9. Ziel der Untersuchung, wesentliche Befunde und weiterführende Fragestellungen

Ziel der vorliegenden Arbeit war es, die Formen und Folgen von Softwaregestaltung in der Energiewirtschaft zu untersuchen. Die Sichtung der bisherigen Forschung hat gezeigt, dass es für den Arbeitsprozess der Softwaregestaltung und die Arbeit der Softwaregestaltenden noch kein passendes Kontrollkonzept gibt. Zudem sind die Folgen unterschiedlicher Formen von Softwaregestaltung für die Softwaregestaltenden und die Arbeit und Organisation der Anwendung bisher überwiegend vernachlässigt worden. Die Fallstudien stellen die Bedeutung der Softwaregestaltung für Software, Arbeit und Organisation in den EVU und in der Branche dar. Die sich aus der Untersuchung ergebenden zentralen Befunde knüpfen an Debatten zur Kontrolle von (IT-)Wissensarbeit und zur digitalen Transformation in der Arbeits- und Industriesoziologie an. Bevor das Schlusskapitel auf die beiden Debatten im Einzelnen eingeht, fasst der nächste Punkt zunächst noch einmal kurz zusammen, warum Softwaregestaltung für die Forschung relevant ist.

9.1. Softwaregestaltung: ein wenig erforschter Arbeitsprozess der Digitalisierung

Industriespezifische Software spielt in Firmen sämtlicher Branchen eine immer größere Rolle (Kapitel 4.1). In softwareintensiven Industrien, in denen Software nicht das Produkt ist, ermöglicht sie gesteigerte Effizienz, Qualität und Individualität von Produkten und Dienstleistungen.

Um die Möglichkeiten der Softwareentwicklung in industriespezifischen Anwendungsbereichen zu nutzen, ist Softwaregestaltung zentral. Das ist jener Teil der Entwicklung, der erarbeitet, welche Anforderungen Programmierende zu programmieren haben. Mit Softwaregestaltung schaffen Organisationen die Voraussetzungen für industriespezifische Software, indem sie zwei Kernprobleme lösen (Kapitel 4.2):

- Erstens das Problem der **softwaretechnischen Interdisziplinarität**, weil im Zuge der Softwaregestaltung die Beteiligten die softwaretechnischen Möglichkeiten mit den Bedarfen einer Branche bzw. eines branchenspezifischen Anwendungsbereiches abstimmen müssen.
- Zweitens das Problem der **softwaretechnischen Gestaltungsmöglichkeiten**, indem a) Organisationen sich entweder auf die Softwaregestaltung oder die Anwendung einer Standardsoftware ausrichten und b) die Softwaregestaltung den softwaretechnischen Zuschnitt (individuell/Standard) erarbeitet.

In manchen Fällen wählen Unternehmen eine industriespezifische Standardsoftware einer Softwarefirma. Andere Unternehmen entscheiden sich für den Primat der Softwareentwicklung, der für digitale Start-ups typisch ist. Sie arbeiten seit ihrer Gründung softwaretechnisch interdisziplinär und sind von Anfang an organisatorisch auf die Gestaltung einer individuellen Software ausgerichtet.

Damit Organisationen die Möglichkeiten der Softwaregestaltung nützen können, müssen sie deren Arbeit kontrollieren. Um die Kontrolle der Softwaregestaltung zu untersuchen, ist ein passendes Konzept notwendig, das die Eigenarten der Softwaregestaltungsarbeit berücksichtigt. Was zeichnet die Softwaregestaltung als Arbeit aus? Sie basiert auf Wissen und Kommunikation und ist auf Kooperation angewiesen (5.1). Das liegt daran, dass ihre materielle Basis Software ist. Software ist eine zeichenbasierte Technologie, die aus mehreren technischen Schichten wie dem Quellcode oder der Anwendungsoberfläche besteht und für deren Entwicklung sprachliche Strukturierungsmittel wie Programmiersprachen oder Begriffe wie Architektur oder Modell genutzt werden. Zudem ändert sich Software, ist beliebig erweiterbar und kann hochkompliziert werden. Entsprechend ändert sich das Wissen über sie, kann stetig anwachsen und sehr umfangreich sein. Es ist Teil der Softwaregestaltung, sich auf eine sich verändernde, komplexer werdende Software einzulassen und sich gleichzeitig tiefergehend mit branchenspezifischen Themen auseinanderzusetzen. Dabei sind die Perspektiven unterschiedlicher Spezialist:innen zu berücksichtigen, weswegen nicht nur Wissen, sondern auch Kommunikation und Kooperation zentral sind. Die Beteiligten müssen sich auf eine Gestaltung einigen, auch wenn sie alle unterschiedliches Wissen und ihre eigene Perspektive auf die Software haben, die Biografie der Software besser oder schlechter kennen und sie nicht (alle) Teil der anwendenden Organisation sind.

Die Geschichte der Softwareentwicklung hat noch kein allgemeines Konzept zur Analyse der Softwaregestaltung hervorgebracht, das auf die untersuchten Fallstudien passen würde. Dabei ist die Softwaregestaltung in den letzten Jahrzehnten immer wichtiger geworden (Kapitel 5.2). Es gibt ein eigenes Forschungsgebiet zur Phase der Anforderungserarbeitung der Softwareentwicklung, wonach Kooperation, Kommunikation und soziale Kompetenzen wichtig sind, um eine gemeinsame Sprache und Übersetzungsfähigkeit zwischen IT- und Branchenfachleuten herzustellen (vgl. Alvarez 2002, Ross/Chiasson 2011, Kaminski 2012, Alsanoosy et al. 2020).

Die Forschung vor allem aus der Arbeits- und Techniksoziologie zeigt, dass die Organisationen in der Realität unterschiedliche Methoden (z.B. Scrum oder Projektarbeit) oder Kontrollformen (permissive oder direkte Kontrolle) für die Softwareentwicklung nutzen (vgl. Feuerstein 2012, Boes et al. 2018, Schulz-Schaeffer/Bottel 2018). Es gibt den

Versuch, mithilfe abstrakter Variablen wie intellektuellem und sozialem Kapital eine allgemeine Theorie der Softwareentwicklung zu schreiben (vgl. Wohlin et al. 2015). Allerdings vernachlässigt solch ein Ansatz strukturelle Faktoren wie eine unterschiedliche Arbeitsteilung zwischen Anwendung und Entwicklung (ob zwischen IT- und Fachabteilungen, EVU und IT-Dienstleistungsunternehmen) oder Konflikte zwischen beteiligten Gruppen wie Management und Softwareentwickelnden. Ganz zu schweigen davon, dass die in diesem Absatz zitierte Forschung nicht zwischen industriespezifischer und anderer Softwareentwicklung unterscheidet und sich nicht auf die Phase der Softwaregestaltung fokussiert.

Um geeignete, allgemeine Konzepte zur industriespezifischen Softwaregestaltung zu entwickeln, hat die vorliegende Untersuchung die Softwareentwicklung in der Energiewirtschaft erforscht und entsprechende Literatur ausgewertet. Durch Branchenanalyse und Fallstudien ist eine Betrachtung der Dynamik zwischen Individual- und Standardsoftwaregestaltung in der Branche möglich. Die Auswertung der Daten mit der Grounded Theory erlaubt es, den konkreten Arbeitsprozess der Softwaregestaltung zu analysieren und wesentliche Kategorien, entscheidende Kontextfaktoren sowie ihre Folgen herauszuarbeiten. Die Datengrundlage dafür ist die Befragung diverser Beschäftigtengruppen, deren jeweilige Perspektiven detaillierte Beschreibungen von Arbeit und Organisation der Softwaregestaltung liefern. Die gewonnenen Kategorien waren Ausgangspunkt für das Konzept der soziotechnischen Netzwerkarbeit und ergeben den Analyserahmen (8.1.3), der die sieben Fallstudien und deren Vergleich strukturiert. Er hilft, die Unterschiede herauszuarbeiten und Idealtypen zu konstruieren.

9.2. Erster Debattenbeitrag: Softwaregestaltung als soziotechnische Netzwerkarbeit

Wie kann eine konzeptionelle Beschreibung des Arbeitsprozesses der Softwaregestaltung und der Arbeit der Softwaregestaltenden aussehen, die sowohl die unterschiedlichen Konstellationen berücksichtigt, in denen sie stattfindet, als auch die Eigenheiten von kooperativer, kommunikationsintensiver und interdisziplinärer Wissensarbeit? Die Antwort auf diese Frage ist das Konzept der soziotechnischen Netzwerkarbeit.

Das 6. Kapitel hat ein erstes Konzept der soziotechnischen Netzwerkarbeit aus existierender Forschungsliteratur erarbeitet. Es stellt die theoretische Lösung des Transformationsproblems für die Softwaregestaltung dar. Es ermöglicht die begriffliche Beschreibung digitaler Arbeitsprozesse, die über verschiedene Organisationen, Teams oder Abteilungen verteilt sind und bei denen Subjektivität, Wissen, Kommunikation, Kooperation und Software im Mittelpunkt stehen (6.2). Das Besondere an dieser Kontrolle von Wissensarbeit ist ihr Netzwerkcharakter. Um zwischen Anwendung und Programmierung zu vermitteln, sind Subjekt, Betrieb und Organisationsnetzwerk gleichermaßen relevant. Weder die (Software-)Technik noch das Management sind dominierende Faktoren bei der Kontrolle der Softwaregestaltung. Das 6. Kapitel hat noch vereinfacht zwischen vier Ebenen unterschieden. Sie zeigen sich exemplarisch bei IT-Projekten wie ERP-Implementationen (6.3) und wirken zusammen, um die Transformation von Arbeitskraft zu gewährleisten (6.4). Aus Mangel an spezifischer Literatur

zur Softwaregestaltung hat das 6. Kapitel mehr allgemeine Literatur zu Wissensarbeit, Arbeit in Netzwerken und mit Software und IT herangezogen, um das Konzept der soziotechnischen Netzwerkarbeit zu entwickeln.

Das Empirie-Kapitel entwickelt das Konzept der **soziotechnischen Netzwerkarbeit** weiter. Es besteht dort aus dem **Arbeitsprozess der Softwaregestaltung** und der **Arbeit der Softwaregestaltenden**, wobei es auch die dazugehörigen Kategorien differenzierter herausarbeitet. Entsprechend grenzt der aus der Empirie entwickelte Analyserahmen klar ab: zwischen soziotechnischer Konstellation, Arbeitsprozess und Arbeit der Softwaregestaltenden.

So ist es dann auch möglich, die **typischen Unterschiede** in den Fallstudien bei der soziotechnischen Netzwerkarbeit zu konstruieren: Entweder liegt ein **dezentraler** oder **zentraler** Arbeitsprozess vor, arbeiten die Softwaregestaltenden in einer **Matrixorganisation** oder einer **reinen Netzwerkorganisation**. Alte Kontrollunterschiede wie direkte und permissive Kontrolle (vgl. Friedman 1977) oder industrialisiert und nicht-industrialisiert (vgl. Boes et al. 2018) passen hingegen nicht zur soziotechnischen Netzwerkarbeit.

Zudem zeigt sich die Kontingenz der Softwaregestaltung in Bezug auf die **soziotechnische Konstellation** (ausführlicher unter 8.6.2.1). Sie ist die Ausgangsbedingung für die Softwaregestaltung. Zugleich stellt sie auch deren Grenzen dar, in denen sich die Möglichkeiten der Softwaretechnik verwirklichen lassen. Je Fallstudie ist es vom **Anwendungsbereich** abhängig, welche Möglichkeiten für die Softwaregestaltung bestehen. Das betrifft den Anteil der Datenverarbeitung im Anwendungsbereich und ob sie einen gesamten Prozess gestalten kann oder nur Teile davon. Aus Sicht der Softwaregestaltung wird die **Arbeitsteilung** zwischen Anwendung und Programmierung zur Wissensgrenze, die überwunden werden muss. Sie kann innerhalb oder zwischen Organisationen bestehen. Als primäre Formen der **Grundkoordination** zeigen sich in den Fällen Markt, Hierarchie oder Netzwerk. Egal welche (Mischung der) Grundkoordination vorliegt, müssen die Beteiligten immer einen Weg finden, um für die Softwaregestaltung zusammenzuarbeiten. Um von einer Netzwerkkoordination zu sprechen, muss die Zusammenarbeit ungehindert von hierarchisch-formalen Wegen, vertraglichen Einschränkungen oder Marktkalkülen zwischen Anwendung und Programmierung möglich sein (wie bei den Fallstudien STARTUP und KOOP3). Die **Softwarearchitektur** hat sich in allen Fällen als prägend für die Softwaregestaltung erwiesen: Erstens entscheidet ihre Aufteilung z.B. in Module darüber, wie sich die Arbeit innerhalb und zwischen den Organisationen verteilt (vor allem für die Programmierenden). Zweitens prägt die Softwarearchitektur in der Mehrzahl der Fälle durch ihren Zuschnitt (individuell, Standard) und die daraus resultierenden Abhängigkeiten den Kommunikationsaufwand und welche Kommunikationswege zwischen Anwendung und Entwicklung existieren.

Dieser Abschnitt stellt zunächst die aus den Unterschieden zwischen den Fallstudien erarbeiteten Typen der soziotechnischen Netzwerkarbeit vor (8.6.1.1), um dann genauer auf die einzelnen Kategorien einzugehen (8.6.2) und abschließend den Beitrag zur Debatte über die Kontrolle von Wissensarbeit in der Arbeitssoziologie zu vertiefen.

9.2.1. Typische Unterschiede in der soziotechnischen Netzwerkarbeit

Die Unterschiede zwischen den Fallstudien im Arbeitsprozess der Softwaregestaltung lassen sich zu Typen zusammenfassen. Wobei es sich um netzwerkspezifische Unterschiede in der Kontrolle des Arbeitsprozesses und der Arbeit der Softwaregestaltenden handelt:

1. Der Arbeitsprozess der Softwaregestaltung findet entweder dezentral statt, z.B. in einem Team für eine individuelle Softwaregestaltung oder zentralisiert in einer Organisation wie einer Softwarefirma, die ein Standardprodukt gestaltet (8.3.1).
2. Die Softwaregestaltenden arbeiten in einer Matrixorganisation, in der sie für ihre Arbeit Hindernisse überwinden müssen, oder in einer reinen Netzwerkorganisation, die ihnen ihre Arbeit einfacher macht, weil dort weder Hierarchien noch Marktbeziehungen die Zusammenarbeit zwischen Anwendung und Programmierung stören (8.4.1).

Die vier Grundtypen der soziotechnischen Netzwerkarbeit sind von der soziotechnischen Konstellation abhängig, die sich aus den typischen Unterschieden der soziotechnischen Netzwerkarbeit ergeben (8.6.1.1): KOORDINIEREND (zentral in einer Matrixorganisation), DIREKT (dezentral in einer reinen Netzwerkorganisation), ERGÄNZEND (dezentral in einer Matrixorganisation) und VIRTUELL (zentral in einer reinen Netzwerkorganisation).

Der Typ **KOORDINIEREND** ist ein **zentralisierter** Arbeitsprozess der Softwaregestaltung in einer **Matrixorganisation**. Wie für einen zentralen Arbeitsprozess typisch, konzentrieren sich **Rollen** wie Anforderungsmanagende auf die Koordination der Anforderungsaufnahme in unterschiedlichen Organisationen. Es existiert ein **Ablauf** in Form eines Anforderungsmanagements, in dem die Beteiligten über Anforderungen verhandeln und Konflikte lösen. **Software**lösungen wie Ticketsysteme dienen dazu, Transparenz zwischen den Organisationen herzustellen und Abstimmungen zu organisieren. Die **kommunikativen Beziehungen** müssen die Spannungen zwischen den unterschiedlichen Interessen ausgleichen können und die teils bürokratischen Abläufe ergänzen, z.B. indem auf persönlicher Ebene ein direkter Austausch stattfindet. Indem softwaregestaltende Organisationen wie IT-DL auf die Softwaregestaltung spezialisiert sind, können sie ein **Beschäftigungssystem** etablieren, das für eine entsprechende Auslastung sorgt. Sie können z.B. den Wünschen der Softwaregestaltenden nachkommen und ihnen Karrieren unabhängig von Hierarchien ermöglichen (z.B. eine Karriere als Fachexpert:in ohne personelle Verantwortung zu machen und auf diesem Wege in eine höhere Vergütungsgruppe zu kommen). Aufgrund der Marktbeziehung zwischen z.B. IT-DL und EVU **kontrollieren** Letztere die Arbeit der Softwaregestaltenden des IT-DL. In Hierarchien sind es die Führungskräfte, welche die Arbeit der Softwaregestaltenden kontrollieren. Zudem führt die Matrixorganisation aufgrund der **Wissensverteilung** zu einer Abhängigkeit, weil eine stetige softwaretechnische Interdisziplinarität fehlt: Einerseits fehlt den EVU bzw. einzelnen Abteilungen das Wissen, um selbst die Möglichkeiten der Softwaregestaltung einschätzen zu können. Andererseits fehlt der programmierenden Abteilung oder Organisation Wissen über die Anwendungsbereiche.

KOOP1, KOOP2 und KOOP2 sind nur zum Teil Beispiele für diesen Typ, weil einzelne Kategorien anders ausgeprägt sind.

Charakteristisch für den Typ **DIREKT** ist ein **dezentraler** Arbeitsprozess der Softwaregestaltung in einer **reinen Netzwerkorganisation**. Typische Merkmale einer dezentralen Softwaregestaltung sind **Rollen**, die direkt Anforderungen von Anwendenden aufnehmen. Die Rollen sind Teil eines **Ablaufs**, der unterschiedliche Methoden wie Scrum, Resonanzgruppen und Workshops nutzt, um Anforderungen direkt aufzunehmen, auszuarbeiten und den Programmierenden übergeben zu können. Die **kommunikativen Beziehungen** sind offen, direkt, langfristig und tief verankert im Anwendungsbereich, nicht getrennt durch Abteilungsgrenzen und Hierarchien und bei Bedarf mit engem Kontakt zu den Anwendenden. Die verwendeten **Werkzeuge** wie Ticketsystem und Chat-Software erlauben es Beteiligten, dezentral Anforderungen ohne bürokratische Hindernisse (wie z.B. fehlende Berechtigungen oder eine formal vorgegebene Abstimmung mit der Führungskraft) aufzunehmen. Ein typisches Element der Arbeit in einem reinen Netzwerk ist ein **Beschäftigungssystem**, das es den Mitarbeitenden erlaubt, in der rollenbasierten Organisation interessante und für Softwaregestaltende adäquate Aufgaben zu bekommen. Es geht nicht darum, in einer Hierarchie aufzusteigen (die es formal auch gar nicht gibt). Es ist eine Organisation, die auf die Softwaregestaltung ausgerichtet ist. Weil dem so ist, findet z.B. die Softwaregestaltung nicht nur temporär statt. Die **Kontrolle** passiert im Wesentlichen horizontal durch die Kollegenschaft und ermöglicht Eigenmotivation, individuelle Steuerung der Arbeitsbelastung und selbstständiges Arbeiten (z.B. wie es Scrum vorsieht). Durch den kontinuierlichen, iterativen Austausch im Netzwerk für die Softwaregestaltung ist eine gemeinsame **Wissensbasis** gesichert. Weil das Wissen nicht hierarchisch oder über einen Markt verteilt ist, bestehen keine Hindernisse, um an es heranzukommen.

Der Typ **ERGÄNZEND** kombiniert einen dezentralen Arbeitsprozess mit einer Matrixorganisation. Bei **VIRTUELLL** existiert ein zentraler Arbeitsprozess der Softwaregestaltung in einer reinen Netzwerkorganisation. Die beiden Typen sind hier nicht ausführlicher vorgestellt, weil sich die bei den beiden Typen oben ausgeführten idealtypischen Eigenschaften wiederholen – nur eben anders kombiniert. Zudem ist keine der Fallstudien vom Typ VIRTUELL und es wäre noch eine weitere Untersuchung mit Fallstudien zu dieser Form industriespezifischer Softwaregestaltung notwendig.

Die vier Idealtypen unterstreichen nochmal, dass es abhängig von der Konstellation ist, welche Rolle Management und Führungskräfte bei der Softwaregestaltung spielen. So haben sie in der Aufbauorganisation einer Matrixorganisation ihre feste Position mit entsprechenden Entscheidungsbefugnissen, Ressourcen und den Fokus auf die Ergebniskontrolle. In den Fallstudien zeigt sich allerdings, dass sie auch dort mal mehr und mal weniger an der Softwaregestaltung teilnehmen.

9.2.2. Gemeinsame Kategorien der soziotechnischen Netzwerkarbeit

Die soziotechnische Netzwerkarbeit besteht aus dem Arbeitsprozess der Softwaregestaltung und der Arbeit der Softwaregestaltenden. Für beide konnte die Analyse der Fallstudien die zentralen Kategorien herausarbeiten.

9.2.2.1. Arbeitsprozess der Softwaregestaltung

Die Transformation der Arbeitskraft im Arbeitsprozess der Softwaregestaltung basiert auf den Rollen, einem Ablauf, kommunikativen Beziehungen, digitalen Werkzeugen und dem softwaretechnischen Zuschnitt.

Das Konzept der **Rolle** hat sich als nützlich erwiesen, dem Netzwerkcharakter der Kontrolle von Softwaregestaltungsarbeit gerecht zu werden. Weil Rollen das Handeln durch Erwartungen leiten und nicht die konkrete Arbeitsausführung festlegen, erlauben sie einen eigenständigen, situativ passenden Beitrag der Einzelnen und dass diese sich je nach Konstellation der Arbeitsteilung zwischen Anwendung und Programmierung einfügen. Es gibt zwar feste Rollen wie Product Owner:in oder IT-Projektleitung, denen in einigen Fällen Schulungen zugrunde liegen. In der Praxis passen die Beschäftigten ihre Rolle in Bezug zu den anderen Beteiligten aber an. Für viele, die bei der Softwaregestaltung mitmachen, gehört es zum Arbeitsalltag, mehrere und wechselnde Rollen einzunehmen. Zuletzt sind die Erwartungen nicht abhängig von Hierarchien, weshalb sie von diesen unabhängig das Handeln leiten können. Neben den Erwartungen an spezifische Rolle wie z.B. IT-Projektleitung kontrollieren allgemeine Erwartungen die Arbeit im Arbeitsprozess, wie das 6. Kapitel gezeigt hat: kooperativ sein, selbstorganisiert arbeiten, aktiv mit Softwareobjekten interagieren und sich ausgehend von ihnen zu koordinieren, mit Nicht-Wissen umgehen können und sich im Netzwerk bewegen. Subjektivierung (vgl. Minssen 2011) und subjektivierendes Arbeitshandeln (vgl. Böhle 2010, Bolte 2017a, Bolte 2017b, Weishaupt/Hösl 2017) sind Kernbestandteile der Kontrolle der Arbeitskraft durch solche Rollen.

Der Netzwerkcharakter des **Ablaufs** zeigt sich allgemein bei IT-Projekten, die Organisationen in ganz unterschiedlichen Kontexten einsetzen. Dabei ist der Grad der Formalisierung sehr unterschiedlich (vgl. Heidling 2018: 224) und nicht entscheidend für die Kontrolle von Arbeit. Vielmehr ist wichtig, dass selbst bei stärkerer Formalisierung Handlungs- und Entscheidungsspielräume für einzelne Projektmitarbeitende bestehen (vgl. Kalkowski/Mickler 2005) und während und nach ERP-Implementierungen ein abteilungsübergreifendes Gestaltungsnetzwerk vorhanden ist, auf dem der Ablauf basiert (vgl. Hohlmann 2007: 353). In den Fallstudien trägt der Ablauf zur Netzwerkarbeit bei, indem er für die Kooperation (ob temporär oder langfristig) und die Kommunikation sorgt und beides nicht z.B. ökonomische Kalküle oder einzelne Führungskräfte untergräbt. Das zeigt sich an drei wesentlichen Aspekten: A) Der Ablauf schafft Möglichkeiten für Feedback(-Schleifen) zwischen Softwareanwendung, -gestaltung und -programmierung durch den Einsatz diverser Methoden – ob durch Treffen, E-Mails, Tests, Chat-Programme, persönliche Gespräche oder Methoden wie Prototyping oder Resonanzgruppen. B) Er bindet situativ die im organisationalen Netzwerk verteilten Beteiligten z.B. über Workshops ein. C) Er pflegt die Beziehungen durch regelmäßige Treffen, Mechanismen zum Abgleich von Erwartungen und zur Lösung von Konflikten. In den Fallstudien passiert dies z.B. durch Strategie-Treffen zwischen IT-Dienstleistungsunternehmen (IT-DL) und EVU, Anforderungsrunden mehrerer Fachbereiche eines EVU, durch Arbeitskreise einer Softwarefirma mit ihrer Kundschaft (EVU) oder Projektlenkungskreise. Grundlage für den Ablauf können unterschiedliche Methoden wie Scrum, Projektarbeit oder Anforderungsmanagement sein.

Neben einem Ablauf zeichnet sich die soziotechnische Netzwerkarbeit der Softwaregestaltung durch **kommunikative Beziehungen** zwischen Organisation(seinheiten) und Einzelpersonen aus. Wie schon die Forschung festgestellt hat, reichen für eine kooperative Zusammenarbeit z.B. Projektstrukturen allein nicht aus (vgl. Rüegg-Stürm/Young 2001) und informelle Strukturen der Kooperation sind wichtig (vgl. Bolte/Porschen 2007). Zudem helfen Beziehungen, die längerfristig sind, weil Vertrauen erst mit der Zeit aus reziproken Beziehungen und durch geteilte Interessen entsteht (vgl. Powell 1990, Uzzi 1997).

In den Fallstudien wird deutlich, welchen Beitrag interpersonale, kommunikative Beziehungen zur Netzwerkarbeit leisten und dass hierfür technische Lösungen wie Ticketsysteme allein nicht ausreichen. Durch kommunikative Beziehungen, die partnerschaftlich, kooperativ, auf Augenhöhe, offen oder von einem Geben und Nehmen geprägt sind, können Softwaregestaltende reine Marktbeziehungen und Barrieren innerhalb von Organisationen (seien es Hierarchien oder Abteilungs- und Teamgrenzen) überwinden. Weil es bei der Softwaregestaltung darum geht, ein gemeinsames Verständnis über energiewirtschaftliche Bedarfe und softwaretechnische Möglichkeiten zu sichern, ist die Kommunikationskompetenz wichtig. Der Mediator in einer Fallstudie zeigt die Verschränkung von Beziehungs- und Kommunikationskompetenz exemplarisch: Er arbeitet an Beziehungen, z.B. indem er Konflikte löst, sorgt für den kommunikativen Austausch und hat auch noch das fachliche Vokabular und Wissen, um sich verständlich machen zu können und für ein gemeinsames Verständnis unter den Beteiligten zu sorgen. Rollen wie Scrum Master:innen kümmern sich ebenso um die zwischenmenschlichen Beziehungen. Bei einem IT-DL ist die Rolle Key Account Management dafür zuständig, die Beziehung zu den EVU zu pflegen und bei Missverständnissen die inhaltliche Klärung herbeizuführen.

Neben diesen interpersonalen Beziehungen spielen auch jene zwischen Organisationen und Organisationseinheiten wie Abteilungen in den Fallstudien eine Rolle. Diese Beziehungen pflegt, wie oben beschrieben, auch der Ablauf auf unterschiedliche Weise. Darüber hinaus basiert die erfolgreiche längerfristige Zusammenarbeit zwischen IT-DL und EVU bei KOOP1 auf einem langjährigen Lernprozess. In den Arbeitsprozessen der Softwaregestaltung der Fallstudien sind die in der Forschung beschriebenen Kooperationshemmnisse oder -hindernisse (siehe 6.4.1) zwar Thema. Sie führen aber nicht zum Abbruch des Arbeitsprozesses, weil grundsätzlich die Kooperationsbereitschaft bestehen bleibt, auch wenn z.B. EVU überlegen, das IT-DL zu wechseln. Insofern ist der Forschung recht zu geben, dass für eine Zusammenarbeit kooperative Beziehungen bestehen müssen. Die Fallstudien zeigen, dass dies für ganz unterschiedliche Konstellationen von Arbeitsteilung und Grundkoordination zwischen Anwendung und Entwicklung gilt.

Neben Ablauf und Beziehungen basiert die Kontrolle der Arbeit im Arbeitsprozess der Softwaregestaltung auf der Software. Sie ist Arbeitsgegenstand und -mittel. Wie die Literatur zeigt (siehe 6.4.2), wirkt sie nicht nur einschränkend, sondern auch ermöglichend. Die zu gestaltende Software koordiniert als gemeinsam gestaltbares, zeichenbasiertes Bezugsobjekt die Wissensarbeit (vgl. Nicolini/Mengis/Swan 2012, vgl. Barrett/Oborn 2010, Carugati et al. 2018, Ponte/Rossi/Zamarian 2009, Bolici/Howison/Crowston 2009 und 2016). Zudem verlangen sie und die verwendeten Software-Werkzeuge wie Ti-

cketsysteme von den Anwendenden wissensevozierendes Engagement und Interaktion (vgl. Darr 2019, Rennstam 2012). Software schränkt die Softwaregestaltung ein, indem sie Eingabemöglichkeiten und Abläufe vorgibt (vgl. Kleemann/Matuschek 2008), digital in einen Prozess integriert (vgl. Sauer 2018) und soziale Strukturen wie Rollen oder Routinen softwaretechnisch stützt (vgl. Mutch 2010, Volkoff et al. 2007). Sie macht Arbeit transparent und bietet Möglichkeiten der Überwachung (vgl. Zuboff 1988). In der Softwareentwicklung geben die softwarebasierten Werkzeuge keinen (software)maschinellen Takt vor, sondern menschliche Intervention macht den digital-basierten Arbeitsprozess aus (vgl. Andrews et al. 2005, Barrett 2005).

In den Fallstudien ermöglichen Softwarelösungen erstens als **digitale Werkzeuge** der Softwaregestaltung die verteilte Teilnahme durch die Ein- und Ausgabe von Daten: ob Ticketsysteme, Entwicklungs- und Testumgebungen von Softwarefirmen wie SAP, Chat-Gruppen oder (geteilte) Excel-Dateien. Zudem stellen die Software und die dahinterliegenden Datenbanken den zentralen Wissensspeicher für die in der Softwaregestaltung arbeitenden Personen dar. Ticketsysteme lassen es zu, dass Anwendende oder andere Beteiligte aus unterschiedlichen Organisationen und Abteilungen Anforderungen aufnehmen, wodurch sie zugleich die Softwaregestaltung dokumentieren und sie für andere nachvollziehbar machen. Im Quellcode selbst ist die Umsetzung dokumentiert und manchmal existieren auch noch separate, softwarebasierte Dokumentationswerkzeuge wie Wikis. Zweitens ist die Kontrollwirkung der softwarebasierten Werkzeuge abhängig von der soziotechnischen Konstellation. In Hierarchien sorgt Software für Transparenz (über Anforderungen, deren Bearbeitungsstatus etc.) für die Führungskraft, in Marktbeziehungen für die Kundschaft und in reinen Netzwerkorganisationen zwischen den gleichgestellten Peers. Auch der zeitliche Takt in der Softwaregestaltung ist konstellationsabhängig: Die Reaktionszeiten werden entweder durch die zwischen IT-DL und EVU vereinbarten SLA, durch Aufwandsschätzungen von Fachleuten wie IT-Beratenden, durch die von der Regulierung vorgegebenen Fristen oder durch die Dringlichkeit von Ad-hoc-Fehlerbehebungen bestimmt. Letztendlich nutzen die Organisationen der Fallstudien die Software weniger zur direkten Kontrolle von Arbeit. So ist es den Arbeitenden möglich, sich auf komplizierte Probleme einzulassen, sich mit der Software und fachlichen Fragen tiefergehend zu beschäftigen und schwierigere Texte bzw. Anforderungen und Software zu schreiben.

Die für Netzwerkarbeit typische verteilte Teilnahme hängt von der Softwarearchitektur ab und es ist Teil des Arbeitsprozesses der Softwaregestaltung, den **softwaretechnischen Zuschnitt** der Software und damit einen Teil der Softwarearchitektur zu verwaltet. In der Fallstudie KOOP1 entscheidet das Anforderungsmanagement für einzelne Anforderungen, ob das IT-DL sie als Teil des Standards umsetzt. Bei INTERN2 priorisiert die Anforderungsrunde, was das EVU individuell programmiert. Dabei zeigen die Fallstudien, dass die Architektur meist zentral vorgegeben ist (z.B. durch eine Softwarefirma). Zudem müssen Organisationen fähig sein, Synergien bei der Softwaregestaltung zu erkennen und Abstimmungsprozesse zu etablieren, um darüber zu verhandeln, was Teil einer Standardsoftware wird. So drückt sich in der jeweiligen Softwarearchitektur in den Fallstudien der Netzwerkcharakter der Arbeit aus: Mal lässt die Architektur eine dezentrale Gestaltung zu, mal nur eine zentrale.

9.2.2.2. Arbeit der Softwaregestaltenden

Zur soziotechnischen Netzwerkarbeit gehört auch, wie die Softwaregestaltenden arbeiten, die zwischen Anwendung und Programmierung tätig sind (6.4.3 und 8.4). Die Fallstudien zeigen, dass hinter der Softwaregestaltung eine eigenständige Gruppe von Beschäftigten steht und für die Transformation ihrer Arbeitskraft das Beschäftigungssystem, ihre individuelle Kontrolle und die Wissensverteilung entscheidend sind.

Die Führungskräfte **kontrollieren** die Softwaregestaltenden vor allem anhand ihrer Ergebnisse und nur bei Auffälligkeiten. Führungskräfte kümmern sich um den Rahmen: Sie legen je nach Fall das Budget fest, stellen Mitarbeitende ein, initiieren Projekte oder klären Konflikte innerhalb (z.B. mit anderen Führungskräften oder Teams) oder zu anderen Organisationen (Softwarefirma, IT-DL, EVU). Die Arbeitsbelastung der Softwaregestaltenden ist sehr unterschiedlich. Manche Befragte beschreiben sie als verhandelbar. Viele sprechen davon, dass sie intrinsisch motiviert sind. Wenn formale Hierarchien vorhanden sind, stehen sie über den Anwendenden. Dabei prägt die Grundkoordination die Kontrolle ihrer Arbeit: Von ihr hängt ab, ob mehr das Management (Hierarchie), die Kundschaft (Markt) oder die Kollegenschaft (Netzwerk) kontrollieren.

Softwaregestaltende können in puncto **Beschäftigungssystem** für ein IT-DL oder eine Softwarefirma arbeiten und für mehrere EVU tätig sein. Sie haben ein geringeres Interesse an einer disziplinarischen Karriere. Wenn EVU anfangen, Software zu gestalten, gibt es teilweise neue zusätzliche Karrierewege für Softwaregestaltende (»Kompetenzkarriere«). Die herkömmlichen Karrieremuster als Aufstieg auf einer Leiter in einer Hierarchie mit disziplinarischer Verantwortung gibt es weiterhin. Im Vergleich zu Anwendenden haben sie mehr (Arbeits-)Marktmacht und eigene Arbeitsmärkte. Die Organisationen setzen sie flexibel ein – ob innerhalb einer Matrixorganisation oder in anderen Organisationen. Das machen sie vor allem dann, wenn sie spezifisches Methodenwissen haben (z.B. Scrum) oder tiefere Kenntnisse einer auch in anderen Branchen genutzten Softwareumgebung (z.B. SAP). Es zeigt sich, was bereits im 6. Kapitel die Forschung feststellt: Sie erhalten Einfluss und ihre Position in den Organisationen, weil sie für diese die Software-Technologie nutzbar machen (vgl. Armstrong 1985, Hohlmann 2007).

Bei der **Wissensverteilung** zeigen die Fallstudien, dass die Softwaregestaltenden über sehr unterschiedliche interdisziplinäre Wissensstände verfügen. Das Spektrum reicht von Softwaregestaltenden, die sich vor allem mit der Koordination der Softwaregestaltung beschäftigen und nur noch wenig inhaltliches Wissen benötigen, bis hin zu IT-Beratenden, die nicht nur Anforderungen aufnehmen, sondern auch gleich noch umsetzen. Allgemein kann bei der Softwaregestaltung von einer interdisziplinären Praxisgemeinschaft gesprochen werden. Die Praxisgemeinschaft zeigt sich zum einen dadurch, dass jemand, der davon ausgeschlossen ist, nicht mehr Softwaregestaltung betreiben kann und von anderen abhängt, weil ihm das Wissen fehlt. Zum anderen zeigt sie sich dadurch, dass sich die beteiligten Personen in kontinuierlichem Austausch befinden, um Software zu gestalten. So entstehen unterschiedliche Praxisbiografien und damit unterschiedliche Lernbiografien für jeden einzelnen Beschäftigten. Hohlmann (2007) hat von einem Gestaltungsnetzwerk gesprochen, welches das interdisziplinäre Wissen hat, um die SAP-Standardsoftware über das Implementierungsprojekt hinaus zu gestalten. In Anlehnung an Wengers *Community of Practice* (1999) kann diese als

Community of Practice and Software Objects bezeichnet werden. Denn für das, was die Softwaregestaltenden wissen, sind weniger Schulungen entscheidend und vielmehr die Praxis mit anderen und der Software. Teil der Wissensverteilung der Softwaregestaltung sind die verwendeten, softwarebasierten Werkzeuge und die Software selbst. In ihnen materialisiert sich das Wissen (z.B. Quellcode, Ticketsystem, E-Mails, Dokumentationen von Umsetzungen und Funktionalitäten). Nicht nur Personen sind Träger des Wissens, sondern auch die Software mit dem entsprechend hinterlegten Wissen. Ob aus digitalen Dokumenten oder der gestalteten Software selbst: Das Lernen ergibt sich situativ in einer Praxis (Learning by Doing ist für die Befragten zentral), in der Software(-Objekte) eine wichtige Rolle spielt(en). In vielen Fallstudien sind Führungskräfte nicht Teil der Softwaregestaltung und haben entsprechend wenig Wissen über sie oder Softwareentwicklung im Allgemeinen.

9.2.3. Beitrag zur Debatte über die Kontrolle von Wissensarbeit

Das oben dargestellte Konzept der soziotechnischen Netzwerkarbeit stellt eine Ergänzung zur Debatte über die Kontrolle von Wissensarbeit in zweifacher Hinsicht dar: Erstens, weil ein netzwerkförmiger Arbeitsprozess die zentrale Ebene der Kontrolle von Arbeit ist. Zweitens, weil die Kontrolle allgemeine Netzwerkcharakteristika aufweist und es netzwerktypische Unterschiede in der Kontrolle gibt und z.B. keine Unterschiede wie jene zwischen permissiver und direkter Kontrolle (vgl. Friedman: 1977). Rennstam spricht von Wissens- statt Verhaltenskontrolle (vgl. Rennstam 2012: 1072). Dem ist zuzustimmen. Wobei es bei der Softwaregestaltung nicht nur um eine Kontrolle des Wissens, sondern der Kommunikation über letztlich individuelle und konstellationsabhängige Anforderungen geht. Der Standardisierung und Formalisierung von Softwaregestaltung und ihrer Koordination über Märkte und Hierarchien sind daher Grenzen gesetzt.

Der Netzwerkcharakter zeichnet sich erstens dadurch aus, dass nicht eine der Ebenen wie Subjekt, Betrieb oder Organisationsnetzwerk zentral für die Transformation der Arbeitskraft ist, sondern die Ebenen zusammenwirken, um zwischen Anwendung und Programmierung Software zu gestalten. Ein Teil der arbeitssoziologischen Forschung rückt den Beitrag der Arbeitenden selbst in den Vordergrund, wenn es um die Kontrolle von Arbeit geht.

> »Vor dem Hintergrund erweiterter Selbstorganisationspotentiale sorgen die Beschäftigten selbst für die effiziente Transformation ihrer eigenen Arbeitskraft in reale Arbeitsleistung« (Marrs 2010: 342).

Allgemein fußen neue Managementkonzepte mehr auf Autonomiespielräumen und weniger auf Fremdkontrolle, wie es beim Taylorismus der Fall ist, der davon ausgeht, dass Beschäftigte nicht gern und von sich aus effizient arbeiten, weshalb direkter Zwang und Kontrolle notwendig sind. Der Extremfall dieser subjektbasierten Transformation der Arbeitskraft ist jener des Arbeitskraftunternehmers von Voß und Pongratz (vgl. Marrs 2010: 339ff.). Die im 6. Kapitel zitierten Studien zu Produktmanagenden in Softwarefirmen nehmen schwerpunktmäßig eine einzelne Rolle in den Blick und wie diese Software gestaltet (vgl. Bolte 2017a, Bolte 2017b, Weishaupt/Hösl 2017). Andere Forschende sehen

für Fragen der Kontrolle von IT-Arbeit den einzelnen Betrieb (vgl. Boes et al. 2018) oder bei ausgelagerter IT-Arbeit die organisationsübergreifende Ebene als entscheidend an (vgl. Flecker/Holtgrewe 2008, Mezihorak 2018).

Im Gegensatz dazu zeichnet die soziotechnische Netzwerkarbeit aus, dass Ablauf, organisationale und interpersonale Beziehungen, Software und Softwaregestaltende alle dazu beitragen, zwischen Anwendung und Programmierung Kooperation und Kommunikation herzustellen, situativ Beschäftigte im Netzwerk einzubinden und allen Beteiligten Spielräume für Wissensaneignung, -verarbeitung und Kommunikation zu geben. Doch auch wenn das alle Arbeitsprozesse der Softwaregestaltung gemeinsam haben, ist der Schwerpunkt der Kontrolle des Arbeitsprozesses der Softwaregestaltung je soziotechnischer Konstellation anders und mal arbeiten nur Beschäftigte eines Betriebs und mal mehrere Organisationen zusammen. Damit sieht die vorliegende Untersuchung die Transformation der Arbeitskraft als etwas an, das auf mehreren Ebenen basiert, und folgt damit Sydow und Windeler (2000), Apitzsch (2006) oder Kalkowski/Mickler (2015), für die Netzwerke ebenfalls aus mehreren Ebenen und Dimensionen bestehen. Wobei diese Autor:innen die Ebene der Software vernachlässigen. Mit dem Konzept der soziotechnischen Netzwerkarbeit schließt die vorliegende Forschungsarbeit Forschungslücken sowohl bei der Verbindung von Arbeitsprozess, organisatorischer Ebene und unternehmensübergreifenden Beziehungen (vgl. Sydow/Helfen 2020: 225) als auch bei »Zusammenhängen zwischen praktischer Kooperation, Netzwerkformen, Nutzung von IuK-Techniken, Wissenstransfer und Arbeit« (Schmiede 2006: 466).

Neben der Debatte, welche Ebene zentral bei der Kontrolle von (IT-)Wissensarbeit ist, gibt es zweitens eine Diskussion darüber, welche Kontrollunterschiede von Arbeit existieren. Die im 5. Kapitel zitierte Forschung zur Softwareentwicklung sieht folgende Unterscheidungen: zwischen direkter und permissiver Kontrolle à la Friedman (1977) (vgl. Feuerstein 2012); zwischen Industrialisierung der Kopfarbeit inkl. digitalem Fließband und autonom agierenden Scrum-Teams (vgl. Boes et al. 2018).

Diesen Unterscheidungen folgt die vorliegende Untersuchung nicht. Sie stimmt vielmehr Autor:innen zu, die eine Gleichsetzung von Wissens- und Industriearbeit ablehnen. Klar grenzen z.B. Andrews et al. (2005) die Arbeit in der Softwareentwicklung von Fließbandarbeit ab, weil weder eingesetzte Methoden zur durchgehenden Standardisierung beitragen, noch die Abfolge der einzelnen Arbeitsschritte fest getaktet ist (vgl. ebd.: 63f.). Programmierende haben ihre Zeit selbst in der Hand und nur die Deadline ist direkt vom Management kontrolliert (vgl. Barrett 2005: 89). Auch in anderen stark digitalisierten Arbeitsbereichen ist die Rede von digitaler Fließbandarbeit irreführend. Zum Beispiel sind Microtasks bei Crowdwork nicht zeitlich in enge Prozessketten eingebunden und zeitliche Verzögerungen von wenigen Minuten führen anders als am Fließband nicht zur Störung des Gesamtprozesses. Im Vergleich zum klassischen Taylorismus ist im Fall des Crowdworking eine erweiterte Autonomie bei zumeist eng begrenzten Handlungsspielräumen vorhanden (vgl. Menz/Nies/Sauer 2019: 193f.).

Die Forschung zur Subjektivierung von Arbeit am Beispiel des Produktmanagements hat bereits herausgearbeitet (siehe 6.4.3.1), dass Softwaregestaltung ganz andere Anforderungen an die Kontrolle von Arbeit stellt. Softwaregestaltung lässt sich nicht im Vorhinein genau finanziell bewerten und standardisieren wie z.B. die Zubereitung eines Burgers, eine Massage, die Massenproduktion von Seife oder Hochleistungschips.

IT-Beratende oder IT-Projektleitende bei ERP-Implementierungen müssen komplexe Aufgaben bearbeiten und vorher unbestimmbare Informationsmengen sammeln, deuten und integrieren. Es wird Selbstmanagement erwartet (vgl. Bläsche/Lappe 2006: 307). Wissensarbeit ist an sich unbestimmt (vgl. Schmiede 2015: 51) und verlangt oftmals situativ zu improvisieren, weshalb sie sich von formal-rational durchorganisierten Prozessen unterscheidet (vgl. Stark 2017: 18f.). Der Typ der permissiven Kontrolle von Friedman ist notwendig, weil Engagement, Kooperation, Kreativität und Qualifikation der Beschäftigten »nicht durch direkte Kontrolle mobilisiert werden« (Marrs 2010: 336) können, wie es für Wissensarbeit typisch ist. Das Management muss die konkrete Arbeitsausführung den Beschäftigten überlassen (vgl. ebd.). Mehrere Autoren erachten Situativität und lokale Praktiken als typisch für Softwareentwicklung (vgl. Friedman/Cornford 1989: 358, Schulz-Schaeffer 1996, Wohlin/Smite/Moe 2015). Dabei ist IT keine reine Steuerungs- und Kontrolltechnik für die Wissensarbeitenden, sondern eine Infrastruktur, die Mitwirkung verlangt. Deshalb werden tayloristische und systemische Rationalisierungen für Organisationen, die IT nutzen, dysfunktional (vgl. Rock/Ulrich/Witt 1990: 43). Einige Autoren betonen die Dysfunktionalität einer Kontrolle von Softwareentwicklung, die zu sehr auf Hierarchien oder Märkte setzt (vgl. Gregory et al. 2013, vgl. Felin/Zenger/Tomsik 2009: 557, vgl. Brusoni/Prencipe/Pavitt 2001: 610). Eine andere Studie zur Softwareentwicklung hat festgestellt, dass es besser sein kann, die Werkzeuge den Kommunikationswegen anzupassen und nicht, wie am Fließband, die Arbeitenden einer Maschine unterzuordnen.

> »One direct implication is that software tools and programs should be designed to take advantage of the naturally occurring patterns of communication and work which spontaneously come about as a result of large scale programming« (Waterson/Clegg/Axtell 1997: 97).

Bei der soziotechnischen Netzwerkarbeit geht es nicht um die Unterscheidung von direkter und permissiver Kontrolle, industrialisierter und nicht-industrialisierter Wissensarbeit. Es geht um eine netzwerkspezifische Form permissiver Kontrolle und ihre typischen Netzwerkcharakteristika.

Was unterscheidet die soziotechnische Netzwerkarbeit von der permissiven Kontrolle? Das zeigt sich nicht nur an jeder der oben genannten Kategorien zum Arbeitsprozess der Softwaregestaltung (zwischen dezentral und zentral) und der Arbeit der Softwaregestaltenden (zwischen Matrix- und reiner Netzwerkorganisation). Die soziotechnische Netzwerkarbeit beruht zudem darauf, dass die Kategorien in ihrem Zusammenspiel eine flexible, situative, verteilte und horizontale Kooperation und Kommunikation ermöglichen. Rollen, Ablauf, kommunikative Beziehungen, digitale Werkzeuge und die Arbeit der Softwaregestaltenden können situativ und flexibel kombiniert auftreten. So kann die Kommunikation über vielfältige Kommunikationskanäle erfolgen – digital oder persönlich. Mal gibt der Ablauf vor, wer mitarbeitet, mal entscheiden das die Softwaregestaltenden eigenständig. Entscheidend ist die (temporäre) Praxisgemeinschaft sämtlicher Beteiligter der Softwaregestaltung unabhängig von ihrer Team-, Abteilungs- oder Organisationszugehörigkeit. In ihr stimmen sich die Beschäftigten über Erwartungen ab und es ergeben sich die Rollen, der genaue Ablauf, das Wissen, die Beziehungen und die ver-

wendeten Werkzeuge. Wer nicht Teil dieser Praxisgemeinschaft ist, hat nicht nur keine Chance mitzugestalten, sondern auch keinen Zugriff auf das Wissen und es fehlen die Möglichkeiten zu lernen. Das Wissen, das in ihr entsteht, zeigt netzwerktypische Eigenschaften: Es ist soziotechnisch im Netzwerk auf Beschäftigte und Software verteilt. Dabei ist eine gemeinsame Wissensbasis nicht garantiert. Das Wissen ergänzt sich erst in der Zusammenarbeit und es existiert ein stetiger Austausch zwischen Anwendung und Programmierung (8.4.3.2). Zu dieser netzwerkförmigen Praxisgemeinschaft gehört ein Mindestmaß an horizontaler Transparenz, ein »nicht-hierarchisches Modell reziproker Sichtbarkeit« (Schmidt 2012: 197), damit sich die Beteiligten in der Praxis untereinander eigenständig abstimmen können. Diese Transparenz zeichnet aus, dass sie soziotechnisch ist: Sie ist softwarebasiert aufgrund von softwarebasierten Werkzeugen wie Ticketsystem oder Softwareentwicklungsumgebung, wo die Beteiligten einsehen können, wer was gerade bearbeitet und welchen Status eine Aufgabe hat. Sie ist aber nicht ausschließlich softwarebasiert. Meist gehören noch Teamtreffen von Angesicht zu Angesicht dazu, die den Rahmen zur Ausübung von »peer group pressure« geben (vgl. Boes et al. 2018: 187).

In den Fallstudien zeigen sich nicht nur die gerade genannten allgemeinen netzwerktypischen Eigenschaften. Es lassen sich auch konkrete Lösungen von Kooperationsherausforderungen in organisationalen Netzwerken erkennen. INTERN1 konnte einen für seine Zwecke angepassten Scrum-Prozess über althergebrachte Abteilungsgrenzen hinweg mit vielfältigen Methoden wie Workshops, Resonanzgruppen und Prototyping etablieren. INTERN2 musste erst interne Konflikte zwischen Fachabteilungen lösen, um einen abteilungsübergreifenden Ablauf einzurichten. Bei KOOP1 kooperieren mehrere EVU und ein IT-DL. Sie haben einen ausgeklügelten Ablauf etabliert, um zu verhandeln, was das IT-DL als gemeinsamen Standard und was die EVU individuell gestalten. Bei KOOP2 ist die Kooperation teilweise prekär. An Stelle des gemeinsamen IT-DL übernehmen einzelne EVU die Softwaregestaltung wieder selbst – mit der Folge, dass diese dann bei eigenen IT-Gestaltungsprojekten intern eine abteilungsübergreifende Kooperation hinbekommen müssen. Bei KOOP3 funktioniert die Kooperation für die IoT-Softwaregestaltung zwischen IT-DL, Softwarefirma und EVU so gut, dass weder Verträge noch Hierarchien bei der Kontrolle der Arbeit eine große Rolle spielen. Die Softwarefirma aus PAKET hat intern energiewirtschaftliche Fachleute, braucht aber trotzdem Input zu Anforderungen und das Feedback ausgewählter EVU für die Gestaltung der komplizierten, industriespezifischen Standard-ERP-Software. STARTUP begreift sich anders als althergebrachte EVU als Softwarefirma, ist von Beginn an auf die Softwaregestaltung ausgerichtet (Primat der Softwareentwicklung) und arbeitet ohne formale Führungskräfte in einer Mischung aus Scrum und Holokratie.

Ist ein Arbeitsprozess der Softwaregestaltung etabliert und wollen das Management oder die Softwaregestaltenden selbst den Prozess weiter optimieren, müssen sie dessen Eigenheiten berücksichtigen. Denn statt wie in einer (Standard-)Massenproduktion ist jede Anforderung individuell in ihrer Ausarbeitung und Umsetzung. Methoden wie Scrum betonen, dass es sich nicht um einen linearen, vorweg detailliert planbaren Prozess bis zur fertigen Software handelt, sondern um einen iterativen Prozess, im Laufe dessen sich z.B. das Wissen über einen Anwendungsbereich einer Software ebenso wie jenes über die Anwendenden ändert. Es muss möglich sein, immer wieder neues Wissen

in Anforderungen aufnehmen zu können und zu lernen. In den Fallstudien zeigt sich keine Prozesstaktung, sondern vielmehr eine situative Synchronisierung von Arbeitsschritten – ob durch Zyklen wie bei Scrum oder Deadlines wie bei Projekten. Wobei diese Synchronisierung des Handelns immer Interventionen, Feedback (z.B. durch Tests) und Priorisierung durch (bestimmte) Beteiligte zulässt. Zwar ist es Teil der Softwaregestaltung, dass die »Kooperations- und Kommunikationspraxis der Arbeitsteams […] aufgedeckt, analysiert und optimiert« (Boes et al. 2018: 26) wird, z.B. durch Scrum Master:in oder die Projektleitung. Ein digitales Fließband ist aber in keiner der Fallstudien zu beobachten. Es ist in den Fallstudien nicht das primäre Ziel, »selbst hochqualifizierte Wissensarbeit systematisch und rational zu organisieren, um sie – mittels Transparenz und Kontrolle sowie einer Kollektivierung von Wissen – plan- und wiederholbar zu machen« (Boes et al. 2018: 180). Es geht darum, Kommunikation und Kooperation zu verbessern. Das schließt keine Standardisierung und Formalisierung aus. Allerdings nehmen sie dann eine andere Form an, z.B. als Standardisierung der verwendeten Softwarewerkzeuge oder Formalisierung des Scrum-Ablaufs und der Rollen. Dabei sind der Standardisierung und Formalisierung klare Grenzen gesetzt, weil die Softwaregestaltung stark von der Subjektivität der Beschäftigten abhängig bleibt und situatives Handeln zwischen Anwendung und Programmierung möglich bleiben muss. Andere Optimierungen sind zielführender. Wobei nur eine befragte Person erwähnt hat, dass ihr EVU Workshops allein mit dem Ziel durchführt, die Kommunikation zwischen den Beschäftigten zu verbessern.

Statt den Begriff der Industrialisierung auf Softwaregestaltung anzuwenden, sind andere Begriffe angebracht, um die netzwerkspezifischen Unterschiede deutlich zu machen und nicht zu suggerieren, Softwaregestaltung sei industrialisierbar. Es steht außer Frage, dass Organisationen die Möglichkeiten von Software nutzen, um zu automatisieren und Beschäftigte zu steuern. Aber für die Softwaregestaltung trifft es deshalb nicht zu, dass »insbesondere in hochqualifizierten Feldern [die] sehr ausgeprägten – typischen Freiheitsgrade der Angestellten« (Boes et al. 2018: 37) verschwinden. Die Freiheitsgrade sind gerade die Ressource, welche die Organisationen nutzen wollen und müssen. Alle Kategorien der soziotechnischen Netzwerkarbeit zeichnen sich dadurch aus, dass es nicht um die direkte Kontrolle des Verhaltens geht.

Thompson/Laaser verwenden den Begriff der qualitativen Intensivierung, um zu beschreiben, wenn Organisationen zusätzliche Fertigkeiten der Beschäftigten wie deren implizites Wissen und Gefühle nutzen (vgl. Thompson/Laaser 2021: 145). Für die vorliegende Arbeit geht es bei qualitativer Intensivierung u.a. neben Beziehungsfähigkeit, Beziehungspflege, intrinsische Motivation, sich aktiv auf Software einzulassen und Lernbereitschaft auch um ein Mehr an interdisziplinärem Wissen, thematisch unterschiedliche Projekte gleichzeitig zu machen, gesamte Prozesse zu verstehen bzw. in digitalen Prozessen denken zu müssen und mehrere Kunden und deren IT-Landschaft zu kennen (8.4.3.2). Andere Autorinnen sprechen von einer Psychologisierung der Arbeitswelt, bei der emotionale und kommunikative Kompetenzen wichtiger werden (vgl. Illouz 2008), und Firmen erwarten, dass Mitarbeitende ihre Gefühle instrumentell für die Dienstleistungsarbeit einsetzen (vgl. Hochschild 1990). Dabei geht es nicht nur darum, die subjektiven Potenziale überhaupt zu nutzen. Es geht auch um eine bestimmte Art und Weise, unter welchen Bedingungen und wie eine Form der Optimierung von Arbeit ausse-

hen und konzeptualisiert werden kann, bei der diese Arbeit in einem Netzwerk aus Subjekt, Software, Ablauf und Organisation(en/-seinheiten) stattfindet. So fällt z.B. auf, dass die Arbeitsbelastung der Befragten sehr unterschiedlich ist. Ist das womöglich typisch für die Arbeit in Netzwerken: Wer sich gut verkaufen, im Netzwerk bewegen und selbst Grenzen setzen kann, ist weniger belastet?

Es geht bei der Softwaregestaltung nicht darum, dass sich die oder der Einzelne einem objektiven, weitgehend standardisierten und formalisierten Prozess unterwirft. Statt entweder Netzwerkarbeit oder keine Netzwerkarbeit geht es bei der Softwaregestaltung darum: Ab welchem Grad an Standardisierung und Formalisierung liegt keine soziotechnische Netzwerkarbeit mehr vor und die Kontrolle der Arbeit über Hierarchien und Märkte wird aus Sicht der Softwaregestaltung dysfunktional und die Transformation der Arbeitskraft misslingt?

9.3. Zweiter Debattenbeitrag: Softwaregestaltung als Arbeit an der digitalen Transformation

9.3.1. Teil der digitalen Transformation: soziotechnische Arbeitsgestaltung der Softwareanwendung durch die Softwaregestaltung

Die Softwaregestaltung betrifft nicht nur die Softwaregestaltenden selbst (ob sie in einer Matrix- oder einer Netzwerkorganisation arbeiten). Sie hat je nach soziotechnischer Konstellation sowohl Folgen für die Software als auch für die anwendende Organisation. Anders als z.B. bei der Einführung eines ERP-Systems geht es nicht um die Folgen einer fertigen Standardsoftware und deren Anpassung (6.5.4). Es geht um die Folgen des Arbeitsprozesses der Softwaregestaltung. Wie Organisationen seine Möglichkeiten ausschöpfen, hängt von seinem Verhältnis zur Softwareanwendung ab. Besteht das Verhältnis in der reinen Zulieferung einer Software, dann sind beide Arbeitsprozesse getrennt voneinander (z.B. in einer Softwarefirma und einem EVU). Beim Primat der Softwareentwicklung bildet der Arbeitsprozess der Softwaregestaltung als Teil der Softwareentwicklung den Kern der anwendenden Organisation. Der Arbeitsprozess der Softwareanwendung organisiert sich um die stetig weiterentwickelte Software herum. Die vorliegende Studie befasst sich mit den Folgen für die Softwareanwendung, die sich aus der Softwaregestaltung ergeben, indem sie das Verhältnis von Softwaregestaltung zur Softwareanwendung betrachtet. Sie bezeichnet dieses Verhältnis mit dem Begriff der soziotechnischen Arbeitsgestaltung der Softwareanwendung.

Die Untersuchung nutzt zwei Kategorien, um in den Fallstudien die soziotechnische Arbeitsgestaltung der Softwareanwendung zu analysieren: den **Einfluss** beider Arbeitsprozesse aufeinander und die **Konflikte** zwischen ihnen. Beide Kategorien führt der nächste Punkt 9.3.2 näher aus.

Dabei zeigen die Fallstudien, dass Betriebsräte innerhalb der EVU die soziotechnische Arbeitsgestaltung mitbestimmen, indem sie intervenieren, aber nicht inhaltlich gestalten. Sie spielen bei Fragen der Partizipation insofern eine Rolle, als sie Rahmenbedingungen mitgestalten und weniger die Software selbst. So prägen sie die Partizipation z.B., wenn ein Betriebsrat einfordert, dass die Softwaregestaltung Anwendende via

Workshops einbezieht. Das Management fällt strategische Entscheidungen, gibt Ziele vor, stellt Ressourcen zur Verfügung und bringt teilweise eigene Anforderungen ein.

Bevor das Schlusskapitel den Beitrag der Softwaregestaltung zur Debatte über die digitale Transformation diskutiert, sollen vier Idealtypen verdeutlichen, welchen Unterschied die Softwaregestaltung machen kann.

9.3.2. Zwischen unabhängiger und abhängiger soziotechnischer Arbeitsgestaltung

Die vier im Folgenden vorgestellten Idealtypen verdeutlichen die Unterschiede der soziotechnischen Arbeitsgestaltung, die davon abhängen, wie Organisationen eines der Kernprobleme der Softwaregestaltung lösen: jenes der softwaretechnischen Gestaltungsmöglichkeiten. Richten sie sich organisatorisch nur auf die Softwareanwendung aus oder (auch) auf die Softwaregestaltung? Gestalten sie eine Individual- oder eine Standardsoftware? Bei den Typen INTERN und PROPRIETÄRER STANDARD sind die anwendenden Organisationen unabhängig in ihrer Arbeitsgestaltung, weil sie die Softwaregestaltung selbst in der Hand haben. Bei den Typen STANDARDPAKET und WERKVERTRAG sind sie hingegen abhängig von anderen Organisationen wie Softwarefirmen oder IT-DL (8.6.1.3.). Hier sind nur zwei Typen ausführlich mit sämtlichen Kategorien und Unterkategorien dargestellt, um die Unterschiede zwischen abhängiger und unabhängiger soziotechnischer Arbeitsgestaltung der Softwareanwendung zu verdeutlichen.

Beim Typ **STANDARDPAKET** entwickelt eine Organisation (wie z.B. in der Fallstudie PAKET) eine industriespezifische Standardlösung, die sie verkauft. Die anwendenden EVU sind von dieser Organisation (z.B. einer Softwarefirma) **abhängig**, was die soziotechnische Arbeitsgestaltung anbelangt. Warum? Das liegt am Einfluss, der sich anhand von vier Kategorien beschreiben lässt: Aufgrund des **Kontrollverhältnisses** der Arbeitsprozesse von Softwaregestaltung und -anwendung zueinander können die EVU nur die Anwendung der industriespezifischen Standardlösung kontrollieren, nicht aber die Softwaregestaltung des Standards. Sie können den Standard nur in den vorgegebenen Möglichkeiten anpassen (z.B. durch Einstellungen). Zudem **partizipieren** vorwiegend Fachexpert:innen und gestalten z.B. über Arbeitskreise oder einzelne Projekte mit. So können die EVU nur den Arbeitsprozess der Softwareanwendung **reorganisieren**. Zuletzt hat die Softwarefirma des Standardpakets das letzte Wort bei Entscheidungen über die **Ziele** der Softwaregestaltung. Das betrifft zum Beispiel Entscheidungen darüber, was in den Standard kommt und ob sie individuellen Wünschen einzelner EVU entgegenkommt. Die Abhängigkeit der EVU bei der soziotechnischen Arbeitsgestaltung zeigt sich auch an den **Konflikten**. Es sind externe Konflikte mit der Softwarefirma und die EVU können diese daher nicht unabhängig von ihr lösen, z.B. wenn es um die inhaltliche Gestaltung der Standardsoftware geht.

Beim Typ **INTERN** entwickeln EVU eine industriespezifische Software, was ihnen eine **unabhängige** Arbeitsgestaltung der Anwendung ermöglicht (wie z.B. die Fallstudien INTERN1 und INTERN2). Die EVU haben **Einfluss** auf beide Arbeitsprozesse von Softwaregestaltung und Softwareanwendung. Sie können die **Ziele** der Softwaregestaltung selbst bestimmen und auf die eigenen Bedarfe ausrichten. Zu einer solchen Ausrichtung kann gehören, dass sämtliche Anwendenden unterschiedlicher Fachbereiche

partizipieren. Das **Kontrollverhältnis** ist so, dass die Softwaregestaltung die Softwareanwendung für ihre Zwecke kontrollieren kann. Die Softwaregestaltung entscheidet, ob sie Führungskräfte der Anwendung mitgestalten lässt und welche Anforderungen die Programmierenden umsetzen. Die beiden Arbeitsprozesse von Softwaregestaltung und -anwendung kann das EVU im Sinne der Softwaregestaltung **reorganisieren** und z.B. Team- und Abteilungssilos auflösen und ein integriertes Prozessteam für die Softwaregestaltung schaffen. Die **Konflikte** sind rein intern in der Organisation, z.B. zwischen Fachbereichen oder mit dem Betriebsrat. Deshalb kann das EVU diese eigenständig lösen.

Beim Typ **PROPRIETÄRER STANDARD** schöpft eine Organisation die Möglichkeiten der Softwaregestaltung maximal aus. Sie gestaltet für die eigenen Zwecke eine individuelle Software, wobei die Softwareanwendung abhängig von der Softwaregestaltung und der entwickelten Software organisiert ist. Zusätzlich verdienen Organisationen dieses Typs mit dem Verkauf der Software auch noch Geld, indem sie andere Organisationen als Standardsoftware anwenden. Dieser Typ trifft auf einen Teil der in der Fallstudie STARTUP gestalteten Software zu.

Beim Typ **WERKVERTRAG** wird eine individuelle, industriespezifische Software durch eine Softwarefirma oder einen IT-DL im Auftrag eines EVU gestaltet. Dem Typ entsprechen KOOP1 und KOOP2 zum Teil, und zwar dann, wenn das IT-DL für ein EVU eine individuelle Software gestaltet. Wobei dies dann im Rahmen der bestehenden Kooperation geschieht und die Interviews zu diesem Fall darauf schließen lassen, dass das EVU dann zumindest mitgestaltet und nicht nur eine Beauftragung macht, wie es für einen reinen Werkvertrag typisch wäre.

9.3.3. Beitrag zur Debatte über die digitale Transformation

Indem Softwaregestaltung die Arbeit der Softwareanwendung prägt, ist sie ein Teil des soziotechnischen Wandels der Arbeitswelt. Dieser wird derzeit unter dem Begriff der digitalen Transformation diskutiert, hinter dem sich laut Pfeiffer und Schrape (2023) aus arbeits- und industriesoziologischer Perspektive allerdings keine allgemeine Theorie verbirgt. Sie nennen drei Ansätze, die versuchen, die digitale Transformation theoretisch zu fassen. Zweien davon ist die vorliegende Untersuchung in ihrer grundlegenden Ausrichtung gefolgt, indem sie wie Apitzsch et al. (2021) die Arbeit an der Digitalisierung in konkreten Arbeitswelten erforscht und wie Hirsch-Kreinsen (2020) von einer inkrementellen Digitalisierung ausgeht (6.5.3). Warum das Schlusskapitel den dritten von Pfeiffer und Schrape genannten Ansatz hier nicht weiter diskutiert, erläutert 9.3.3.2 weiter unten. Unabhängig vom Ansatz benennen Pfeiffer und Schrape als Forschungslücke, wie die konkrete Gestaltung der digitalen Transformation abläuft (2023: 137). Die vorhergehenden Kapitel haben sich damit befasst. Die folgenden Seiten erläutern, was die vorliegende Studie von den beiden oben genannten Ansätzen unterscheidet und welche Gemeinsamkeiten bei der Analyse der Gestaltung und des Verlaufs der digitalen Transformation bestehen.

Dabei sei vorweg darauf hingewiesen, dass anders als bei Hirsch-Kreinsen (2020) und Apitzsch et al. (2021) die vorliegende Untersuchung über soziotechnischen Wandel bzw. die digitale Transformation durch Softwaregestaltung gleichzeitig eine Unter-

suchung über Softwareentwicklung ist. Bei industriespezifischer Software ist die Technikgestaltung nicht mehr von der Gestaltung von Arbeit und Organisation zu trennen, weil die Folgen der Technikgestaltung für die anwendende Organisation so gravierend sind und davon abhängen, welche Möglichkeiten Organisationen bei der Softwaregestaltung nutzen. Einerseits lässt sich ein Teil des Wandels der Energiewirtschaft als soziotechnischer Wandel durch Softwareentwicklung beschreiben (z.B. durch ihre Akteur:innen, Methoden und Wissensbestände). Indem Softwareentwicklung ein Teil von (bisher) Nicht-IT-Branchen und -Firmen wird, prägt sie deren Organisation und Arbeitsweise. Andererseits zeigen die Fallstudien wie auch das Konzept der soziotechnischen Netzwerkarbeit, dass industriespezifische Softwaregestaltung und damit industriespezifische Softwareentwicklung ein soziologisches Problem sind. Die Analyse der Softwaregestaltung in der Energiewirtschaft und der Zusammenhänge zwischen soziotechnischer Konstellation, Arbeitsprozess der Softwaregestaltung, Arbeit der Softwaregestaltenden und soziotechnischer Arbeitsgestaltung der Softwareanwendung ist zugleich ein Beitrag zum besseren Verständnis von industriespezifischer Softwareentwicklung.

9.3.3.1. Gestaltung – zwischen dezentral und zentral, zwischen Standard- und Individualsoftware

Bei der Gestaltung der digitalen Transformation rückt Hirsch-Kreinsen die betriebliche Ebene in den Vordergrund und dass die Betriebe ganz unterschiedliche Wege einschlagen können, was er anhand von drei Typen zeigt (weitreichende Digitalisierer, selektive Digitalisierer, skeptische Unternehmen) (vgl. Hirsch-Kreinsen 2020: 43). Wie auch Apitzsch et al. vertritt er keinen Technikdeterminismus und sieht Betriebe als ein soziotechnisches System aus Technik, Mensch und Organisation an (vgl. ebd.: 88). Für Apitzsch et al. (2021) hängt die Gestaltung von mehreren Ebenen ab (Betrieb, Arbeitsprozess, Subjekt, organisationales Feld, Arbeitsbeziehungen, Recht). Womit sie anders als Hirsch-Kreinsen die überbetriebliche Ebene in den Fokus rücken. Sie konstatieren, dass Betriebe Gestaltungsspielräume von IT nicht nutzen, weil »prozessferne Akteure (zentrale Planungsbereiche oder externe Technikanbieter) eine dominante Rolle spielen« (ebd.: 27). Die Partizipation betrifft »oftmals […] die reine Implementation einer bereits vorgegebenen Technik und hängt jedenfalls stark von den vorhandenen betrieblichen Partizipationskulturen ab« (ebd.).

Die vorliegende Untersuchung betont, dass auch die Entwicklung von Software kein technikdeterministischer, sondern ein soziotechnischer Gestaltungsprozess ist. Zudem ist in den Fallstudien der Betrieb eine wichtige Ebene der Gestaltung und es bestehen große Unterschiede zwischen den EVU, wie sie die Möglichkeiten der Softwaregestaltung nutzen. Wie die Fallstudien zeigen, kann ein dezentraler Arbeitsprozess der Softwaregestaltung in einem EVU auch nur aus wenigen Beschäftigten bestehen, die für einen Anwendungsbereich eine Software gestalten. Damit ergeben sich vielfältige Gestaltungsmöglichkeiten innerhalb eines Betriebs. Aber wie bei Apitzsch et al. (2021) spielt bei der Mehrzahl der Fallstudien bei der Softwaregestaltung die überbetriebliche Ebene eine Rolle. So gestalten in der Fallstudie KOOP1 mehrere EVU kooperativ einen Standard, und auch individuelle Programmierungen für einzelne EVU erledigt zum Teil das IT-DL. Solch ein zentralisierter Arbeitsprozess der Softwaregestaltung verlangt allerdings Kompromisse, denen sich die einzelnen Betriebe fügen müssen.

Apitzsch et al. (2021) ist insofern recht zu geben, als dass die EVU oftmals eine vorgegebene Technik in Form einer Standardsoftware einsetzen und häufig externe technikanbietende Unternehmen wie Softwarefirmen über die Gestaltung entscheiden. Allerdings erlaubt es der Fokus auf den Arbeitsprozess der Softwaregestaltung, die vielfältigen Formen der Softwaregestaltung und damit der Gestaltungsmöglichkeiten in den Blick zu nehmen. So zeigen die Fallstudien die typischen Muster einer Gestaltung durch soziotechnische Netzwerkarbeit: Gleichzeitigkeit und Wechsel zwischen Zentralisierung und Dezentralisierung, zwischen Individual- und Standardsoftware, zwischen einer rein anwendenden Organisation und einer (mit)gestaltenden Organisation. Zudem ist die Partizipation an der Softwaregestaltung in einem Netzwerk verteilt, bei dem Anwendende, Betriebsrat und Management unterschiedlich stark beteiligt sind.

Es wirken nur ausgewählte EVU an der Gestaltung der industriespezifischen ERP-Software der Fallstudie PAKET mit. Die Vielzahl der anwendenden EVU hat sehr geringen Einfluss auf die Softwaregestaltung. Es zeigen sich Unterschiede, wie EVU die Einstellungsmöglichkeiten am Standard nutzen und sich die anwendenden Organisationen an der Standardlösung ausrichten. In den Fallstudien, die SAP einsetzen, zeigt sich beides: Anwendung eines Standards und dessen individuelle Gestaltung und damit sowohl eine Abhängigkeit und Unabhängigkeit bei der Arbeitsgestaltung der Softwareanwendung. In den Fällen INTERN1 und INTERN2 nehmen die EVU weitreichende Erweiterungen oder Anpassungen des SAP-Standards selbst vor. Auch in den EVU der Fallstudien KOOP1 und KOOP2 können einzelne Teams eigenständig Software gestalten. Vor allem KOOP1 zeigt die Dynamik innerhalb von EVU, dezentral Rollen und Abläufe für die Softwaregestaltung auf- und abzubauen, Key User:innen als permanent Softwaregestaltende und/oder flexible Softwaregestaltende wie IT-Beratende einzusetzen. Auch wenn die vorliegende Untersuchung nicht sämtliche Arbeitsprozesse der Softwaregestaltung der EVU ausleuchten konnte, entsteht doch vor allem bei jenen Fallstudien, für die viele Interviews vorliegen (KOOP1, KOOP2), der Eindruck, dass die EVU an allen Ecken und Enden Software im größeren und kleineren Umfang gestalten. Das liegt auch daran, wie der Fall KOOP3 zeigt, dass kleinere Entwicklungen nicht aufwendig sind. In der Fallstudie erlaubt es die modulare IoT-Softwarelösung, dass sowohl EVU wie IT-DL selbst gestaltete und programmierte Module anschließen. Ein EVU von KOOP2 etabliert eine eigene, kleine Entwicklungsplattform, um kleinere Anwendungen dezentral in unterschiedlichen Fachbereichen entwickeln zu können.

Neben diesen typischen Mustern zwischen Zentralisierung und Dezentralisierung und Individual- und Standardsoftware zeigen sich große Unterschiede, ob EVU rein anwendende Organisationen sind oder auch mitgestalten. In den Fallstudien sind es die größeren EVU, die eher Ressourcen dafür haben, um selbst Software und damit die eigene Arbeit zu gestalten. EVU und IT-DL aus vier Fallstudien (INTERN1, INTERN2, KOOP1, KOOP2) nutzen die Entwicklungsplattform von SAP, die ermöglicht, dass Unternehmen die ERP-Standardsoftware individuell anpassen und erweitern. Kleine EVU fokussieren sich auf die reine Anwendung. Sie sind wie bei PAKET abhängig von Softwarefirmen, die keine Entwicklungsplattform zur Verfügung stellen, wodurch die EVU nur Einstellungen vornehmen können – wofür sie teilweise gar nicht die Kapazitäten haben. Dabei kann die Gestaltung einer industriespezifischen Software und die Abhängigkeit von der Softwarelieferfirma so weit gehen, dass eine industriespezifische

Anwendungsplattform auf Basis einer Standardsoftware entsteht, die die EVU nutzen. Zwei Befragte sprechen davon, dass das IT-DL bzw. das Softwareunternehmen Stadtwerke »spielen« (so ein Befragter) können und dies auch tun. Sie trennt von einem EVU nur noch, dass sie Netze besitzen oder Verträge mit Endkunden haben. Die Software, um die notwendigen industriespezifischen Prozesse abzuwickeln, haben sie. Ein Beispiel dafür ist die Standard-ERP-Software, die das IT-DL von KOOP1 für mehrere EVU betreibt und gestaltet. Das IT-DL übernimmt alle Anpassungen durch Einstellungen an der Standardsoftware oder die Programmierung von Erweiterungen. Ein anderes Beispiel ist die Zusammenarbeit zwischen einer Softwarefirma und einem EVU in der PAKET-Fallstudie. Das EVU betreibt den von der Softwarefirma entwickelten industriespezifischen Standard mit entsprechenden Standardeinstellungen auf einem Server. In beiden Fällen bieten IT-DL bzw. Softwarefirma an, einzelne Prozesse via BPO für EVU zu übernehmen, d.h. nicht nur die (fertig eingestellte) industriespezifische Software direkt nutzbar zur Verfügung zu stellen, sondern auch ihren Betrieb zu verantworten und die Anwendenden zur Verfügung zu stellen, die die Software bedienen.

Der Fokus auf den Arbeitsprozess der Softwaregestaltung lässt es zu, in jeder Fallstudie zu untersuchen, inwiefern Anwendende in ihn eingebunden sind. Eine direkte Partizipation der Anwendenden findet in den Fallstudien allerdings nur punktuell und meist vermittelt z.B. über Key User:innen oder Anforderungsmanagende statt. Oftmals schreiben fachliche Expert:innen Anforderungen, welche die Software selbst gar nicht anwenden. So bewegt sich die Softwaregestaltung in den Fallstudien zwischen einer repräsentativen Technokratie und einer kapitalistischen Technokratie (Näheres zu diesen Partizipationstypen unter 8.5.3.3). Vielmehr gehören die Softwaregestaltenden zur Gruppe der Höherqualifizierten, die die Arbeit der Anwendenden gestalten und in den betrieblichen Hierarchien höhergestellt sind. Die Mehrzahl der Anwendenden sind das Objekt der Softwaregestaltung und müssen mit ihren Ergebnissen leben. So zeigen die Fallstudien, dass meistens die Grundstruktur der Technikgestaltung bleibt: Einige wenige gestalten die Arbeit vieler. Das zeigt sich vor allem dann, wenn die Softwaregestaltung außerhalb der anwendenden Organisation stattfindet. Ein Extremfall ist eine Cloud-Anwendung, auf deren Gestaltung der befragte Anwender und sein EVU gar keinen Einfluss haben (EVU von KOOP2). In einem anderen Fall ist der Anwender fast ein Jahr aufgrund von Burnout ausgefallen (EVU von PAKET), weil über einen längeren Zeitraum eine Kombination aus Termindruck durch die Regulierung und stetigen Updates der extern entwickelten Standardsoftware bestand. In beiden Fällen fungieren Anwendende als Puffer. Sie sind zwar abhängig von einer funktionierenden Software, haben aber ihre Gestaltung nicht in der Hand und müssen fehlerhafte oder nicht fertige Software durch ihre Arbeit kompensieren. Eine im 6. Kapitel (6.5.4) zitierte Studie kommt zu dem Schluss, dass nach einer ERP-Implementierung die Anwendenden mit einer nicht passenden Software zurechtkommen mussten:

> »In short, users were left to pick up the pieces after a so called ›successful‹ implementation.« (Lyytinen/Newman 2015: 90)

Ob dies bei Standardsoftware oder Cloud-Lösungen häufiger vorkommt und unter welchen Bedingungen Anwendende als eine Art Puffer fungieren müssen, wäre noch weiter

zu untersuchen. Aber selbst bei den Fallstudien, bei denen die anwendende Organisation eine eigene individuelle Software gestaltet, ist eine direkte Partizipation der Anwendenden und eine Umsetzung ihrer Anforderungen nicht garantiert. STARTUP gestaltet zwar unabhängig die eigene Software, bezieht die Anwendenden aber nur ein, um eine fertige Umsetzung zu testen oder Feedback zu ihr zu geben.

Ob die interne Gestaltung von Software mit den Partizipationskulturen zusammenhängt, wie Apitzsch et al. (2021) schreiben, kann die vorliegende Untersuchung nur dahingehend stützen, dass der Betriebsrat zwar keinen direkten Input zur Softwaregestaltung liefert, aber bei INTERN1 explizit die Beteiligung der Anwendenden angefordert hat. Sonst interveniert er vor allem, um eine individuelle Leistungskontrolle zu verhindern (8.5.3.2). Software zu gestalten, um die »Potenziale einer qualifikationsorientierten Gestaltung der Arbeit bestmöglich auszuschöpfen« (Hirsch-Kreinsen 2020: 88), kam in keiner der Fallstudien vor. Es wäre zu untersuchen, wie so ein Prozess der Softwaregestaltung aussehen und unter welchen Bedingungen ihn ein EVU etablieren würde.

Grundsätzlich ist Hirsch-Kreinsen (2020) zuzustimmen, wenn er von »widersprüchlichen Perspektiven für das untere und mittlere Management« (77) schreibt und davon, dass es neue Aufgaben für das Management gibt, weil IT- und Produktionskompetenz in manchen Managementpositionen verschmelzen (vgl. ebd.: 78). Der Blick auf die soziotechnische Arbeitsgestaltung durch die Softwaregestaltung zeigt, dass nicht das Management die Technik- und Arbeitsgestaltung und damit die Rationalisierung in den Organisationen allein betreibt (8.6.4). Die Führungskräfte – falls vorhanden – sind Rahmengeber und nur teilweise direkt involviert. Sie haben die Kontrolle über finanzielle und personale Ressourcen, fällen strategische Entscheidungen, setzen Ziele und kontrollieren Ergebnisse. Sie entscheiden über betriebsinterne Karrieren – egal ob von Personen der Softwareanwendung, -gestaltung und -programmierung. Sonst unterscheidet sich von Fall zu Fall, welche Rolle das Management und bestimmte Führungsebenen spielen: ob sie ein Hindernis für die Softwaregestaltung sind oder diese selbst vorantreiben und entsprechendes Wissen dazu haben. Wenn die Softwaregestaltung mehrere Abteilungen und Teams betrifft, hängen die Softwaregestaltenden in einer Matrixorganisation davon ab, dass die Führungskräfte die Softwaregestaltung unterstützen. Wenn die Softwaregestaltenden hierarchisch nicht höhergestellt sind, hängt es von der Kooperationsbereitschaft anderer Führungskräfte ab, wie viel Einfluss der Arbeitsprozess der Softwaregestaltung auf die Softwareanwendung innerhalb eines EVU hat. Die Anforderungsrunde von INTERN2 hat z.B. erst funktioniert, als die Fachbereiche kooperationswillig waren. Basiert eine Firma auf einer Netzwerkorganisation mit flachen Hierarchien, sind Teile des Managements überflüssig. Lagert ein EVU die Softwaregestaltung komplett aus, muss sich das Management zwar nicht mehr um die Kontrolle dieser Arbeit kümmern. Es ist dafür aber von externen Softwarefirmen abhängig. Es ist offen, inwiefern Softwaregestaltende Teil des Managements werden. Wenn nicht nur die informationsvermittelnde Funktion des mittleren Managements durch Softwarelösungen wie ERP-Systeme entfällt (Hohlmann 2007: 365f.), sondern auch die der disziplinarischen Führung bei Netzwerkarbeit und Teile der Arbeitsgestaltung zur (externen) Softwaregestaltung wandern: Welche Hierarchien und welches Management brauchen anwendende Organisationen dann überhaupt? Zudem wäre noch zu vertiefen, aus welchen Gründen sich das Management für welche Strategie entscheidet und inwiefern und welche EVU bei der

Softwaregestaltung abhängig sind. Denn der Markt für industriespezifische Software ist groß (7.2.1.1). Die EVU können auch einzelne, industriespezifische Module von unterschiedlichen Softwareherstellern beziehen und so ist deren Marktmacht beschränkt. Andererseits wird es gerade jenen EVU schwerfallen, das softwarezuliefernde Unternehmen zu wechseln, die viel in die individuelle Anpassung und Erweiterung von z.B. SAP investiert haben.

9.3.3.2. Verlauf – die Möglichkeiten der Softwaregestaltung ausreizen

Was den Verlauf der digitalen Transformation anbelangt, grenzen sich Hirsch-Kreinsen (2020) und Apitzsch et al. (2021) von Ansätzen ab, die allgemein einen disruptiven Wandel beobachten. Dabei befasst sich Hirsch-Kreinsen mit industrieller Arbeit, bei der er eine inkrementelle Optimierung gegebener Prozesse und eine Pfadabhängigkeit ausmacht. Die Betriebe setzen bei eingespielten Arbeitspraktiken und vorhandenen Qualifikationen an, um in kleinen Schritten die Produktion durch den Einsatz von digitaler Technik zu verbessern (vgl. Hirsch-Kreinsen 2020: 40ff.). Auch Apitzsch et al. machen eine Pfadabhängigkeit aus, wobei sie branchenunabhängig eine strukturierte Vielfalt, Ungleichheiten, Widersprüche und eine plurale Digitalisierung wahrnehmen (vgl. Apitzsch et al. 2021: 20f.).

Auch in den Fallstudien der vorliegenden Untersuchung zeigen sich ein inkrementeller Wandel und Pfadabhängigkeiten. Anders als Hirsch-Kreinsen (2020) und Apitzsch et al. (2021) fokussiert sie aber den konkreten Arbeitsprozess der Softwaregestaltung, der den soziotechnischen Wandel vorantreibt. So wird eine besondere Form einer technikentwicklungsbezogenen Rationalisierung sichtbar und dass der Verlauf der digitalen Transformation auch davon abhängt, inwieweit Organisationen die Möglichkeiten der Softwaregestaltung ausreizen. Damit erklärt das Zusammenspiel von Arbeit, Organisation, Software und Softwaregestaltung, wie die digitale Transformation verläuft. Gestalten EVU nicht nur eine Software (mit), sondern ändern auch ihre Organisation, um die Softwaregestaltung zu optimieren? Wo liegen die Grenzen des Einsatzes von Software in einem Anwendungsbereich und inwieweit kann der Verlauf der Softwaregestaltung disruptive Veränderungen erklären? Setzen sich EVUs mit Individualsoftware in der Branche durch oder Softwareunternehmen bzw. IT-DL mit einer Standardanwendungsplattform? Im Folgenden werden diese Fragen zusammenfassend beantwortet.

Softwaregestaltung ist ein inkrementeller Wandel in Form einer **technikentwicklungsbezogenen Rationalisierung** von Software und ihres Anwendungsbereichs (8.6.3). Organisationen der Branche setzen das Mittel der Softwaregestaltung und -programmierung für die eigenen Zwecke ein. Besonderes Kennzeichen dieses Rationalisierungstyps ist der Fokus auf die Arbeitsteilung zwischen Softwareanwendung, -gestaltung und -programmierung, stetiges Abgleichen branchenspezifischer Bedarfe mit softwaretechnischen Möglichkeiten und die Dynamiken zwischen zentraler und dezentraler Softwaregestaltung. Organisationen prüfen ausgehend von ihrem Status quo, ob in einem Anwendungsbereich ein Standard oder eine individuelle Gestaltung zielführender ist, ob sie sich auf die Kontrolle der Anwendung einer Software konzentriert oder auch die Kontrolle des Arbeitsprozesses der Gestaltung übernimmt. Stetig ist abzuwägen zwischen niedrigen Kosten und ausgelagerter Softwaregestaltung für einen (automa-

tisierten) Standardsoftwareprozess – mit den entsprechenden Nachteilen, dass so z.B. nur begrenzt eine individuelle Prozessgestaltung möglich ist. Gleichzeitig ist eine Standardsoftware in Bereichen, in denen die Software keine Wettbewerbsvorteile bringt, zielführender und vor allem für jene EVU die bessere Wahl, die sich mit der internen Kontrolle der Softwaregestaltung schwertun. Zu einer individuellen Softwaregestaltung in der eigenen Organisation müssen EVU fähig sein und die entsprechenden Methoden und Beschäftigten organisieren können: sei es IT-Projekte durchzuführen, Scrum-Teams aufzubauen und Product Owner:innen oder Anforderungsmanagende einzustellen und mit Programmierenden zusammenzuarbeiten. Wie die Fallstudien zeigen, kann ein EVU unterschiedliche Entscheidungen für die vielzähligen Anwendungsbereiche treffen und jeweils mal eine industriespezifische Standardsoftware einsetzen und mal eine individuell gestalten.

Neben diesem kontinuierlichen Prozess, die Möglichkeiten der Softwaregestaltung zu prüfen, zeigen die Fallstudien, dass der Verlauf der digitalen Transformation davon geprägt ist, dass die EVU die Möglichkeiten der Softwaregestaltung unterschiedlich ausschöpfen. Allgemein verändern sich die EVU in den Fallstudien iterativ von ihrem Status quo aus. Wenige EVU ändern sowohl die Software als auch ihre Organisation und z.B. die Teamgrenzen, um die prozessübergreifende, interdisziplinäre Zusammenarbeit für die Softwaregestaltung zu verbessern. Die anwendenden Organisationen behalten ihre alten Strukturen aus Abteilungen und Hierarchien bei und etablieren die Softwaregestaltung quer dazu, so dass eine Matrixorganisation entsteht. Teils nutzen die Organisationen neuste digitale Werkzeuge, teilweise aber unterschiedliche Ticketsysteme je Abteilung oder nur Excel und E-Mails für die Softwaregestaltung. Organisatorisch hat nur INTERN1 die Anwendung reorganisiert und die Hierarchieebene der Meister:innen abgeschafft – und dass nicht für den Arbeitsprozess der Softwaregestaltung, sondern aufgrund der neu entwickelten Software. INTERN2 denkt über eine Reorganisation hin zu einer Prozessorganisation nach, die keine Abteilungs- und Teamgrenzen mehr kennt und einen Vorteil für den Arbeitsprozess der Softwaregestaltung bietet. Wie der Wandel von INTERN2 aussehen könnte, wenn z.B. Team-Silos aufgelöst sind und es nur noch Führungskräfte auf der Ebene des digitalen Gesamtprozesses gibt, zeigt das Beispiel einer auf Softwaregestaltung ausgerichteten softwaretechnischen Prozessorganisation (8.5.3). Bei EVU2 von KOOP2 fällt auf, wie sehr die alten Strukturen bestehen bleiben und die Arbeitsprozesse der Softwaregestaltung sich in diese einfügen müssen. Einige EVU von PAKET richten sich organisatorisch weniger an Abteilungen und Teams aus und mehr an übergreifenden Prozessen, z.B. indem sie übergreifende Projekte durchführen oder übergreifend die ERP-Software einstellen. Diese EVU haben aber, was die Standardsoftware anbelangt, wenige Einflussmöglichkeiten auf die Softwaregestaltung. Unternehmen wie STARTUP, die den Primat der Softwareentwicklung leben und sich von Anfang an als softwarebasierte Unternehmung begreifen, haben bei der Softwaregestaltung einen Vorteil gegenüber all jenen, die noch an althergebrachten Strukturen festhalten.

Die Firmen reizen die Möglichkeiten der Verwendung von Software je nach Anwendungsbereich unterschiedlich aus. Denn neben dem Verhältnis der beiden Arbeitsprozesse zueinander ist der Anteil der Datenverarbeitung im Anwendungsbereich entscheidend dafür, was die Softwaregestaltung überhaupt leisten kann und wie weit die digitale

Transformation geht. Kann die Arbeit weitgehend in Software abgebildet werden, dann ist eine weitgehende Automatisierung möglich und die Anwendenden bearbeiten vorwiegend (noch) nicht automatisierbare Restfälle. Ist die eigentliche Arbeit im Anwendungsbereich nicht digitalisierbar, geht es um automatisierte Steuerung von Arbeitskräften, deren Arbeit Software nicht erledigen kann. Dann werden die Anwendenden Teil eines digitalen Prozesses, der ihre Arbeit steuert, aber nicht ersetzt. In den Fallstudien liegt der Schwerpunkt auf Anwendungen, in denen nur Daten verarbeitet werden oder die Arbeit gesteuert wird, also nicht wie bei Hirsch-Kreinsen (2020) auf dem Einsatz in der Industrie zur Steuerung von Maschinen. Die Grenzen der Softwaregestaltung in einem Anwendungsbereich liegen in den Fallstudien darin, dass Organisationen häufig Software nur für einen Teil eines Prozesses gestalten. Andere Teile des Prozesses sind in der Verantwortung anderer Teams oder Firmen. Somit hängt die Durchschlagskraft der Softwaregestaltung auch davon ab, ob ein gesamter Arbeitsprozess der Anwendung unter ihrer Kontrolle ist oder nicht. Womit wir wieder beim Verhältnis der beiden Prozesse zueinander wären.

Kann der Arbeitsprozess der Softwaregestaltung auch einen disruptiven Verlauf der digitalen Transformation erklären? Für das Unternehmen aus der Fallstudie STARTUP ist Softwaregestaltung im Unterschied zu den anderen der Kern von Arbeit und Organisation. Es ist ein Musterbeispiel dafür, die Möglichkeiten individueller Softwaregestaltung auszureizen: Die Organisation basiert auf dem Primat der Softwareentwicklung. Die erbrachte Leistung, mit der das Unternehmen Geld verdient, ist weitestgehend in Software abgebildet. Zusätzlich bietet STARTUP die selbst entwickelte Software auch noch anderen an, verdient damit Geld und ist somit ein Beispiel für den oben beschriebenen Typ PROPRIETÄRER STANDARD. Trotzdem passt der Begriff disruptiv für STARTUP nicht. Warum? STARTUP bedient mit dem Emissionshandel für E-Mobilität nur eine Nische, von der aus eine Disruption der Energiebranche nicht möglich ist. Es gibt auch keine etablierte Konkurrenz, die durch die digitale Abbildung des Geschäfts zu verdrängen wäre.

Solche Nischen der Regulierung mit Software zu bedienen, ist typisch für die Branche (siehe Kapitel 7). In der Energiewirtschaft konnte sich in den Kerngeschäftsfeldern wie Netze, Lieferung, Handel und Erzeugung noch kein EVU allein durch eine individuelle Software (und deren Verkauf) durchsetzen (auch wenn einige EVU mit IT-Dienstleistungen Geld verdienen). Die meisten untersuchten EVU nutzen eine Mischung aus Standardsoftwarepaketen und individuell gestalteter Software. Viele EVU müssen die Möglichkeiten individueller Softwaregestaltung nicht nutzen, um Gewinne zu erwirtschaften und am Markt zu bestehen. Ob der Anteil an Standardsoftware in den EVU größer oder kleiner wird, ist eine offene Frage. Damit kann die vorliegende Untersuchung für die von Bessen (2022) beschriebene Strategie des »Competing on Complexity« (4.1) kein exemplarisches Unternehmen in der Energiewirtschaft finden. Die IT-Landschaften der EVU werden zwar umfangreicher. Es gibt allerdings keine Hinweise dafür, dass sie sich gerade dadurch entscheidend am Markt durchsetzen. So erhofft sich das EVU in Fallstudie INTERN1 dank individueller Software die Instandhaltung immer besser erledigen und sich so von konkurrierenden Firmen absetzen zu können, wenn es um die Konzessionsvergabe von Netzen geht. Ein EVU von KOOP1 versucht mit einem individuellen Portal für seine Kundschaft zu punkten. Was solch eine Softwaregestaltung wirklich an

zusätzlichen Marktanteilen oder Umsätzen bringt, müsste eine weitere Untersuchung erforschen.

Eine andere Form von starkem soziotechnischem Wandel wäre es, wenn sich eine Softwarefirma mit einer industriespezifischen Standardlösung durchsetzt, auf der EVU oder das Softwareunternehmen selbst sämtliche digitalen Kernprozesse der Branche abwickeln würden: eine Anwendungsplattform, auf der die große Mehrheit der Anwendenden der Branche arbeitet. Das wäre allerdings vor allem eine starke Veränderung für all jene Beschäftigten, die derzeit in den EVU oder IT-DL Software gestalten und programmieren, weil diese ihre Arbeit verlieren würden. Solch eine neue Softwarefirma, die mit ihrer Standardlösung den Markt erobert, zeichnet sich aktuell nicht ab. Zahlen dazu, wie viele Anwendende auf den aktuell existierenden Anwendungsplattformen der diversen IT-DL arbeiten, die auf den Standard-ERP-Systemen wie von SAP, SIV oder Schleupen basieren, oder wie viele Anwendende Standard-Cloud-Angebote nutzen, gibt es jedoch nicht (7.2.1). Das gilt auch für spezielle Standardsoftwarelösungen und Plattformen wie für den Energiehandel. Es ist auch nicht zu unterschätzen, dass Softwarefirmen intern nicht all das industriespezifische Wissen haben, um eine Standardsoftware zu entwickeln. Sie sind von der Kooperation einzelner EVU abhängig. Es sind große Investitionen notwendig, bevor eine Industriestandardlösung vorhanden ist. Neue anbietende Unternehmen wie Powercloud fangen deshalb erst damit an, für einzelne Geschäftsfelder oder Anwendungsbereiche Lösungen anzubieten.

Letztendlich hängt ein disruptiver Verlauf der digitalen Transformation noch von anderen Dingen wie der Softwaregestaltung ab. Hier kommt der dritte Ansatz ins Spiel, den Pfeiffer und Schrape (2023) anführen und dem die vorliegende Untersuchung nicht gefolgt ist, weil er weniger zur Fragestellung passt: Butollo et al. (2021) sehen Produktions- und Geschäftsmodelle als wichtig an, um die digitale Transformation zu untersuchen. Wenn z.B. ein Stromlieferant den Markt erobern will, indem sie eine Discounter-Strategie fährt, bietet es sich für sie möglicherweise eher an, auf eine Standardlösung zu setzen. Sie muss z.B. die Software nicht individuell gestalten, weil sie keine individuellen Produkte anbietet. Geschäftsmodelle in die Analyse einzubeziehen, könnte womöglich auch erklären, warum Stadtwerke, die seit über zwei Jahrzehnten eine eigene industriespezifische Standardsoftware haben, trotzdem nicht den Markt erobert haben: Weil es möglicherweise nicht Teil ihrer Strategie war?[1] Es ist einerseits naheliegend, dass die Möglichkeiten einer digitalen Disruption beschränkt sind, weil Teil des Produktionsmodells der Energiewirtschaft eine integrierte Netz-Infrastruktur, ein physikalisches Gesamtsystem (Produktion und Konsum von Strom müssen sich zu jedem Zeitpunkt die Waage halten) und eine starke Regulierung sind. Andererseits müsste das aber noch genauer untersucht werden. Eine offene Forschungsfrage ist darüber hinaus, inwiefern sich kommunale, staatliche und private Unternehmen in der digitalen Transformation unterscheiden.

1 In einigen Bundesländern sind den wirtschaftlichen Aktivitäten von kommunalen Unternehmen Grenzen gesetzt (7.1.2.3). Inwiefern dies in diesem Fall und in welchem Umfang bei der Digitalisierung eine Rolle spielt, wäre zu untersuchen.

9.3.3.3. Besonderheiten der digitalen Transformation in der Energiewirtschaft

Für die digitale Transformation der Energiewirtschaft ist die Regulierung ein besonderer Treiber und es gibt Besonderheiten in der Dynamik zwischen Individual- und Standardsoftware. Die zunehmende Digitalisierung der Branche zeigen zudem Zielkonflikte in ihrer Governance auf, wie der folgende Abschnitt ausführt.

Das Besondere an der Energiewirtschaft ist, dass ein Teil der digitalen Transformation weniger darauf zurückzuführen ist, dass Unternehmen mit Softwaregestaltung Märkte erobern oder die Energiewende vorantreiben können, sondern vielmehr darauf, dass die Energiewirtschaft mithilfe von Software zum Markt wird (7. Kapitel). Die von der staatlichen Regulierung verfolgte Markt-Governance seit 1999 für Strom und 2007 für Gas basiert auf Software. Das gilt für den Wechsel des energieliefernden Unternehmens durch Privatkund:innen per Webseite, die diversen Transparenzvorschriften, wie sie auf der Stromrechnung sichtbar sind (u.a. der Herkunftsnachweis für Ökostrom), den Datenaustausch zwischen den Tausenden von energiewirtschaftlichen Organisationen, den Energiehandel oder die Vermarktung von dezentralen Erzeugungsanlagen erneuerbarer Energie via virtuelle Kraftwerke. Die Regulierung basiert auf Softwareentwicklung und weil sie sich die letzten Jahrzehnte stetig geändert hat, ist auch stetige Softwaregestaltung notwendig.

Dabei zeigt sich eine besondere Dynamik aus individueller und Standardsoftware in der Energiewirtschaft. Das liegt wieder an der Regulierung. Für Software, die auf Regulierung beruhende Prozesse umsetzt, bietet sich eine Standardgestaltung an, weil diese Prozesse für alle Unternehmen gleich sind. Das ist ein Vorteil für die Zulieferindustrie von Software, die an der Gestaltung von Standardsoftware interessiert ist. Auf individuelle Softwaregestaltung setzen EVU eher im nichtregulierten Bereich des Energievertriebs, um sich von der Konkurrenz abzugrenzen. Daneben ist für die Dynamik aus Individual- und Standardsoftware in der Energiewirtschaft typisch, dass sie sich in organisationsübergreifenden, regionalen Kooperationen (KOOP1, KOOP2, KOOP3) für die Softwaregestaltung niederschlägt. In diesen verhandeln mehre Organisationen, was sie als gemeinsamen Standard und was jede für sich individuell umsetzt. Sowohl die Branchenanalyse im 7. Kapitel wie auch die Fallstudien legen zwar nahe, dass größere und mittlere EVU mehr Geld für Software ausgeben und stärker in die Softwaregestaltung involviert sind. Doch wäre der Zusammenhang zwischen EVU-Größe und Softwaregestaltung noch näher zu untersuchen. Dies könnte zum Beispiel geschehen, indem an erforscht, wann kleinere EVU nur fertige industriespezifische Standardpakete nutzen und wann und wie sie selbst Software gestalten.

Software und ihre Gestaltung ist Teil der Industrie-Governance der Energiewirtschaft geworden, weil sie eine IT-basierte Branche geworden ist. Indem die Grenzen zwischen IT- und Energiewirtschaft verschwimmen (z.B. indem EVU IT-DL werden, Softwarefirmen energiewirtschaftliche Geschäftsprozesse abwickeln oder digitale Startups entstehen) und die Markt-Governance so sehr auf digitalen Prozessen basiert, wird Software Teil der Governance-Strukturen der Branchen. Es geht nicht mehr nur darum, eine Energieinfrastruktur zu betreiben, sondern auch eine digitale Infrastruktur. Datenschutz, IT-Sicherheit, digitale Geschäftsfelder, die Macht einzelner Softwareunternehmen und IT-DL sowie der IT-Fachkräftemangel treiben die EVU um. Die enorme Bedeutung von Software schlägt sich auf den vier Ebenen der Industrie-Governance

nieder: Erstens prüfen Unternehmen ihre Strategien beim Softwareeinsatz – ob für das gesamte Unternehmen, einzelne Wertschöpfungsstufen oder Anwendungsbereiche. Zweitens sind Produkte wie Stromtarife der EVU digital verfügbar und lassen sich digital verkaufen. Drittens ist der Datenaustausch entlang der Wertschöpfungskette von Erzeugung, Handel, Netzen, Vertrieb softwarebasiert und es gibt unzählige Kooperationen zwischen EVU zur Entwicklung von Software. Viertens schlägt sich die gesteigerte Bedeutung von softwaretechnischer Interdisziplinarität und Wissensarbeit in einer Akademisierung der Beschäftigten nieder. Sie arbeiten zunehmend an automatisierten, softwarebasierten Prozessen. Die in der Branche weitverbreiteten Betriebsräte sind gefragt, bei der Softwaregestaltung und dem Softwareeinsatz die Interessen der Beschäftigten zu vertreten.

Bei der Debatte über Veränderung der Governance einer Industrie durch die Liberalisierung spielt die IT nur am Rande eine Rolle (vgl. Mayntz 2009). Doch entstehen durch die IT Zielkonflikte in der Branche. Sie resultieren daraus, dass Software Strukturen ermöglicht, erhält und schafft, die deutlich bürokratischer und damit teurer sind als jene vor der Liberalisierung. Lohnt sich der ganze digitale Aufwand, nur um wettbewerbliche Strukturen und einen europäischen Strommarkt zu etablieren, um selbst kommunale Versorgungsunternehmen dort einzugliedern? Denn die Energiepreise sind seit Beginn der Liberalisierung stetig gestiegen[2], viele Konsument:innen wechseln das Gas-/Stromlieferunternehmen selten oder nie[3] und die reduzierten CO_2-Emissionen lassen sich mehr auf andere, CO_2-arme Produktionsanlagen und weniger auf eine zunehmend digitale Organisation der Branche zurückführen. Für wen bietet die viele Arbeit an industriespezifisch gestalteter Software eigentlich welche Vorteile?

Wer nun womit Geld verdient (ob mit Software oder Energie), sich wie durchsetzt (dank Softwaregestaltung und/oder Softwareanwendung) und wie nachhaltig diese Strukturen sind: Für die einzelnen Betriebe stellt sich die Frage, wo die Reise der technikentwicklungsbezogenen Rationalisierung durch Softwaregestaltung in der Branche hingeht und ob jene EVU, die führend in der Softwaregestaltung sind, in Zukunft auch führend in der Branche sein werden. Zu einer soziotechnischen Organisation, welche die Möglichkeiten der Softwaregestaltung (ob individuell oder Standard) für ihre Zwecke einsetzt, gibt es aktuell in der Energiewirtschaft keine Alternative.

9.3.3.4. Folgen für die Arbeit in den EVU

Neben der soziotechnischen Arbeitsgestaltung durch Softwaregestaltung, die bedeuten kann, dass Anwendende den Bedarfen des Arbeitsprozesses der Softwaregestaltung untergeordnet sind, gibt es noch weitere Folgen für die Arbeit in den anwendenden EVU. Hirsch-Kreinsen beschreibt drei Entwicklungsszenarien der Digitalisierung von Arbeit: Substitution (Arbeit ersetzen), Upgrading (Qualifikationsniveaus steigen) und

2 Haushaltskunden: https://de.statista.com/statistik/daten/studie/914784/umfrage/entwicklung-der-strompreise-in-deutschland-verivox-verbraucherpreisindex/, abgerufen am 29.02.2024. Industriekunden: https://de.statista.com/statistik/daten/studie/252029/umfrage/industriestrompreise-inkl-stromsteuer-in-deutschland/, abgerufen am 29.02.2024.

3 https://de.statista.com/statistik/daten/studie/155532/umfrage/versorgerwechsel-der-haushalte-in-der-stromversorgung-seit-2005/, abgerufen am 29.02.2024.

Polarisierung (zunehmend Gering- und Hochqualifizierte, weniger Mittelqualifizierte) (vgl. Hirsch-Kreinsen 2020: 50ff.). Theoretisch können Softwaregestaltende zu allen drei Szenarien beitragen. Sie automatisieren Arbeit durch Softwaregestaltung und erhöhen das Qualifikationsniveau und den Anteil Hoch- und Mittelqualifizierter. Das ist in den Fallstudien allerdings nur zum Teil der Fall. So ersetzt die individuell gestaltete Software bei INTERN1 z.B. die Arbeit der Meister:innen in der Instandhaltung (Substitution). Anwendende wie Monteur:innen benötigen allerdings weiterhin ihre bisherigen Qualifikationen. In Anwendungsbereichen mit einem hohen Anteil an Datenverarbeitung und Automatisierung führt die Softwaregestaltung dazu, dass die Beschäftigten nur noch die komplizierteren, nicht automatisierbaren Restfälle bearbeiten und mehr interdisziplinär tätig sind, wofür exemplarisch die Rolle der Key User:innen steht (Upgrading). Ob ein zunehmender Bedarf an höherqualifizierten Softwaregestaltenden dazu führt, dass es weniger Beschäftigte mit mittlerer Qualifikation gibt (Polarisierung), zeigt sich nicht eindeutig. So erledigen in der Fallstudie STARTUP nur zwei bis drei Minijob-Beschäftigte einfache Arbeit, indem sie automatisierte Prozesse prüfen und ergänzen. Die restlichen ca. 18–22 Beschäftigten sind in der Mehrzahl mittel bis hoch qualifiziert und nahezu allesamt Akademiker:innen. Rechnet man jene Anwendenden, welche komplizierte Restfälle bearbeiten und internen Support für die Softwaregestaltung komplexer Softwarepakete wie ERP-Systeme leisten (Key User:innen), zu den Mittelqualifizierten, dann würden in solchen EVU wenige Geringqualifizierte übrigbleiben und die Mehrzahl wären mittel bis hoch qualifizierte Anwendende, Gestaltende und Programmierende.

In den Fallstudien fällt zudem auf, dass die Anwendenden der EVU vom Wandel der Software durch den Betrieb, für den sie arbeiten, geschützt sind. Selbst wenn Betriebe Anwendungsplattformen einsetzen, werden deren Anwendende nicht gleich zu Gig-Arbeitenden. Eindeutige Folgen der Softwaregestaltung für sämtliche Anwendende wie z.B. mehr Routinearbeit, eine stärkere soziale Isolation oder eine Dequalifizierung haben die Fallstudien nicht gezeigt. Für eine fundierte Aussage dazu wären quantitative Daten notwendig.

9.4. Methodische Grenzen und weiterführende Fragestellungen

Was hat das alles mit Max Weber zu tun, der am Anfang der Untersuchung zitiert wurde? Das wäre en détail noch herauszuarbeiten. Doch weder durch den Begriffsapparat von Weber noch von jenem z.B. von Giddens' Strukturationstheorie wollte sich die vorliegende Arbeit einschränken und überfordern lassen. Es war das primäre Ziel, eine eigene soziotechnische Begriffsbildung zu betreiben, welche den Besonderheiten der Softwaregestaltung gerecht wird. Mit welchen bestehenden theoretischen Ansätzen die hier entwickelten Konzepte am besten kompatibel sind, wäre eine noch zu leistende Aufgabe.

Das Methoden-Kapitel hat bereits weitere Grenzen der vorliegenden Untersuchung genannt. Darunter waren auch die allgemeinen Grenzen qualitativer Untersuchungen. Um die Grenze der Verallgemeinerung zu überwinden, wäre die Erforschung anderer Branchen naheliegend. Ziegler hat ähnliche Phänomene wie die in der vorliegenden Arbeit in einem großen Industriekonzern und dessen Bemühungen, im IoT-Bereich ein Geschäftsfeld aufzubauen, untersucht. Auch dort gibt es crossfunktionale Teams, wel-

che die Linienorganisation ersetzen (vgl. Ziegler: 264f.). Auch dort ist die Herausforderung, agile Methoden und standardisierte Prozesse in einer Organisation zu vereinen (vgl. ebd.: 266). Und auch dort wird reorganisiert, um »domänenübergreifende[…] Organisationseinheiten« (ebd.: 282) aufzubauen. Lässt sich das hier vorliegende Analyseschema also verallgemeinern auf sämtliche Formen und Orte der Softwaregestaltung? Wie und wer gestaltet Software für die Versicherungswirtschaft oder für Banken und gibt es dort auch vielfältige Kooperationen? Gestalten die softwareanwendenden Organisationen in anderen Branchen mehr oder weniger individuell? Sind sie mehr oder weniger abhängig in der soziotechnischen Arbeitsgestaltung der Softwareanwendung? Wie sieht eine größere Organisation in der Logistik, dem Online-Handel oder der Verwaltung aus, in der von Anfang an der Primat der Softwareentwicklung gilt? Wie unterscheidet sich z.B. die industriespezifische Softwareentwicklung zwischen Amazon und Zalando? Zudem wirkt der Sprung vom Arbeitsprozess der Softwaregestaltung zur industriespezifischen Anwendungsplattform noch groß. Die gesamte Organisation hinter der Anwendungsplattform hat z.B. die Fallstudie PAKET nicht untersucht. Es gibt sicherlich noch mehr Faktoren als die Softwaregestaltung, die maßgeblich dafür sind, dass eine Anwendungsplattform entsteht.

Zudem konnte die Frage nicht ausführlich beantwortet werden, welchen Unterschied es für die Tätigkeit der einzelnen Anwendenden macht, wenn sie mit einer individuell gestalteten Software arbeiten. Dafür müssten deutlich mehr Anwendende befragt werden und es wäre bspw. die Untersuchung unterschiedlicher Formen der Softwaregestaltung für den gleichen Anwendungsbereich notwendig. Es gibt wahrscheinlich auch Unterschiede zwischen Beschäftigtengruppen, was das IT-Budget für Softwaregestaltung angeht. Können Beschäftigte aus dem Energiebörsenhandel mehr Änderungen mit dem Hinweis einfordern, dass sie dadurch bessere Ergebnisse erzielen? Fällt dies Beschäftigten aus dem Kundenservice schwerer, weil sie ihre Tätigkeiten formalisierter und standardisierter erledigen müssen? Auch könnten Fragen von Partizipation vertieft werden. Es gibt eine seit langem geführte Diskussion dazu (vgl. Becker/Brinkmann 2017). Unterschiede zu Partizipationstypen wie jenem des »democratic Taylorism« (Adler 1995) könnten herausgearbeitet werden.

Es ist noch offen, wie sich das softwaretechnische, interdisziplinäre Wissen nun genau zusammensetzt: Der Unterscheidung von Hohlmann (2007) von Organisations-, Prozess- und Technologiewissen wurde in dieser Arbeit nicht gefolgt und anders als Hohlmann spricht diese Arbeit nicht von Integrationswissen, sondern von interdisziplinärem Wissen. Wie sieht aber dann die Zusammensetzung aus? Welches Wissen ist für wen in welcher Organisation nun zentral für die Softwaregestaltung: über den Anwendungsbereich, über ein einzelnes EVU, die Regulierung, das ERP-Paket und sein Customizing, über Programmierung, Methoden und Kompetenzen der Softwaregestaltung, spezifisches Wissen einzelner Anwendender? Wo spielt welches implizite Wissen eine Rolle, in welcher Phase der Softwareentwicklung und wo ist explizites Wissen über Regulierung oder Geschäftsprozesse wichtiger? Wie wichtig sind betriebliche Qualifikationen? Einerseits sind betriebsspezifische Qualifikationen nützlich (langfristige Zusammenarbeit mit externen Entwickelnden, langfristige Zusammenarbeit mit IT-DL, lange Betriebszugehörigkeit von Anwendenden, Programmierenden

oder Gestaltenden). Andererseits wechseln die EVU trotzdem den IT-DL und es gibt Personalfluktuation in den Organisationen.

Zudem bietet die vorliegende Forschungsarbeit vielfältige Anknüpfungspunkte zu anderen Forschungsgebieten wie zur Labour Process Theory. Dort gibt es zwar Forschungsarbeiten zur Softwareentwicklung, aber keine zum Arbeitsprozess der Softwaregestaltung. In der Organisationssoziologie sieht Carlile (2004) zur Überwindung von Wissensgrenzen ein gemeinsames Verständnis, eine gemeinsame Sprache, gute Beziehungen und gemeinsame Interessen als wichtig an. Welchen Beitrag kann die vorliegende Untersuchung zu diesem Forschungsfeld der Wissensgrenzen leisten? Es gibt Anknüpfungspunkte zum Thema Innovationen in Organisationen, zum Requirements Engineering bzw. Anforderungsmanagement in der Informatik oder zu neuen beruflichen Identitäten und Professionalisierungen in der Berufsforschung: Qualifikation, Karrierewege, Arbeitsweise, Gehalt und Tätigkeiten der Softwaregestaltenden. Eine tiefergehende Auseinandersetzung mit der Techniksoziologie erscheint notwendig, ob mit den Arbeiten von Pollock et al. (2007) zur »generification work« bei Standardsoftwarepaketen oder zur digitalen Transformation u.a. durch Plattformen (vgl. Dolata/Schrape 2023).

Es fehlt an Forschung zu Wertschöpfungsketten von Software in der Soziologie oder der Wirtschaftsgeografie. Was ist der Unterschied zu anderen Wertschöpfungsketten? Dafür sind andere Konzepte notwendig wie jene der Global Production Networks (Henderson et al. 2002, Coe et al. 2008) oder Global Value Chains (vgl. Gereffi et al. 2005). Denn Software ist nichts, was wie in der Automobilwirtschaft die Fahrzeughersteller (OEM) weiterverarbeiten oder montieren. Softwaregestaltung entscheidet über die Organisations- und Arbeitsgestaltung. Der Wechsel der zuliefernden Softwarefirma hat entsprechend andere Folgen, als wenn ein Unternehmen ein Teil einer Maschine von einem anderen Zulieferer bezieht. Die Untersuchung konnte auch nicht der Frage nachgehen, warum EVU sich für welche Möglichkeiten zwischen Standard- und Individualsoftwaregestaltung entscheiden. Spielt dabei das organisationale Feld der Energiewirtschaft (vgl. Bleicher 2006) eine Rolle, in dem eine Form der Isomorphie wirkt (vgl. DiMaggio/Powell 1983)?

Es wäre theoretisch zu klären, wie sich die Begriffe von Macht, Herrschaft, Koordination, Steuerung, Governance, Kontrolle und Transformation der Arbeitskraft analytisch trennscharf auseinanderhalten lassen. Dazu gehört, Ansätze wie die Unterscheidung zwischen technischer, bürokratischer und direkter Kontrolle (vgl. Edwards 1979), Input-, Verhaltens-, Ergebnis-, Clan- und Selbstkontrolle (vgl. Wiener et al. 2016), normative Kontrolle durch Professionen und Firmenkultur (vgl. Kunda 1992, Fleming/Sturdy 2011) und neuere Ansätze des algorithmischen Managements zu reflektieren und voneinander theoretisch abzugrenzen. Inwiefern sind die Ergebnisse dieser Arbeit anschlussfähig an Arbeiten von Vincent August zum Thema Kybernetik und Herrschaft durch Netzwerke (vgl. August 2019, August 2021)? Ist eine neue Theorie der Macht und Herrschaft notwendig, wenn Technik, Organisation und Arbeit so eng verzahnt sind, sich wechselseitig verändern und auf mehrere Organisationen verteilt sind?