte die Zusammenarbeit mit dem Softwareanbieter verbessert werden?

Vertiefend: Ihre Arbeitsbedingungen

- Was waren die letzten Schulungen, die Sie besucht haben?
- Welche Rolle spielt Learning by Doing für Sie?
- Würden Sie gerne mehr von anderen lernen/wissen?
- Welches Wissen oder Qualifikation wäre für Ihre Arbeit noch hilfreich?
- Inwiefern spielt fachliche Wissen eine Rolle? (Prozesse, Branche, Organisation)
- Wie ist das Verhältnis zwischen Softwarewissen und fachlichem Wissen in Ihrer Arbeit?
- Welche Rolle spielt Ihre Führungskraft bei Ihrer Arbeit?
- Welche Entscheidungen trifft die Führungskraft?
- Wie werden Ihnen Ihre Aufgaben zugewiesen?
- Inwiefern können Sie Ihre Arbeit eigenständig planen und ausführen?
- Welche Rolle spielen Zielvorgaben in Ihrer Arbeit?
- In welcher Form wird Ihre Arbeit kontrolliert?
- Wie viel Zeit arbeiten Sie alleine vor sich hin?
- Inwiefern können Sie Verbesserungsvorschläge einbringen?
- Inwiefern können Sie anbringen, wenn Sie etwas frustriert?
- Inwiefern haben Sie Kundenkontakt?

Vertiefend: Team

- Gibt es regelmäßige Teamtreffen?
- Um was geht es bei den Teamtreffen? (detaillierte Planung, Priorisierung, Kontrolle, Austausch)
- Welche Rolle spielt die Kommunikation zwischen den Teammitgliedern bei der Arbeit?
- Inwiefern findet ein Wissensaustausch statt?
- Wie könnte die Zusammenarbeit verbessert werden?
- Inwiefern spielen konkrete Arbeitsanweisungen eine Rolle?
- Wie werden die Aufgaben verteilt?
- Inwiefern ist jede:r Spezialist:in in seinem bzw. ihrem Gebiet und arbeitet selbstständig?
- Welche Rolle spielt Ihre Führungskraft im Team?

- Gibt es Zielvorgaben für das Team?
- In welcher Form wird die Arbeit im Team kontrolliert?

Vertiefend wenn IT-DL: Austausch mit EVU

- Betreuen Sie unterschiedliche EVU?
- Wie sieht die Zusammenarbeit mit diesen aus?
- Wie wird hauptsächlich kommuniziert?
- Inwiefern arbeiten Sie mit deren IT-Abteilung zusammen?
- Inwiefern arbeiten Sie mit Anwendenden direkt zusammen?
- Mit wem arbeiten Sie sonst noch zusammen?
- Welche Rolle spielen regelmäßige Treffen?
- Wie viele Mitarbeiter sind dort und welche Aufgaben erledigen sie?
- Inwiefern spielen die Verträge eine Rolle bei der Zusammenarbeit?
- Inwiefern spielen Berechtigungen eine Rolle?
- Inwiefern spielen die Führungskräfte eine Rolle bei der Zusammenarbeit?
- Wie würden Sie die Beziehung zu den EVU beschreiben?
- Inwiefern spielt eine gute persönliche Beziehung eine Rolle?
- Wie könnte die Zusammenarbeit verbessert werden?
- Inwiefern haben EVU fachliches Wissen?
- Inwiefern haben EVU Systemwissen (EDM)?
- Inwiefern haben EVU Wissen zur Regulierung?
- Inwiefern haben EVU Wissen zu Softwareentwicklung?
- Inwiefern findet ein Wissensaustausch statt?
- Inwiefern gibt es Interessengegensätze?
- Inwiefern gibt es Konfliktlösungsmechanismen?
- Wie können Kunden Verbesserungsvorschläge einbringen?
- Inwiefern ist Software für Kunden individualisiert?
- Wer übernimmt die Projektleitung bei Projekten zusammen mit den Kunden?
- Wie laufen die Tests ab?
- Wie läuft die Produktivsetzung ab?
- Wie ist Betrieb/Wartung organisiert? DevOps?
- Inwiefern ist Support und Entwicklung/Projekte getrennt?
- Wie werden Anwender über Änderungen informiert?

Vertiefend wenn EVU: IT Org

- Haben Sie eine hausinterne IT-Abteilung?
  - Was macht die?
  - Wie viele Mitarbeiter?
  - Auch Programmierer?
  - Anwendungsbetreuer?
  - Wie viele SAP-Experten? (die hohes Systemwissen haben, d.h. bspw. customizen können)

- Inwiefern sammelt die interne IT Anforderungen und gibt sie an den IT-DL weiter?
- Oder haben die Fachbereiche selber Kontakt IT-DL?
- Inwiefern arbeiten Sie mit der IT zusammen?
- Werden IT-Projekte von der IT-Abteilung durchgeführt oder gibt es ein separates Projektteam?
- Wer kümmert sich um die Architektur der IT?
- Wer kümmert sich um das Thema IT-Sicherheit?
- Inwiefern spielen externe Arbeitskräfte eine Rolle (Freelancer, Berater etc.)?
- Inwiefern werden Sie als Externer anders eingebunden als Interne?
- Haben Sie andere Qualifikationsprofile?
- Erledigen Sie andere Aufgaben?
- Inwiefern findet ein Wissensaustausch statt?
- Wie werden Anforderungen an die IT übergeben bzw. gesammelt?
- Inwiefern spielt ein Ticketsystem eine Rolle?
- Inwiefern spielen E Mails zusätzlich noch eine Rolle?
- Oder gibt es noch andere Kommunikationskanäle? (Sharepoint etc.)
- Inwiefern spielen regelmäßige Treffen eine Rolle?
- Inwiefern werden diese von Ihnen beauftragt?
- Gab es eine Lernkurve in puncto Zusammenarbeit?
- Inwiefern gibt es unterschiedliche Erwartungen?
- Inwiefern gibt es Konfliktlösungsmechanismen?
- Welche Methoden werden eingesetzt bei der Zusammenarbeit mit der IT/Fachbereich? (Kanban?)
- Gibt es eine Digitalisierungsstrategie? Wie sieht diese aus?
- Inwiefern gibt es eine strategische Abstimmung zwischen IT-Abteilung und den Fachbereichen?
- Wie viele Anwender hat IS-U bei Ihnen?
- Inwiefern haben diese unterschiedliche Berechtigungen?
- Verwenden Sie noch andere Softwarepakete für die energiewirtschaftlichen Prozesse?

Vertiefend Softwareentwicklung/IT:

- Haben Sie auch etwas mit den Programmierenden zu tun?
- Inwiefern ist der Entwicklungsprozess formalisiert?
- Gibt es Richtlinien für die Quellcode-Codierung, die Kommentierung?
- Wer kümmert sich um die Architektur der IT?
- Wie werden Anforderungen an die IT übergeben bzw. gesammelt?
- Gibt es ein festes IT-Budget für Sie bzw. Ihre Abteilung?
- Inwiefern spielt ein Ticketsystem eine Rolle?
- Inwiefern spielen E Mails zusätzlich noch eine Rolle?
- Oder gibt es noch andere Kommunikationskanäle? (Sharepoint etc.)
- Welche Rolle spielen regelmäßige Treffen?
- Wie viele Mitarbeiter sind dort und welche Aufgaben erledigen sie?
- Was waren größere Projekte in den letzten Jahren, bei denen die IT involviert war?

- Inwiefern spielen die Verträge eine Rolle bei der Zusammenarbeit?
- Inwiefern spielen die Führungskräfte eine Rolle bei der Zusammenarbeit?
- Wie würden Sie die Beziehung zur IT beschreiben?
- Inwiefern spielt eine gute persönliche Beziehung eine Rolle?
- Wie würden Sie die Zusammenarbeit mit der IT beschreiben?
- Wie könnte in der Zusammenarbeit mit der IT-Abteilung verbessert werden?
- Inwiefern hat die IT fachliches Wissen?
- Inwiefern findet ein Wissensaustausch statt?
- Inwiefern gibt es Interessengegensätze?
- Inwiefern gibt es Konfliktlösungsmechanismen?
- Wie wird diese betreut bzw. weiterentwickelt?
- Inwiefern spielen externe Arbeitskräfte eine Rolle (Freelancer, Berater etc.)?
- Inwiefern werden diese von Ihnen beauftragt?
- Inwiefern werden Sie als Externer anders eingebunden als Interne?
- Haben Sie andere Qualifikationsprofile?
- Erledigen Sie andere Aufgaben?
- Inwiefern findet ein Wissensaustausch statt?
- Welche Methoden werden eingesetzt bei der Zusammenarbeit mit der IT/Fachbereich? (Kanban?)

Vertiefend Wissen/Qualifikation

- Was waren die letzten Schulungen, die Sie besucht haben?
- Inwiefern war die Schulung in IPMA/PRINCE2 hilfreich?
- Inwiefern spielt fachliches Wissen eine Rolle? (Prozesse, Branche, Organisation)
- Inwiefern wäre mehr Wissen über die Branche hilfreich?
- Inwiefern wäre mehr Wissen über die fachlichen Prozesse hilfreich?
- Inwiefern spielt IT-Wissen eine Rolle? (Software, Architektur, Entwicklung)
- Arbeiten Sie selber mit SAP? Inwiefern wäre weiteres SAP-Wissen hilfreich?
- Welches Wissen oder Qualifikation wäre für Ihre Arbeit noch hilfreich?
- Wie ist das Verhältnis von IT- und energiewirtschaftlichem Wissen in der Arbeit: Was nimmt welchen Anteil ein?
- Welche Rolle spielt Learning by Doing?

Vertiefend falls vorhanden: Externe

- Inwiefern arbeiten Sie mit SAP/Softwareanbieter zusammen?
- Inwiefern nehmen Sie eine Lernkurve in puncto Zusammenarbeit mit Externen wahr?
- Inwiefern nehmen Sie eine Lernkurve bei sich selbst wahr in puncto Zusammenarbeit?
- Inwiefern spielen die Verträge eine Rolle bei der Zusammenarbeit?
- Inwiefern spielen die Führungskräfte eine Rolle bei der Zusammenarbeit?

- Inwiefern spielt Fakturieren eine Rolle?
- Inwiefern spielt es eine Rolle, von den Externen etwas zu lernen?
- Inwiefern spielt eine gute persönliche Beziehung eine Rolle?
- Inwiefern spielt »auf Augenhöhe sein« eine Rolle?
- Gab es mal ein größeres Projekt zur Softwareentwicklung/Anpassung, wo Dritte beteiligt waren?
- Zu den Dritten: Kannte man sich vorher? Was haben die gemacht? Welches Wissen haben die eingebracht? Inwiefern war diese Zusammenarbeit anders als andere? Wie sahen die Verträge aus?
- Was war der Auslöser für die Entwicklung/Anpassung?
- Gibt es neben Projekten auch noch andere Formen der Zusammenarbeit mit Kunden?
- Welche Rolle spielen regelmäßige Treffen?
- Welche Rolle spielen die Fachbereiche?
- Welche Rolle spielt die IT?
- Inwiefern findet ein Wissensaustausch statt?
- Welche Rolle spielt das Wissen über die Firmen und deren Prozesse?
- Inwiefern spielt branchenspezifisches Wissen bei der Entwicklung von IT-Lösungen (Regulierung, Politik etc.) eine Rolle?
- Inwiefern gibt es Unterschiede zwischen Netz und Vertrieb beim Einsatz von EDM? (Anreizregulierung…)
- Wie könnte die Zusammenarbeit verbessert werden?

Falls beteiligt: Projekt XY

- Was ist/war Ihre Rolle in dem Projekt?
- Was war der Auslöser für die Entwicklung/Anpassung?
- Welche Zielvorgaben gibt es?
- Welche Funktion soll die Software erfüllen?
- Wer sind die Anwendenden der Software?
- Wie integriert sich die Lösung in die bestehende IT-Landschaft?
- Handelt es sich um eine Eigenentwicklung oder um die Anpassung einer Standardsoftware?
- Inwiefern macht das aus Ihrer Sicht einen Unterschied?
- Wie ist das Projekt bisher verlaufen?
- Was ist eine besondere Herausforderung bei dem Projekt?
- Welche Rollen spielen Vorgaben für die Projektarbeit? (Handbücher etc.)
- Welche Rollen spielen bestimmte Methoden für die Projektarbeit?
- Inwiefern spielen Quality Gates oder Meilensteine eine Rolle?
- Inwiefern war die Planung detailliert?
- Gibt es ein Budget und feste Deadlines?
- Gibt es für die einzelnen Teilnehmer des Projektes Ziele?
- Welche Methoden werden eingesetzt? (Scrum etc.)

- Welche Werkzeuge? (Jira etc.)
- Wer hat alles mitgearbeitet?
- Welche Rolle hat der Fachbereich gespielt?
- Inwiefern haben Sie mit der Netze-IT zusammengearbeitet?
- Inwiefern haben Sie mit der Konzern-IT zusammengearbeitet?
- Welche Rolle hat die externe Softwarefirma/IT-DL etc. gespielt?
- Wie wurden die Teilnehmer ausgewählt?
- Wie lief die Projektkoordination ab?
- Gibt es regelmäßige Treffen, Veranstaltungen o. ä.?
- An welchen Treffen oder an welchen Austauschen nehmen Sie teil?
- Nehmen Sie an den Stand-ups teil?
- Wie laufen diese Treffen ab?
- Inwiefern tauschen sich die Mitarbeiter sonst noch aus?
- Wie wurden Aufgaben verteilt?
- Wie ist Arbeit organisiert? (Autonom etc.)
- Waren alle an einem Ort?
- Arbeiten die Projekt-Teilnehmer noch woanders mit?
- Welche Rolle spielte IT-Wissen bei der Zusammensetzung der Beschäftigten?
- Waren unterschiedliche Qualifikationsprofile involviert?
- Wer brachte welches Wissen ein?
- Welche Rolle haben externe Mitarbeiter gespielt?
- Kannten man sich vorher?
- Was haben die gemacht?
- Welches Wissen haben die eingebracht?
- Inwiefern war diese Zusammenarbeit anders als andere?
- Wie sahen die Verträge aus?
- Gibt es Regeln für die Zusammenarbeit?
- Inwiefern spielt informeller Austausch eine Rolle?
- Inwiefern kommt es zu einem Wissenstransfer?
- Wie unterschied sich die Zusammenarbeit zwischen Internen und Externen?
- Gibt es regelmäßigen Austausch?
- Wie war die Kommunikation organisiert?
- Was waren zentrale Herausforderungen in der Zusammenarbeit mit den Externen?
- Was war anders als bei anderen Kooperationen?
- Welche Rolle spielen die Führungskräfte? (wenn TN für andere Wertströme, Projekte etc. arbeiten)
- Inwiefern spielte zentrales Controlling eine Rolle?
- Welche Softwareentwicklungsmethoden, Programmiersprachen und Entwicklungsumgebung wurden eingesetzt?
- Welche Regeln gibt es für die Programmierung?
- Gibt es Richtlinien für die Quellcode-Codierung, die Kommentierung?
- Inwiefern wird die Softwarequalität kontrolliert?
- Ist der Entwicklungsprozess formalisiert?

- Wie kam es zu der Softwarearchitektur?
- Was waren die Prioritäten? (Wartbark., Sicherheit, Performance, schnelle Ums., Flexibilität etc.)
- Inwiefern müssen Sie sich mit IT-Architekten in der IT abstimmen?
- Wie werden Daten und Datenflüsse integriert?
- Inwiefern wäre auch eine andere Lösung möglich gewesen?
- Inwiefern waren Anwendende an der Entwicklung beteiligt?
- Wer wurde sonst alles mit eingebunden (Stakeholder)?
- Wie laufen die Tests ab?
- Werden automatisierte Tests oder Unit-Tests eingesetzt?
- Wie läuft Produktivsetzung ab?
- Wie ist Betrieb/Wartung organisiert? (DevOps?)
- Wie wurde dokumentiert?
- Wie hat sich die Arbeit durch die Lösung verändert? (Verlagerung Tätigkeiten, Wissenstransfer etc.)
- Wurden die betroffenen Bereiche reorganisiert?
- Sind neue Tätigkeiten entstanden?
- Welche Rolle spielt Automatisierung?
- Inwiefern wird das Projekt evaluiert?

Abschluss

- Wie sehen mögliche Karrierewege für Sie aus?
- Wie würden Sie Ihre Arbeitsbelastung beschreiben?
- Welche Rolle haben Betriebsräte und Gewerkschaften bei dem Thema Kooperationen und Digitalisierung ge-spielt?
- Gibt es aus Ihrer Sicht noch etwas Wichtiges, was wir noch nicht angesprochen haben?
- Kennen Sie noch jemanden, der sich für ein Interview zur Verfügung stellen würde?