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-

---

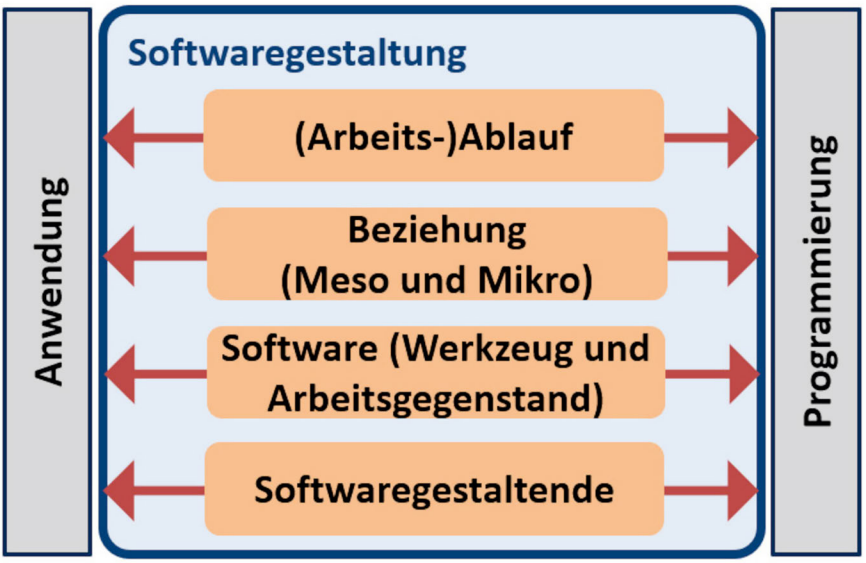
und reflexive Nachdenken, wie diese Kommunikation zu gestalten ist (vgl. ebd. 295). Seine befragten Personen verstehen sich allesamt als Dolmetschende zwischen zwei Welten (vgl. Kruse 2004: 315). Sie realisieren ihre Arbeitsvollzüge durch Kommunikation (vgl. ebd. 324). Für sie ist ein kommunikativer 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.