RESEARCH

Dzifa Ametowobla

Zur Soziologie der Software

Die Rolle digitaler Technik bei der Kontrolle von Unsicherheiten

OPEN ACCESS

Springer VS

Zur Soziologie der Software

Dzifa Ametowobla

Zur Soziologie der Software

Die Rolle digitaler Technik bei der Kontrolle von Unsicherheiten

Springer VS

Dzifa Ametowobla
Technische Universität Berlin
Berlin, Deutschland

Zugl.: Berlin, Technische Universität, Dissertation, 2020

Diese Publikation wurde aus dem Open-Access-Publikationsfonds der Technischen Universität Berlin unterstützt.

ISBN 978-3-658-37255-2 ISBN 978-3-658-37256-9 (eBook)
https://doi.org/10.1007/978-3-658-37256-9

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Planung/Lektorat: Marta Schmidt
Springer VS ist ein Imprint der eingetragenen Gesellschaft Springer Fachmedien Wiesbaden GmbH und ist ein Teil von Springer Nature.
Die Anschrift der Gesellschaft ist: Abraham-Lincoln-Str. 46, 65189 Wiesbaden, Germany

Danksagung

Vor allem danke ich Arnold Windeler, der diese Arbeit von Anfang an begleitet und mich durch alle Fallstricke des Forschungsprozesses hindurch unterstützt hat. Mit seinem scharfen Blick für das Wesentliche hat er mir aus mehr als einer gedanklichen Sackgasse geholfen und mir gezeigt, dass sich meine Überlegungen nicht einfach im Kreis drehen, sondern spiralförmig auf ein Ziel zubewegen.

Großer Dank geht auch an Lutz Prechelt, der sich relativ spontan auf das Experiment eingelassen hat, die Zweitbetreuung für eine fachfremde Promotion im Endstadium zu übernehmen. In der Zusammenarbeit habe ich gelernt, die Komplexität von Software selbst besser zu verstehen und meine Gedanken dazu so in Worte zu fassen, dass dies auch anderen leichter gelingt.

Nina Baur danke ich für ihre Unterstützung vor allem am Anfang und am Ende der Promotionszeit und für gute Ratschläge, wann immer sie nötig waren.

Viele Gespräche mit den Kolleg*innen an der TU Berlin haben Aspekte dieser Arbeit und mein Verständnis von Soziologie geprägt. Einige von ihnen haben mir mit Kommentaren zu Entwürfen der Dissertation und anderen Texten geholfen, meine Gedankengänge zu sortieren und deren Darstellung zu verbessern. Für inhaltliche Hinweise, kontroverse Debatten und moralische Unterstützung danke ich vor allem Matthias Bottel, Christopher Grieser, Anina Engelhardt, Henning Mohr, Robert Jungmann, Emily Kelling, Fabian Schroth, Julian Stubbe und den Mitgliedern des Write Clubs am Institut für Soziologie.

Die Förderung im DFG-Graduiertenkolleg *„Innovationsgesellschaft heute. Die reflexive Herstellung des Neuen“* hat es mir ermöglicht, mich zu Beginn der Promotionszeit ganz auf meine Doktorarbeit zu konzentrieren. Dank dieser Freiheit konnte ich grundlegende Aspekte der Forschungsfrage in Ruhe ausformulieren und so das Projekt auf ein solides Fundament stellen.

Wer promoviert vergisst manchmal, dass das Leben aus mehr als Wissenschaft besteht. Zum Glück habe ich Freunde und eine Familie, die Verständnis für diesen Tunnelblick hatten und es geschafft haben, mich immer wieder daran zu erinnern, was sonst noch wichtig ist. Ich danke Sarah Waterfeld für ihr kritisches Lektorat und für Unterstützung, Spaß und Streit, wann immer es nötig war. Mein größter Dank geht an Andreas, für einfach alles.

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung 1

Software ist spätestens seit Mitte des 20. Jahrhunderts elementarer Bestandteil moderner Gesellschaften. Als Forschungsgegenstand der Soziologie ist sie lange nur punktuell in einzelnen Teilbereichen der Disziplin aufgetaucht (z. B. Suchman 1987; Ortmann et al. 1990; Orlikowski und Robey 1991; Rammert et al. 1998) oder wurde dann zum Thema, wenn aufsehenerregende Gesellschaftsanalysen (z. B. Beniger 1986; Zuboff 1988) die Aufmerksamkeit von Soziolog*innen kurzzeitig auf das Vordringen digitaler Technik in neue gesellschaftliche Bereiche lenken konnten. Mit der Zeit ist so eine Reihe an soziologischen Erkenntnissen über das Thema Software entstanden. Eine *Soziologie der Software*, also einen permanenten Diskurs und eine systematische Wissensbasis über die Rolle, die Software in modernen Gesellschaften spielt und über Vorgehensweisen, mit denen diese Rolle soziologisch untersucht werden kann, gibt es bisher jedoch nicht. Es ist Zeit, das zu ändern.

Erst in den letzten Jahren ist die Frage, wie der Umgang mit digitaler Technik unsere Gesellschaften verändert, zu einem Dauerthema in verschiedenen soziologischen Debatten geworden. Grundsätzlich scheint Einigkeit darüber zu bestehen, dass die zunehmende Rolle, die digitale Technik in allen Lebensbereichen spielt, für die Soziologie als Ganzes relevante Fragen aufwirft (Jarke 2018). Unklar ist, ob die Beziehung zwischen digitaler Technik und Sozialem sich qualitativ von ihrem Pendant im Analogen unterscheidet (Baecker 2017) oder ob nur allgemeine Zusammenhänge von Technik und Sozialem heute mehr Publikum finden (Dolata 2011), weil digitale Technik uns überall vor Augen führt, wie sehr Soziales und Technisches zusammenhängen. Klar ist jedoch, dass dieser Zusammenhang kein Nischenthema mehr ist, wenn es um digitale Technik geht.

D. Ametowobla, *Zur Soziologie der Software*,
https://doi.org/10.1007/978-3-658-37256-9_1

Bei der Untersuchung dieses Zusammenhangs stellen Soziolog*innen regelmäßig das „Digitale“ heraus (Funken und Schulz-Schaeffer 2008; Kallinikos et al. 2013; Tilson et al. 2010; Hirsch-Kreinsen 2015; Kirchner und Beyer 2016; Kropf und Laser 2019). Dabei identifizieren sie bestimmte Eigenschaften dieses „Digitalen“ als sozial relevante Einflussfaktoren und untersuchen empirisch Veränderungen, die diesen Einflussfaktoren zugeschrieben werden. Ob die gewählten Eigenschaften aber auch widerspiegeln, was das „Digitale“ im Sozialen bedeutsam macht, wird selten thematisiert. Die Frage, was an der digitalen Technik für das Verhältnis zwischen sozialem Handeln und sozialer Ordnung tatsächlich einen Unterschied macht, ist offen. Häufig wird thematisiert, dass digitale Technik nicht nur Informationsverarbeitung und Kommunikation verändert, sondern auch neue Wege eröffnet, um Kontrolle über Soziales auszuüben (Beniger 1986; Kallinikos 2005).

Zwei schwer zu vereinbarende Narrative durchziehen dabei (auch) die soziologische Forschung: Auf der einen Seite betrachten Wissenschaftler*innen digitale Technik als „*multi-sited associative or concatenated entity*“ (Mackenzie 2006), als Nicht-Technik, ungreifbares Verbindungsglied zwischen sozialen Kontexten, das in nie dagewesener Form Beziehungen über Zeit und Raum hinweg verknüpft, sich dauerhaft und überall verändert und dadurch keine eigene Stabilität besitzt. Auf der anderen Seite zeigen empirische Untersuchungen, dass dieser angeblich so flexible Gegenstand denen, die mit ihm umgehen, häufig erstaunlich dauerhafte Widerständigkeit entgegenbringt (Ortmann et al. 1990).

Dieser scheinbare Widerspruch ergibt sich nicht allein aus der Komplexität des Forschungsgegenstands, sondern auch aus einem fehlenden Austausch zwischen Forscher*innen, die mit verschiedenen Perspektiven auf das Verhältnis von digitaler Technik und Sozialem blicken. Wenn gefragt wird, wie soziale Faktoren die Entwicklung digitaler Artefakte und den Umgang mit ihnen beeinflussen, scheint Software dem engsten Wortsinn nach eine weiche Technik zu sein: extrem wandelbar und geeignet, alle Vorstellungen abzubilden und sich an beliebige Umgebungen anpassen zu lassen. Wenn aber untersucht wird, wie sich privater Alltag, Organisationen und Gesellschaften nach der Einführung von digitaler Technik verändern, werden die „harten“ Aspekte von Software stärker in den Vordergrund gerückt: dass sie bestimmte Umgangsformen vereinfacht und andere erschwert, und zwar aus Gründen, die keineswegs zufällig sind und kaum von den Akteur*innen abhängen, die davon am meisten beeinflusst werden.

Obwohl sich Soziolog*innen spätestens seit der ersten Welle der Informatisierung in den 1980er Jahren mit der Entstehung, Verbreitung und Nutzung digitaler Technik beschäftigen (Zuboff 1988; Orlikowski 1992; Weltz und Ortmann 1992; Rammert et al. 1998; Funken und Schulz-Schaeffer 2008; Bowker 1997; Pollock

und Williams 2009; Suchman 1987; Star und Ruhleder 1996; Fulk und Steinfield 1990; Leonardi et al. 2012; Kallinikos et al. 2013; Houben und Prietl 2018), gibt es bis heute kein gemeinsames Forschungsfeld, in dem unterschiedliche Perspektiven auf den Gegenstand systematisch zusammengeführt würden. Stattdessen werden traditionell vor allem in der Organisations- und der Techniksoziologie getrennte Debatten über Software geführt, die sich nur gelegentlich und in Teilen berühren. Seit einigen Jahren existiert zwar ein eigenes Feld der Software Studies in den Science and Technology Studies (STS), das punktuell an die vorangehenden Arbeiten anknüpft. In diesem können die soziologischen Diskurse aber nur begrenzt integriert werden, weil die Grundannahmen darüber, was „Soziales“ und „Technisches“ ausmacht, von denen in der klassischen Soziologie abweichen.

Mit der vorliegenden Arbeit stelle ich mir die Aufgabe, Erkenntnisse aus diesen verschiedenen Debatten zusammenzuführen und dadurch eine Antwort auf die Frage anzubieten, wie digitale Technik im Sozialen relevant wird. Ich stelle dabei die Themen Macht und Kontrolle in den Mittelpunkt und betrachte Software als Artefakt, welches immer auch geschaffen und genutzt wird, um die Unsicherheiten des Sozialen zu kontrollieren. Mit diesem Fokus schließe ich an eine Tradition der soziologischen Auseinandersetzung mit digitaler Technik an (Beniger 1986; Zuboff 1988). Gleichzeitig nehme ich dadurch die primär instrumentelle Perspektive auf, mit der Software in der Regel von Informatiker*innen und auch von den meisten anderen Akteur*innen betrachtet wird, die sie konstruieren und nutzen. Diese instrumentelle Perspektive basiert auf einer Annahme, die mit dem konstruktivistischen Blick, mit dem ich in der vorliegenden Arbeit auf Software schaue, deutlich erkennbar wird. Diese Annahme lautet: Soziales kann über Software gezielt kontrolliert werden. Worauf diese Annahme fußt, wer sie aufstellt und welche Auseinandersetzungen geführt werden, um sie aufrecht zu erhalten, wird in dieser Arbeit beschrieben. Mein Ziel ist es, eine soziologische Perspektive auf Software zu entwickeln, mit der diese Auseinandersetzungen sichtbar gemacht werden können. Diese Perspektive soll es Soziolog*innen ermöglichen, die Besonderheiten digitaler Technik ernst zu nehmen, ohne ihre Bedeutung für das Soziale zu überhöhen.

Mein Ausgangspunkt ist dabei die an Max Webers (2005 [1922]) Soziologieverständnis angelehnte Überzeugung, dass eine soziologische Perspektive das Verhältnis von sozialem Handeln und sozialer Ordnung ins Zentrum stellen muss. Digitale Technik wird dieser Überzeugung nach insofern zum relevanten Forschungsgegenstand der Soziologie, als sie *in dieses Verhältnis eingebettet* ist. Aus einer in diesem Sinne soziologischen Perspektive wird digitale Technik nicht losgelöst von sozialer Ordnung oder sozialem Handeln betrachtet, auch dann nicht, wenn in konkreten Studien auf einen der Aspekte fokussiert wird. Stattdessen

wird digitale Technik als Mittel und Produkt des Verhältnisses von Handeln und Ordnung untersucht und danach gefragt, welche Rolle sie in diesem Verhältnis spielt.

Eine erste Schwierigkeit bei einer allgemeinen soziologischen Auseinandersetzung mit digitaler Technik ist, dass der Gegenstandsbereich so unterschiedliche Phänomene umfasst. Sowohl einfache Geräte (z. B. Digitalwecker) als auch komplexe Systeme (z. B. GPS oder automatisierte Übersetzungsprogramme) beinhalten digitale Technik. Ihr Einsatz ist heute in beinahe jedem sozialen Prozess selbstverständlich (z. B. in der Industrieproduktion) oder zumindest vorstellbar (z. B. bei der Partner*innensuche). Eine zweite Schwierigkeit liegt darin, dass digitale Technik komplex ist. Besonders für technische Lai*innen, wie Soziolog*innen es in der Regel sind, ist schwer verständlich, aus welchen Elementen digitale Technik besteht und wie diese Elemente zusammenspielen. In der Soziologie besteht daher keine Einigkeit darüber, was fokussiert werden soll, wenn die komplexen sozio-technischen Phänomene untersucht werden, in denen digitale Technik eine Rolle spielt. Soziolog*innen stellen in ihrer Forschung zum Beispiel Algorithmen (z. B. Seyfert und Roberge 2017), Daten (z. B. Houben und Prietl 2018), Informationssysteme (z. B. Walsham 1993) oder Computer (z. B. Rammert et al. 1998) in den Mittelpunkt und diskutieren die Bedeutung ihrer jeweiligen Forschungsgegenstände für das Soziale in verschiedenen, größtenteils getrennten Debatten. In jeder dieser Debatten werden plausible Argumente benutzt, um zu begründen, warum der jeweilige Fokus auf digitale Technik deren sozial relevante Charakteristika erfasst. Die möglichen Alternativen, die sich in den anderen Debatten ja finden lassen, werden jedoch nicht diskutiert. So bleibt ungeklärt, in welchem Verhältnis die Forschungsgegenstände der einzelnen Debatten zu einander stehen, was eine Zusammenführung der Ergebnisse erschwert. Daher beginne ich meine Arbeit mit einer Bestimmung der zentralen Begriffe, die in der Soziologie genutzt werden, um die sozial bedeutsamen Charakteristika digitaler Technik zu erfassen, und ihrer Beziehungen zueinander. Wie bereits mit dem Titel der Arbeit signalisiert wird, gehe ich davon aus, dass sich diese Charakteristika mit dem Begriff der Software am besten erfassen lassen. Dies wird im folgenden Abschnitt begründet.

Definitionen

In der vorliegenden Arbeit identifiziere ich Software als das Element digitaler Technik, über welches sich ihre Rolle für das Verhältnis von sozialem Handeln und sozialer Ordnung am besten untersuchen lässt. Unter *Software* verstehe ich dabei Folgendes:

> *„Software consists of lines of code: Instructions and algorithms that, when combined and supplied with appropriate input, produce routines and programs capable of complex digital functions. Put simply, software instructs computer hardware – physical, digital circuitry – about what to do (...)“ (Kitchin und Dodge 2011, S. 3)*

Wie auch in der Informatik üblich, ist Software damit zuerst über ihren Unterschied zu Hardware bestimmt: Während die *Hardware* der digitalen Technik aus physischen Bauteilen besteht, die elektrische Spannungen verarbeiten, beinhaltet die Software alle Anweisungen[1], die diese Bauteile steuern und damit die Funktionen verursachen, die von der Hardware ausgeführt werden. Ein *Computer* setzt sich aus Soft- und Hardware zusammen, ein *Informationssystem* besteht wiederum aus möglicherweise mehreren Computern, die auf den gemeinsamen Einsatz in einem sozio-technischen System ausgerichtet sind[2]. In Software wird durch *Codezeilen* festgelegt, wie ein Computer unter welchen Umständen mit Eingaben verfahren soll, die aus dem Informationssystem oder von dessen Nutzer*innen stammen. Der Begriff Software schließt damit einerseits *Daten* (als Informationen über die Umwelt in für den Computer bearbeitbarer Form) und *Algorithmen* (als Anweisungen dafür, wie mit diesen Daten im Computer umgegangen werden soll) als Elemente ein und ist andererseits selbst Element von *Computern* und *Informationssystemen*. Wie im Verlauf der Arbeit gezeigt wird, lässt sich die Art und Weise, wie digitale Technik in das Verhältnis von sozialem Handeln und sozialer Ordnung eingebettet ist, anhand des Softwarebegriffs spezifischer untersuchen, als dies mit den anderen hier erwähnten Begriffen möglich ist. Die statischen Aspekte der Software werde ich als *Code* (*Codezeilen*), die dynamischen Aspekte als *Funktionen*[3] bezeichnen.

---

[1] Für den hier verfolgten Definitionszweck lassen sich Instruktionen und Algorithmen gleichermaßen als Anweisungen der Software an die Hardware verstehen. Ein Algorithmus ist eine *„detaillierte und explizite Vorschrift zur schrittweisen Lösung eines Problems“* (Herold et al. 2011, S. 146). Instruktionen sind dann einfache Anweisungen, die einen Schritt beschreiben, Algorithmen komplexe Anweisungen, die aus mehreren Instruktionen bestehen.

[2] Statt Informationssystem wird häufig auch von IT-System oder IuK-System gesprochen, in der Regel ohne explizite Definition. Ich folge hier der im Forschungsfeld Information Systems (IS) Research üblichen Praxis, in der als Informationssystem die Elemente digitaler Technik in einem sozio-technischen System betrachtet werden, die aufeinander und auf den Einsatz im sozio-technischen System abgestimmt sind. Diese Praxis der Begriffsverwendung ist trotz gelegentlicher Forderungen nach einem umfassenderen Verständnis des Begriffs (vgl. Walsham 1993, S. viii) dominant und bietet sich auch für diese Arbeit an, in der die Beziehung zwischen Technischem und Sozialem in sozio-technischen Systemen fokussiert wird.

[3] Als „Funktionen“ bezeichne ich das Verhalten des Computers, welches auftritt, wenn Software ausgeführt wird. Funktionen sind also (prinzipiell) beobachtbare Reaktionen des Computers auf die durch den Code beschriebenen Anweisungen zur Laufzeit der Software.

Eine Menge aufeinander Bezug nehmender Codezeilen, die dafür geschaffen sind, unter den gleichen technischen Bedingungen ausgeführt zu werden, um bestimmte Funktionen auszulösen, wird in dieser Arbeit *Programm* genannt.

Nach dieser kurzen Begriffsbestimmung werde ich im Folgenden die zentralen Linien der soziologischen und soziologie-nahen Debatten skizzieren, in denen Forscher*innen sich mit der Rolle von Software im Sozialen beschäftigt haben. Diese Debatten bilden den Hintergrund für die Forschungsfrage der vorliegenden Arbeit und für die soziologische Perspektive auf Software, die hier entwickelt werden soll. Beides wird im Anschluss an den Überblick über die bestehende Forschung ausformuliert.

Software in soziologischen Debatten

In der Technik- und der Organisationssoziologie ist die Rolle von Software im Sozialen seit mehreren Jahrzehnten immer wieder thematisiert worden. Forscher*innen in beiden Teildisziplinen haben dabei grundlegende Erkenntnisse darüber produziert, wie soziale Bedingungen beeinflussen, wie Akteur*innen Software entwickeln und nutzen und wie diese Nutzung auf das Soziale zurückwirkt. In beiden Teildisziplinen wurde außerdem Grundlegendes darüber herausgefunden, wie Technik allgemein im Sozialen und speziell in Organisationen bedeutsam wird. Solche Grundlagen, diesmal ausschließlich zu Software und dem Sozialen, finden sich auch in zwei interdisziplinären Forschungsfeldern, nämlich in der Forschung zu Computer Supported Cooperative Work (CSCW) und in den Software Studies. Diese Grundlagen aus den vier Forschungsgebieten betrachte ich als die Basis, auf der eine Soziologie der Software aufgebaut werden kann. Im Folgenden stelle ich diese Basis vor und zeige, welche Forschungslücken ich in den einzelnen Debatten in Bezug auf die Rolle von Software im Sozialen sehe. Im Anschluss an diesen kurzen Überblick werde ich dann ausformulieren, welchen Beitrag ich mit dieser Arbeit leisten will, damit auf dieser Basis eine Soziologie der Software aufgebaut werden kann.

Techniksoziologie

Bevor die Besonderheiten *digitaler* Technik untersucht werden können, ist es nötig, zuerst Klarheit über die Beziehung von Technik und Sozialem im Allgemeinen zu bekommen. Traditionell teilt sich die techniksoziologische Forschung in Arbeiten, in denen die Beziehung von Technik und Sozialem für Prozesse der Technikentwicklung beleuchtet wird, und andere, in denen Prozesse der Techniknutzung im Vordergrund stehen. Nur in wenigen Arbeiten wird beides zugleich betrachtet.

In der Forschung zur Technikentwicklung lassen sich drei Arten von Fragestellungen identifizieren. Sie unterscheiden sich vor allem dadurch, wie genau sie die konkreten Eigenschaften technischer Artefakte thematisieren: (1) Welche sozialen

Rahmenbedingungen wirken auf Prozesse der Technikentwicklung ein? (2) Was für übergeordnete Konzepte nutzen Technikentwickler*innen, um die gemeinsame Arbeit zu koordinieren? (3) In welcher Beziehung stehen solche übergeordneten Konzepte zu den konkreten Eigenschaften der technischen Artefakte, die entwickelt werden?

(1) Eine der wichtigsten Grundlagen für die Untersuchung des Einflusses sozialer Rahmenbedingungen auf Prozesse der Technikentwicklung stellt der Ansatz der Social Construction of Technological Systems (SCOT) dar (Pinch und Bijker 1987). Diesem zu Folge entstehen Artefakte quasi-evolutionär dadurch, dass sich verschiedene soziale Gruppen miteinander und mit ihren jeweiligen Vorstellungen von der für ihre Zwecke erwünschten Gestalt der Technik auseinandersetzen. Solche Gruppen bilden sich um die geteilte Interpretation eines entstehenden Artefakts und der von ihm zu lösenden Probleme. Durch eigene Konstruktionen oder Einflussnahme auf Ingenieur*innen und Unternehmer*innen, die die Technik herstellen, wirken diese Gruppen darauf hin, dass das Artefakt für bestimmte Zwecke optimiert wird. Wozu es dienen und nach welchen Kriterien es weiterentwickelt werden soll, kann dabei ebenso von den Vorstellungen kleinerer Nutzer*innengruppen (Pinch und Bijker 1987; Bijker et al. 1987) wie von den ökonomischen und politischen Interessen einflussreicher institutioneller Akteur*innen abhängen (Bijker et al. 1987, S. 199–268). Im Laufe der Zeit setzt sich eine der vielen möglichen Interpretationen des Artefakts durch, die *interpretative Flexibilität* der Anfangsphase verschwindet und es kommt zur *Schließung*, also zur diskursiven Einigung auf dominante Zwecke und Funktionen des Artefakts. Beides, interpretative Flexibilität und Schließung, werden vor allem als diskursive Prozesse betrachtet. Konkrete Eigenschaften der Technik sind Ergebnisse dieser Diskurse und nicht „an sich“ sozial bedeutsam, also nicht, ohne dass sie von Menschen in Diskurs oder Praxis aufgegriffen werden (Grint und Woolgar 1992, S. 367)[4]. Dass Artefakte Eigenschaften mitbringen, die den Verlauf der diskursiven Prozesse während der Entwicklung beeinflussen könnten, ist in diesem Ansatz nicht vorgesehen.

(2) Während bei SCOT ursprünglich davon ausgegangen wird, dass sozial relevante Gruppen anhand geteilter Vorstellungen vom Artefakt identifiziert werden können (Pinch und Bijker 1987, S. 30), sind spätere Arbeiten auch der Frage gewidmet, wie diese Vorstellungen innerhalb der sozial relevanten Gruppen entstehen, aufrechterhalten werden und die Kooperation in den Gruppen strukturieren. Auf dieser Ebene der Betrachtung steht der *technological frame* im Fokus. Als solcher wird

[4] „[T]he technical capacity of technology is essentially indeterminate […] and any particular determination is the upshot of social construction.“ (Grint und Woolgar 1992, S. 367).

eine Sammlung kognitiver und pragmatischer Konzepte bezeichnet, die gemeinsame Problemdefinitionen, Ziele, Heuristiken zu Zweck-Mittel-Zusammenhängen und Designprinzipien umfassen und an denen sich Mitglieder der Gruppe orientieren (Bijker 1995, S. 119 ff). Die Existenz dieses Systems an geteilten Deutungen ist die Voraussetzung dafür, dass für die Technikentwicklung relevante Interaktionen innerhalb der sozial relevanten Gruppe stattfinden können. Die Herausbildung eines gemeinsamen technological frame ist damit ein entscheidender Teil der frühen Technikgenese. Er stellt Einigkeit über den Weg her, der beschritten werden muss, um ein bestimmtes Artefakt herzustellen. Der technological frame ist damit ein soziales Ordnungsmuster auf mittlerer Ebene, das soziales Handeln ermöglicht, indem es den Akteur*innen erlaubt, Fragen nach den Auswirkungen dieses Handelns auf übergeordnete Muster sozialer Ordnung zurückzustellen und einfach tätig zu werden.

Über technological frames werden mögliche Handlungsoptionen für den Umgang mit konkreten Problemen, die im Prozess der Technikentwicklung aufkommen können, definiert und damit die Entscheidungen in diesem Prozess direkt vorstrukturiert. Im Gegensatz dazu stehen beim Begriff des *Leitbilds* die erwarteten Auswirkungen der Technikentwicklung auf die soziale Ordnung im Vordergrund. Leitbilder sind

> *„Vorstellungen über gegebene oder herstellbare technische Möglichkeiten [...], die sich zu vorausdeutenden Technikentwürfen verdichten und als wahrnehmungs-, denk-, entscheidungs- und handlungsleitender Orientierungsrahmen für individuelle und kollektive Akteure in technikgenetischen Prozeßnetzwerken wirken." (Dierkes et al. 1992, S. 11)*

Leitbilder bringen also Erwartungen über mögliche gesellschaftliche Folgen einer (noch zu konstruierenden) Technik zum Ausdruck, an denen sich Akteur*innen orientieren, die durch die Entwicklung von Technik Soziales verändern wollen. In der Technikgeneseforschung wird davon ausgegangen, dass diese gewöhnlich vagen Zukunftsbilder vor allem drei Zwecke erfüllen: Erstens helfen sie dabei, Kommunikationsbarrieren zwischen Akteur*innen mit unterschiedlichen Wissenshintergründen abzubauen, zweitens dienen sie dazu, Ziele und Handlungen zu vereinheitlichen und drittens helfen sie dabei, in Entwicklungsprojekten mit ungewissem Ausgang die Motivation aller Beteiligten aufrecht zu erhalten (Dierkes et al. 1992). Leitbilder ermöglichen es unterschiedlichen Akteur*innen, die bei der Technikentwicklung mitwirken, abstrakte Wert- und Zielvorstellungen mit konkreten, prototypischen Lösungsmustern und damit auch mit den Artefakten in Verbindung zu setzten, die aus ihrer jeweiligen Arbeit hervorgehen (Hellige 1996). Anders als technological frames vereinheitlichen Leitbilder Vorstellungen über die Ziele der

Technikentwicklung, legen jedoch die Wege, die dorthin führen, nicht konkret fest. Verschiedene, miteinander unvereinbare Lösungsmuster können unter dem gleichen Leitbild versammelt werden, ohne dass dies den Entwicklungsprozess beeinträchtigen muss (a.a.O., S. 26 f). Dies zeigt, dass Leitbilder für den konkreten Umgang mit der Technik zweitrangig sind: In mehreren Studien zur Entwicklung von „Expertensystemen" (Rammert et al. 1998) zeigt sich, dass Leitbilder zwar erfolgreich genutzt werden, um im o.a. Sinn das Handeln verschiedener Akteur*innen auf ein gemeinsames Entwicklungsziel hin auszurichten. Der konkrete Bezug zu den Artefakten ist jedoch gering, denn die Leitbilder *„prägen stärker die Rhetorik und weniger die Praxis"* (Rammert 2000, S. 90) bei der Technikentwicklung. Sie bieten nur *„Visionen der Technisierung"* (a.a.O., S. 86) an, also Fernziele, auf die Akteur*innen hinarbeiten. Die Entscheidungen darüber, wie diese Visionen zu erreichen sind, werden jedoch stärker von größtenteils impliziten *„Konzepten der Konstruktion"* (a.a.O.) beeinflusst, die in den professionellen Paradigmen der Entwickler*innen, ihren praktischen Erfahrungen und konkreten Rahmenbedingungen der Arbeit gründen. Während die Konzepte der Konstruktion grundsätzliche Entscheidungen über Eigenschaften der Technik vordefinieren, sind die Details der Konstruktion durch das *kulturelle Modell* bestimmt. Dieser Begriff bezeichnet die Vorstellungen der Entwickler*innen davon, wie ihre Nutzer*innen sind, was sie können und wollen (a.a.O., S. 87).

(3) In den vorgestellten Ansätzen wird der Stellenwert übergeordneter Deutungs- und Handlungsmuster für die Technikentwicklung thematisiert. Es wird aber wenig darüber ausgesagt, wie diese konkrete Handlungsentscheidungen von Entwickler*innen beeinflussen. Erst über solche Handlungsentscheidungen in den Entwicklungsprozessen werden übergeordnete Deutungs- und Handlungsmuster in konkrete Eigenschaften technischer Artefakte übersetzt. Um den Zusammenhang von sozialen Ordnungsmustern und konkreten technischen Eigenschaften geht es primär in Forschungsarbeiten, in denen untersucht wird, wie das Nutzer*innenbild von Entwickler*innen ihre Entscheidungen über technische Eigenschaften der Artefakte prägt, die sie konstruieren (Grint und Woolgar 1992; Oudshoorn et al. 2004; Hyysalo et al. 2016; Woolgar 1991). Die Grundideen in diesem Forschungsstrang stelle ich hier beispielhaft anhand der Arbeit von Nelly Oudshoorn, Els Rommes und Marcelle Stienstra (2004) vor. Die Autorinnen vergleichen mehrere Entwicklungsprozesse, in denen die Auseinandersetzung mit der Frage, was zukünftige Nutzer*innen ausmachen könnte, entfällt, weil die Technik sich an „everybody" richtet. Ein solch maximal unspezifisches Nutzer*innenbild führt dazu, dass Entwickler*innen die „Ich-Methodologie" anwenden: Die explizite Konstruktion der Nutzer*in als „everybody" führt in der Praxis dazu, dass die Nutzer*in implizit als Spiegelbild der an der Entwicklung beteiligten Gruppe konstruiert wird. Diese

Gruppe ist nicht im Geringsten repräsentativ für die tatsächlich anvisierte Gruppe der Nutzenden, was dazu führt, dass die fertiggestellte Technik die Anforderungen der Nutzer*innen in der Praxis nicht erfüllt. Die konkreten Eigenschaften der Technik und damit die Möglichkeiten, die deren Nutzer*innen später im Umgang mit ihr haben, ergeben sich durch dieses Vorgehen aus den unreflektierten Vorlieben der Entwickler*innen. Oudshorn et. al schlussfolgern, dass sich in diesen Eigenschaften mehrere Ebenen sozialer Ordnung widerspiegeln: politische und ökonomische Rahmenbedingungen, die dazu führen, dass potentielle zukünftige Nutzer*innen nicht in irgendeiner Form in den Entwicklungsprozess einbezogen werden, sowie die Geschlechterordnung. Dieser schreiben die Autor*innen sowohl die Zusammensetzung des Entwicklungsteams als auch die Vorlieben der Entwickler*innen für bestimmte technische Eigenschaften zu (Oudshoorn et al. 2004).

Die Vorstellung, dass Nutzer*innen durch konkrete Eigenschaften der Technik „konfiguriert“ werden (Woolgar 1991) schlägt eine Brücke von der Technikentwicklungs- zur Techniknutzungsforschung. Auch hier finden sich mehrere Strömungen, die sich darin unterscheiden, wie viel Bedeutung konkreten Eigenschaften von Artefakten zugemessen wird, wenn es darum geht zu verstehen, weshalb Nutzende auf eine bestimmte Art mit Technik umgehen. Sehr dominant ist hierbei die Perspektive auf den Konstruktivismus, die von den Anhängern der SCOT vertreten wird. Dieser zu Folge führt die Betrachtung konkreter Eigenschaften von Technik in die Irre, weil davon ausgegangen werden muss, dass diese Eigenschaften nur das Ergebnis einer durch Diskurse und routinisierte Handlungsmuster geprägten Interpretation der Technik durch die Nutzenden sind, aber nicht für sich stehen (David und Pinch 2008; Grint und Woolgar 1992). Auch wenn theoretisch zugestanden wird, dass diese Interpretation nicht vollständig unabhängig von materiellen Bedingungen sein kann, werden diese materiellen Bedingungen in der Forschungspraxis ausgeblendet und verschwinden damit aus dem Bereich der für die Soziologie bedeutsamen Forschungsgegenstände (z. B. David und Pinch 2008, S. 364).

Die meisten Forscher*innen, die sich trotz dieser Einwände auf konkrete Eigenschaften der Technik fokussieren, betrachten die untersuchten Artefakte nicht als Ergebnis bestimmter Verhältnisse zwischen sozialem Handeln und sozialer Ordnung im Entwicklungsprozess. Sie nehmen deren Eigenschaften entweder als faktisch außersoziale Tatsachen hin (z. B. Leonardi 2013) oder wenden sich grundsätzlich gegen den Versuch, Technisches von Sozialem zu unterscheiden. Letztere Herangehensweise stützt sich auf die Akteur-Netzwerk-Theorie (ANT), der zu Folge soziale Phänomene nicht aus einem Verhältnis von Handeln und Ordnung heraus verstanden werden können, in das technische Artefakte eingebettet sind. Vielmehr sollen sie als Effekte heterogener Netzwerke betrachtet werden, die Menschen, Dinge, Handeln, Ordnung und alles andere, was wir beobachten können, hervorbringen, (Latour

2001; Law 1992). In der ANT sind konkrete Eigenschaften von Technik Ergebnis einer *Inskription* (Akrich 2000): Entwickler*innen eines technischen Artefakts stellen sich nicht nur die Nutzenden, sondern die gesamte Situation der zukünftigen Nutzung vor und gestalten die Eigenschaften des Artefakts so, dass sie optimal in diese projizierte Zukunft passen:

> *„Designer definieren […] Akteure mit besonderem Geschmack, besonderen Kompetenzen, Motiven, Zielen, politischen Vorurteilen und vielem anderen, und sie nehmen an, dass Moral, Technik, Wissenschaft und Ökonomie sich auf bestimmte Weisen entwickeln werden." (Akrich 2006, S. 411)*

Bei der Inskription werden Handlungsmodelle, sogenannte *„Skripte"* (a.a.O.), in materielle Eigenschaften des Artefakts übersetzt. Die Skripte basieren nicht nur auf Vorstellungen von Bedürfnissen und Kompetenzen der Personen, die Technik nutzen sollen, sondern auch auf Annahmen über die Umstände, unter denen diese mit dem Artefakt umgehen. Sie beschreiben ein Netzwerk aus Menschen, Technik und Rahmenbedingungen. Verwenden die Nutzenden die Artefakte so, wie die Skripte es vorsehen, setzen sie sich selbst und die Technik in die Positionen im Netzwerk, die ihnen die Skripte zuschreiben, und tragen damit dazu bei, dass das Netzwerk, dass Entwickler*innen erdacht haben, in der Nutzungssituation entsteht (a.a.O., S. 420).

Das Konzept der Inskription wird auch in techniksoziologischen Arbeiten aufgegriffen, in denen explizit die Nutzung von Software thematisiert wird. Die Erkenntnisse aus diesen Arbeiten, die für die soziologische Auseinandersetzung mit Software relevant sind, werden in Kapitel 2 verarbeitet.

Insgesamt zeigt sich bei der Betrachtung des Forschungsstands zur Techniknutzung, dass der Versuch, sowohl die Eigenschaften technischer Artefakte als auch den Umgang mit diesen Artefakten als Ausdruck eines Verhältnisses zwischen sozialem Handeln und sozialen Ordnungen zu betrachten, selten unternommen wird. Techniksoziolog*innen berücksichtigen bei der Untersuchung dieses Verhältnisses fast ausschließlich die Aspekte sozialer Ordnung, die sich direkt auf die Technik beziehen: Leitbilder, technological frames, sozial relevante Gruppen und Vorstellungen vom Artefakt, den Nutzenden und der Nutzungssituation. Was größtenteils ausgeblendet wird, ist die Tatsache, dass Technikentwicklung und –nutzung im Rahmen von sozio-technischen *Systemen* stattfindet. Soziale Ordnung bezieht sich in diesen Systemen nicht nur auf den Umgang mit Technik, sondern auch auf Formen des Handelns, in denen Technik auf den ersten Blick keine Rolle spielt. Die grundsätzliche Bedeutung von Systemen für Technik betont z. B. Ingo Schulz-Schaeffer mit seinem Konzept der Dualität von Ressourcen und Routinen, das Technik mit

Expert*innensystemen[5] gleichsetzt (Schulz-Schaeffer 1999). Die Arbeit ist symptomatisch dafür, wie in der Techniksoziologie häufig mit dem Thema sozialer Systeme umgegangen wird: Die Bedeutung sozialer Systeme für die Funktion der Technik wird zuerst angedeutet, im Anschluss spielen diese dann keine weitere Rolle für das Konzept. Die Arbeit von Schulz-Schaeffer wird in Kapitel 2 vorgestellt; in Kapitel 3 wird die Verbindung von Technik und Expert*innensystemen ausgearbeitet.

Organisationssoziologie

Das allgemeine Verhältnis von Handeln und Ordnung in sozialen Systemen ist Kernthema der Organisationssoziologie. Anders als in der Techniksoziologie ist die Frage danach, wie Software (nicht nur Technik allgemein) in das Soziale eingebettet ist, ein Forschungsschwerpunkt in dieser Teildisziplin. In einem Großteil der Forschungsarbeiten mit diesem Schwerpunkt wird die Nutzung von Software analysiert. Eine Grundannahme in den Arbeiten, in denen Technik nicht grundsätzlich als außersozialer Gegenstand betrachtet wird, ist, dass Software das Handeln von Organisationsmitgliedern beeinflusst, aber nicht determiniert. Welche Rolle Software in Organisationen spielt, hängt demnach davon ab, ob und wie Organisationsmitglieder sich die von der Technik vorgegebenen Abläufe aneignen (Orlikowski 2000), sie umgehen (Ortmann et al. 1990, S. 77–170) oder damit unvorhergesehene neue Praktiken realisieren (Boudreau und Robey 2005). Durch die Auseinandersetzung mit Software verändern Organisationsmitglieder ihre Deutungsmuster (Fulk und Steinfield 1990) und Praktiken (DeSanctis und Poole 1994). Auf die Dauer kann diese Veränderung zu neuen Rollen und Interaktionsmustern (Barley 1990), neuen institutionellen Bedingungen des Handelns (Orlikowski 1992) und einer Verschiebung im Machtgefüge der Organisationen (Zuboff 1988) führen.

Angesichts ihrer grundlegenden Perspektive unerwartet ist, dass auch in der Organisationssoziologie verhältnismäßig selten berücksichtigt wird, welche Bedeutung Organisationen als soziale Kontexte mit eigenständigen Regeln und Dynamiken für den Umgang mit Software haben. Nur in wenigen organisationssoziologischen Arbeiten wird zum Thema, dass Organisationen eine eigene Ebene der sozialen Ordnung darstellen, die beeinflusst, wie Nutzende sich der Software nähern und welche Arten und Weisen des Umgang sie entwickeln können (Ortmann et al. 1990; Orlikowski 1992). Auch die Tatsache, dass die Umgangsformen mit Software nicht

[5] Unter „Expertensystemen“ (der Begriff wird in der vorliegenden Arbeit nachträglich gegendert) versteht Schulz-Schaeffer dabei nicht die von Werner Rammert et al (1998) untersuchte Form der Software, sondern, in Anlehnung an Anthony Giddens „expert systems“ (Giddens 2010 [1995]), spezielle Formen der systematischen Zusammenstellung von Expert*innenwissen, durch die gesicherte Ereigniszusammenhänge hergestellt werden. Das Konzept wird in Kapitel 3 genauer vorgestellt.

nur von der Organisation, sondern darüber hinaus auch von den konkreten Eigenschaften der Software abhängen, die die Akteure sich aneignen, wird nur selten untersucht. Wo dies geschieht (D'Adderio 2008; Volkoff et al. 2007) und die Spezifika der Software in die Analyse der Entwicklung von Nutzungsweisen einbezogen werden, erscheint die Software als quasi-neutrales Übertragungsmedium von sozialen Strukturen, also von Ordnungsmustern, die intentional von Entwickler*innen eingebaut worden sind. Als solches füge sie die Strukturen in die Nutzungskontexte ein, wo sie dann Handlungsoptionen beschränken (DeSanctis und Poole 1994).

Eine solch vereinfachende Perspektive auf Software bleibt nicht nur hinter dem Stand der Forschung in der Techniksoziologie zurück, sondern unterschlägt auch, dass Entwickler*innen Software nicht vollständig beliebig gestalten können. Obwohl Code häufig als ein Material betrachtet wird, das einfacher zu manipulieren ist als jeder physische Stoff, müssen Akteur*innen sich bei der Entwicklung von Software ebenso an Regeln orientieren, die ihre Fähigkeiten, ihr Material zu manipulieren, einschränkt, wie bei der Entwicklung jeder anderen Technik. Code ist kein Stahl, aber er besitzt Besonderheiten, die bei der Softwareentwicklung ebenso berücksichtigt werden müssen wie die physischen Gesetze, nach denen sich Maschinenbauer*innen bei ihrer Arbeit richten müssen. Diese Besonderheiten sind schwerer zu fassen, im wörtlichen Sinne: Was die „Materialität" von Software ausmacht und wie diese in empirischen Analysen gefasst werden kann, ist in der Soziologie umstritten. Wenig überraschend konzentrieren sich daher auch Studien in der Organisationssoziologie auf die Akteur*innen und ihren sozialen Kontext und blenden die technischen Eigenschaften der Software aus. Die Forderung, dies zu ändern und die „Materialität" von digitaler Technik auch empirisch zu berücksichtigen, wird seit langem regelmäßig erhoben (Monteiro und Hanseth 1996; Orlikowski und Iacono 2001; Orlikowski 2005; Leonardi und Barley 2010), in der Praxis aber nur selten umgesetzt (aber z. B. Mormann 2016).

Eine Ursache dafür liegt in der Konzentration der Arbeiten auf die Einführungs- und Nutzungsphase (Leonardi und Barley 2008, S. 166 f). Dadurch erscheint die Software zu Beginn der Untersuchung primär als Objekt mit statischen technischen Eigenschaften, die erst durch die Interpretationen und Interaktionen der Nutzenden sozial bedeutsam und für Soziolog*innen zugänglich werden (Leonardi und Barley 2008; Leonardi 2009). Um zu prüfen, ob und wie die konkrete technische Ausgestaltung der Software für die Art der Interpretationen und Interaktionen relevant ist, müsste sie als Produkt sozialer Prozesse in die Analysen der Nutzung einbezogen werden. In dieser Hinsicht treffen Organisationssoziolog*innen bei der Untersuchung von Software auf die gleiche Problematik wie Techniksoziolog*innen, auch wenn sie sich dieser vorzugsweise aus der Richtung der Techniknutzung nähern: Entwicklungs- und Nutzungsprozesse sind aufeinander bezogen, lassen sich jedoch

aufgrund der Bedingungen der Forschungspraxis (v. a. der zeitlichen und räumlichen Begrenzung von Forschungsprojekten) nur schwer gemeinsam untersuchen.

Dass dies vor allem ein Problem der Forschungspraxis, nicht eines der Theorie ist, zeigt Wanda Orlikowski mit dem Konzept der Dualität von Technik (Orlikowski 1992), das sie in den frühen 1990er Jahren in der Auseinandersetzung mit Organisationssoftware entwickelt hat. Das Konzept basiert auf der Erkenntnis, dass Formen des Umgangs mit (Informations-)Technik nur verstanden werden können, wenn man die Akteure, die mit Software umgehen, die konkreten Eigenschaften der Technik und die „*social and historical circumstances*" (Orlikowski 1992, S. 411), unter denen Technik entwickelt und genutzt wird, gemeinsam berücksichtigt. Die Arbeit, in der das Konzept der Dualität von Technik entwickelt wird, stellt zusammen mit der konzeptionell darauf aufgebauten Studie zur Bedeutung von Nutzungspraktiken (Orlikowski 2000) immer noch eine der meist zitierten Arbeiten im organisationssoziologischen Diskurs um Software dar. Eindringlich werden in diesem Konzept die drei zentralen Aspekte benannt, die betrachtet werden müssen, wenn das Zusammenspiel von Software und Sozialem verstanden werden soll: die Akteur*innen, die mit Software umgehen, die konkreten technischen Eigenschaften der Software und die sozialen Bedingungen, unter denen dieser Umgang stattfindet. Obwohl diese zentralen Aspekte im Konzept klar als solche benannt werden, hat die Prominenz des Konzept der Dualität von Technik bisher nicht dazu geführt, dass die Trias aus Akteur*innen, konkreten technischen Eigenschaften und Bedingungen des Umgangs mit Software bei der soziologischen Untersuchung von Softwareentwicklung und –nutzung regelmäßig in ihrem Zusammenspiel betrachtet worden wäre. Die vorliegende Arbeit knüpft hier an. Das Konzept der Dualität von Technik wird in Kapitel 2 genauer vorgestellt. Es liegt auch der Ausarbeitung des Forschungsrahmens in Kapitel 3 zu Grunde.

Computer Supported Cooperative Work

Die naheliegende Verbindung von technik- und organisationssoziologischen Erkenntnissen versuchen Wissenschaftler*innen im Forschungsfeld der Computer Supported Cooperative Work (CSCW) zu leisten. In diesem Feld wird untersucht, wie Softwareentwicklungsprozesse organisiert sind (Rönkkö et al. 2005) und wie sie sich diese Organisation auf Entscheidungen auch über technische Fragen im Prozess der Softwareentwicklung auswirkt (Button und Sharrock 1998). CSCW-Forscher*innen betrachten, wie Software durch die Einbindung zukünftiger Nutzer*innen in den Entwicklungsprozess auf spezifische Weise geformt wird (Egger und Wagner 1993) und welche Faktoren die erfolgreiche Zusammenarbeit zwischen Entwickler*innen und potentiellen Nutzer*innen systematisch

erschweren (Petersen 2004; Star und Ruhleder 1996). Die große Stärke der CSCW-Forschung für die Frage nach der Rolle von Software im Sozialen liegt darin, dass hier der Fokus auf der Beziehung zwischen Entwicklungs- und Nutzungsphase liegt und technik-, organisations- und auch arbeitssoziologische Ideen zusammengeführt werden, um diese Beziehungen besser zu verstehen. Mehr als in den anderen hier erwähnten Bereichen finden sich in der Forschung zu CSCW auch Anknüpfungspunkte für Informatiker*innen, die ihren Gegenstand als eine besondere und sozial bedeutsame Technologie betrachten. Die Frage, wie konkrete Eigenschaften von Software gestaltet werden können und welche Folgen sie für die Nutzung haben, steht für diese Forscher*innen im Vordergrund (Schmidt und Bannon 2013).

Forscher*innen in den CSCW nehmen das Verhältnis von sozialem Handeln und sozialer Ordnung vor allem dadurch auf, dass sie die Einbettung von Software in Praktiken fokussieren. Dieser Fokus lässt sich damit begründen, dass sich Praktiken konkret beobachten und rekonstruieren lassen (Schmidt 2018; Pipek und Wulf 2009; Wulf et al. 2018). Als Praktiken werden dabei sich wiederholende, normativ geleitete Aktivitäten verstanden, die Regeln (eine „Theorie“) besitzen und von Akteur*innen unter kontingenten Bedingungen ausgeführt werden. Wenn Akteur*innen mit Software umgehen (in der „Praxis“), berücksichtigen sie im Handeln sowohl die abstrakten Regeln als auch die konkreten Bedingungen, unter denen sie aktuell handeln (Schmidt 2018). Da soziologische Forschung und anwendungsorientierte Softwareentwicklung in der CSCW oft im gleichen Projekt stattfinden, wird in Arbeiten aus diesem Forschungsfeld besonders sichtbar, dass das Verhältnis von Software und Sozialem rekursiv ist: Einmal eingeführt, verändert Software (meist) sowohl die „Theorie“ der Praktiken, die Vorlage für ihre Entwicklung waren, als auch (immer) die Bedingungen, unter denen sie genutzt wird (a.a.O., S. 53). Die CSCW-Forschung zeigt, dass es möglich ist, einerseits zu berücksichtigen, dass Software sowohl bei der Entwicklung als auch bei der Nutzung in das Verhältnis von sozialer Ordnung und sozialem Handeln eingebettet ist, anderseits aber auch zu untersuchen, welche konkreten Eigenschaften Software hat und wie diese Eigenschaften die Einbettung beeinflussen.

Zwar füllt die CSCW-Forschung durch die Konzentration auf den Zusammenhang zwischen Entwicklungs- und Nutzungsprozessen eine Lücke, die sich zwischen Technik- und Organisationssoziologie auftut, doch sie besitzt auch einen großen blinden Fleck: In aller Regel werden nur kleine Entwicklungsprojekte untersucht, in denen Software für einen Kreis wohl bekannter Nutzer*innen und meist mit ihrer Beteiligung gestaltet wird. Dies führt konzeptionell zu einer Überbetonung der lokalen Besonderheiten des untersuchten Einzelfalls (Monteiro et al. 2013). Dadurch gelingt es zwar, ein tiefes Verständnis über das Zusammenspiel von Software und Sozialem im jeweiligen Fall zu entwickeln, es wird jedoch kaum thematisiert, welche

Aspekte dieses Zusammenspiels verallgemeinerbar und welche auf die besondere Entwicklungssituation zurückzuführen sind. Die Erkenntnisse aus solchen Arbeiten lassen sich nur begrenzt auf die aktuell und (absehbar) zukünftig dominanten Formen von Software übertragen. Diese dominanten Formen von Software werden zum einen stark von Standardisierung beeinflusst, sei es durch die Verwendung standardisierter Einzelteile oder durch die Orientierung an gemeinsamen Protokollen und Schnittstellen (mehr dazu in 2.1.3). Zum anderen ist Software, die speziell für eine kleine Gruppe bekannter Personen (oder die Mitglieder einer konkreten Organisation) entwickelt wird, immer stärker durch Standardsoftware verdrängt worden (Sawyer 2001). Diese Verschiebung ist in der soziologisch orientierten Auseinandersetzung mit Software, die ihre Wurzeln in den 1980er Jahren hat, als maßgeschneiderte Software dominant war, noch nicht ausreichend nachvollzogen worden.

Standardsoftware wird für noch unbekannte Gruppen zukünftiger Nutzer*innen konstruiert, deren Praktiken und Rahmenbedingungen nicht beobachtet werden können, sondern im Entwicklungsprozess vorhergesagt werden müssen. Praktiken und Rahmenbedingungen des Umgangs mit Software und die Software selbst müssen dann bei der Einführung aneinander angepasst werden. Neben der Konzentration auf Nicht-Standardsoftware bleibt auch in den CSCW die Tatsache unterbelichtet, dass die Nutzung von Software in sozialen Systemen stattfindet, deren Ordnung den Umgang mit der Technik ebenso beeinflusst wie die konkreten Eigenschaften der Artefakte. Mit der seit den 1990er Jahren anwachsenden Forschung zu Infrastrukturen (Star und Ruhleder 1996; Hanseth und Monteiro 1997; Pollock et al. 2007; Monteiro et al. 2013) soll diese Schwäche ausgeglichen werden. Diese Forschung leidet aber an einer ähnlichen konzeptionellen Unschärfe des zentralen Gegenstands wie die Forschung zum „Digitalen“ (Lee und Schmidt 2018). In Kapitel 2 werden einige zentrale Erkenntnisse der Infrastrukturforschung aufgegriffen und gezeigt, dass damit allgemeine Besonderheiten von Software beschrieben werden.

Software Studies

Während in der Technik- und Organisationssoziologie sowie der CSCW konzeptionell zwischen Software und Sozialem unterschieden wird, um die Wechselbeziehungen zwischen beidem zu betrachten, wird diese Unterscheidung in den Software Studies explizit abgelehnt. Adrian Mackenzie schlägt vor, Software nicht als Technik zu betrachten, sondern als *„eine Menge an Möglichkeiten, Handlungsfähigkeit*[6] *zwischen Menschen, Maschinen und in Code festgehaltenen symbolischen Umwelten zu verteilen“* (Mackenzie 2006, S. 19, eigene Übersetzung). Software wird

[6] Der im Original verwendete Begriff der „agency“ wird weiter unten diskutiert.

aus dieser Perspektive nach sozial relevant, weil ihr Code mögliche Beziehungen zwischen Produzent*innen, Empfänger*innen und repräsentierten Phänomenen („Protoypen", a.a.O., S. 16) beschreibt und der Umgang mit Software solche Beziehungen auf eine Art und Weise verändert, die mit der Beschreibung im Code zusammenhängt. Wie in den CSCW, aber mit größerem Anspruch auf Verallgemeinerung, wird in den Software Studies das rekursive Aufeinander-Bezogen-Sein von Prozessen der Softwareentwicklung und Prozessen der Softwarenutzung ins Zentrum der Theorie gestellt. Der Fokus liegt jedoch nicht auf Praktiken, sondern auf Code. Er wird als Produkt hochgradig organisierter, durch Macht, Deutungsmuster und Formen der Bewertung beeinflusster sozialer Prozesse verstanden, deren Hauptziel die Veränderung des Sozialen ist (a.a.O., S. 173 ff). Die Beziehungen, in denen diese Prozesse stattfinden, und die für sie charakteristische (ungleiche) Verteilung von Handlungsmöglichkeiten, werden durch die Produktion des Codes in die Software „eingefaltet" (a.a.O., S. 181 f)[7]. Dieses „Einfalten" darf aber nicht als irgendwie statische Form der Ablagerung verstanden werden, da Software für ihre Ausführung auf ganz bestimmte technische und soziale Bedingungen angewiesen ist, die sich ständig verändern und an die auch der Code ständig angepasst werden muss (a.a.O., S. 12). Der Code regt bestimmte Beziehungskonfigurationen beim Umgang mit Software an[8], indem er Nutzende, Beziehungen und Formen sozialer Prozesse in abstrakter Form beschreibt (a.a.O., S. 15). Zwischen dem, was der Code beschreibt, und dem, was bei der Ausführung von Software geschieht, klafft jedoch immer eine Lücke, da der Umgang mit Software auch durch die Beziehungskonfigurationen bei der Ausführung beeinflusst wird (a.a.O., S. 177 ff). Die Grenze zwischen Softwareentwicklung und Softwarenutzung ist darüber hinaus nicht allgemein bestimmbar, da Code von Programmierer*innen nicht nur hervorgebracht, sondern auch selbst genutzt wird. Dies führt dazu, dass auch Softwareentwicklung von den Beziehungskonfigurationen beeinflusst ist, die in den Code „eingefaltet" sind. Dadurch kann

[7] Der von Mackenzie verwendete Begriff ist „involution", den er auf ein anthropologisches Konzept von Geertz zurückführt. „Involution" meint, dass eine Menge unterschiedlicher Beziehungen miteinander verknüpft und zunehmend aufeinander bezogen wird, sich dadurch insgesamt stärker nach innen (in die Menge) wendet und von außen weniger durchschaubar, sozusagen „dichter", aber gleichzeitig auch leistungsfähiger wird (Mackenzie 2006, S. 181 f).

[8] Auch in diesem Fall muss meine Übersetzung erläutert werden: Der im Original verwendete Begriff „solicit" bezeichnet sowohl (eindringliches) Werben als auch (dringliches) Erbitten oder (hartnäckiges) Nachfragen, wobei der Aspekt der Hartnäckigkeit je nach Verwendungskontext stark im Vordergrund steht oder überhaupt nicht relevant ist (Merriam-Webster 2019).

Software insgesamt niemals eindeutig einer Beziehungskonfiguration zugeschrieben werden, denn jeder Umgang mit Software ist sowohl von aktuellen als auch von früheren Beziehungen und darin relevanten Formen der Ordnung abhängig:

> *„Even if a programmer is working on his or her own, the program is implicitly the result of a collective endeavor – the programmer uses a formalized coding language, proprietary coding packages (and their inherent facilities and defaults), and employs established disciplinary regimes of programming – ways of knowing and doing regarding coding practices and styles, annotation, elegance, robustness, extendibility, and so on" (Kitchin und Dodge 2011, S. 33)*

Vertreter*innen der Software Studies konzentrieren sich durch diesen Ansatz konsequenter als alle anderen hier vorgestellten Debatten auf die Einbettung von Software in das Verhältnis von sozialem Handeln und sozialer Ordnung. Ich werde mich bei der Argumentation in dieser Arbeit daher stark an der hier vorgestellten Perspektive auf Software orientieren. Die soziologische Auseinandersetzung mit Software kann sich jedoch nicht allein auf diese Perspektive stützen. Dies liegt daran, dass die zentrale Fragestellung in den Software Studies die nach der Verteilung von „Agency" durch Software ist.

Als „Problem der Agency" bezeichnet Mackenzie die Frage *„who or what does what to whom or what"* (Mackenzie 2006, S. 7) und schließt damit an die Grundannahme der ANT an, dass Gesellschaft nicht aus dem aufeinander bezogenen Handeln von Menschen entsteht, die unter Bedingungen handeln, die sie und andere zu einem Großteil selbst hervorgebracht haben, sondern aus Effekten, die durch die Verknüpfung menschlicher und nicht-menschlicher Einheiten produziert werden, die a priori ununterscheidbar sind (Law 1992). „Agency", im allgemeinen (auch in dieser Arbeit weiter oben) als Handlungsfähigkeit übersetzt, steht damit nicht in vergleichbarer Beziehung zu „sozialem Handeln", wie in der klassischen Soziologie, denn *„Handlungsfähigkeit wird nicht allein menschlichen Personen, sondern auch naturalen und technischen Gegenständen, pflanzlichen und tierischen Lebewesen zugesprochen"* (Kneer 2009, S. 20). Bei der Untersuchung von Software wird auf Basis dieses Konzepts von Agency nicht gefragt, was Menschen warum, wie und unter welchen Bedingungen mit Software tun können, sondern vor allem, was Software oder Code tut und wie beide dazu beitragen, dass auch *„machines, technical systems and infrastructures"* sozial Relevantes tun (Kitchin und Dodge 2011, S. 39). Diese Verschiebung des Fokus von menschlicher Handlungsfähigkeit zum Ursprung von beobachtbaren Effekten hat es den Software Studies ermöglicht, die hochgradig komplexen und rekursiven Beziehungen zwischen Entwicklungs- und Nutzungsprozessen, die die Rolle von Software im Sozialen auszeichnen, besser zu

durchdringen, als dies Anhänger*innen klassisch soziologischer Perspektiven bisher gelungen ist (Rose et al. 2005). Forscher*innen, die die Perspektive der Software Studies einnehmen, müssen im Austausch dafür jedoch darauf verzichten, menschliches Handeln und die spezifische Art und Weise, wie dieses sozial eingebettet ist, konzeptionell von einem technischen Wirken zu unterscheiden.

Ausrichtung der Arbeit

Statt die primär theoretische Kritik von Soziolog*innen an diesem Verständnis von Agency nachzuzeichnen (Kneer 2009), greife ich die Erkenntnisse der Software Studies in dieser Arbeit auf eine Weise auf, die mit der gesellschaftlichen Aufgabe vereinbar ist, die C.W. Mills für die Soziologie formuliert hat. *„Social science deals with problems of biography, of history, and of their intersections within social structures"* (Mills 2000 [1959], S. 143). Die Soziologie untersucht dazu, auf welche Weise die Schwierigkeiten der Individuen („private troubles") und die Probleme der Allgemeinheit („public issues") verbunden sind (a.a.O., S. 8) und weist auf die Handlungsmöglichkeiten hin, die Individuen haben, wenn sie Schwierigkeiten und Probleme lösen wollen (a.a.O., S. 20). Ich teile Mills Überzeugung, dass Verstehen in der Soziologie kein Selbstzweck ist, sondern letztendlich dazu dienen sollte, Menschen ihre Handlungsmöglichkeiten aufzuzeigen. Menschliche Handlungsfähigkeit, nicht der Ursprung beobachteter Effekte, spielt für dieses Verstehen die Hauptrolle:

> *„We study the structural limits of human decision in an attempt to find points of effective intervention, in order to know what can and what must be structurally changed if the role of explicit decision in history-making is to be enlarged. […] We study history to discern the alternatives within which human reason ad human freedom can now make history. […]. Freedom is, first of all, the chance to formulate the available choices, to argue over them—and then, the opportunity to choose. […] Within an individual's biography and within a society's history, the social task of reason is to formulate choices, to enlarge the scope of human decisions in the making of history. […] The future is what is to be decided— within the limits […] of historical possibility." (Mills 2000 [1959], S. 174)*

Ziel soziologischer Forschung ist es herauszufinden, welche sozialen Bedingungen die Entscheidungsfreiheit der Menschen beschränken und was Menschen unter den gegebenen Bedingungen tun können, um ihre Entscheidungsspielräume gezielt auszuweiten. Was Mills mit *„history"* und *„structural limits of human decision"* bezeichnet, ist das, was in dieser Arbeit „soziale Ordnung" genannt wird: Die immer spezifischen sozialen Bedingungen, die beeinflussen, wer in welcher Situation was tun kann und wem welche dieser Handlungsmöglichkeiten bekannt sind. Diese Handlungsmöglichkeiten, die *„points of effective intervention",* sind einerseits

„historisch“ begrenzt, also für alle Zeitgenoss*innen gleichermaßen, andererseits „biographisch“, also für verschiedene Menschen zum gleichen Zeitpunkt auf verschiedene Weise. Aufgabe der Soziologie ist es, herauszuarbeiten, wo diese Grenzen für wen in welchen Situationen verlaufen, welche Handlungsoptionen sich für wen ergeben und welche Konsequenzen für die soziale Ordnung zu erwarten sind, wenn diese oder jene Option gewählt wird. Auf diese Weise können Handlungsoptionen zum Gegenstand bewusster Entscheidung werden. Soziologische Forschung ist also Mittel zum Zweck: sie soll diejenigen, die untersucht werden, über Bedingungen, Folgen und Grenzen ihres eigenen Tuns aufklären.

Folgt man diesem Verständnis von Soziologie, produziert eine Soziologie der Software Erkenntnisse über das Verhältnis von sozialem Handeln und sozialer Ordnung und über die Rolle, die Software darin spielt, um durch diese Erkenntnisse Menschen in die Lage zu versetzen, informierte Entscheidungen darüber zu treffen, wie sie und andere mit Software umgehen sollten.[9] Um solche Erkenntnisse zu erlangen ist es notwendig, dass zwischen dem, was Menschen tun, und dem, was Software (Code, Hardware etc.) tut, jederzeit unterschieden werden kann; nicht, weil die Gleichsetzung philosophisches Unbehagen verursacht, sondern weil diese Forschung sich ausschließlich an Menschen richtet. *Deren* informierte Entscheidungen soll soziologische Forschung ermöglichen. Das ANT-Konzept von Agency mag mit Blick auf diese Aufgabenstellung hilfreich sein, um initial zu verstehen, wie komplexe Technik menschliches Entscheiden in konkreten Situationen beeinflusst; es verdeckt jedoch die Tatsache, dass technisch verursachte Effekte aus menschlichem Handeln und damit auch aus Entscheidungen folgen, die durch soziologische Forschung informiert werden können. In dieser Arbeit setze ich daher voraus, dass menschliche Handlungsfähigkeit nicht allein die Fähigkeit ist, Effekte zu produzieren, sondern immer ein Moment der Handlungsfreiheit beinhaltet. Handlungsfreiheit meint, dass Menschen prinzipiell in der Lage sind, im Rahmen der Bedingungen, die sie vorfinden, (bewusst oder unbewusst) Handlungsalternativen zu entdecken und auszuwählen, und dass ihre Wahl zwar von diesen Bedingungen abhängt, aber nie vollständig von ihnen determiniert wird. Menschliche Handlungsfähigkeit ist damit prinzipiell mit Unsicherheit verbunden, weil Menschen immer auch anders handeln können (Giddens 2008 [1984]). Wie die vorangegangenen Ausführungen zur Aufgabe der Soziologie gezeigt haben, gehe ich weiterhin davon aus,

[9] Wenn im weiteren Verlauf der Arbeit von „Soziologie“ oder einer „soziologischen Perspektive“ gesprochen wird, so ist damit immer dieses Verständnis gemeint.

dass Menschen die Bedingungen ihres Handelns durch das, was sie tun, mitgestalten und verändern können.[10]

Forschungsfrage und Vorgehen

Die Frage, die ich zu Beginn der Einleitung gestellt habe, war, wie Software im Sozialen relevant wird. In dieser Arbeit will ich diese Frage nicht theoretisch beantworten, sondern eine Antwort finden, die Forschenden weiterhilft, wenn sie die Rolle von Software auf eine im obigen Sinne soziologische Art und Weise untersuchen wollen. Meine Forschungsfrage lautet daher:

> Wie wird Software im Sozialen relevant und wie lässt sich untersuchen, welche Rolle Software für soziale Phänomene spielt?

Die Forschungsfrage wird in zwei Schritten bearbeitet. Zunächst werden die Grundlinien einer Soziologie der Software skizziert, indem bestehende Forschungsergebnisse zum Thema aufgegriffen und mit Blick auf den ersten Teil der Forschungsfrage integriert werden (Kapitel 2). Im Anschluss wird aufbauend darauf vorgestellt, wie diese Grundlinien in einen Forschungsrahmen überführt werden können, mit dem konkrete Forschungsfragen an empirischen Gegenständen soziologisch untersucht werden können (Kapitel 3). Die Anwendung eines solchen Forschungsrahmens wird dann anhand einer Fallstudie demonstriert (Kapitel 4).

Mein Hauptkritikpunkt an den bestehenden soziologischen Debatten zum Thema ist, dass diese Softwareentwicklung und –nutzung zu stark isoliert betrachten und damit die zentrale Besonderheit von Software ignorieren, die in den Software Studies so deutlich herausgearbeitet wird: Über Software sind die Bedingungen aller sozialen Kontexte, die das Artefakt im Laufe seiner Existenz durchläuft, aufeinander bezogen. Für das Verständnis der heute gebräuchlichen Formen von Software ist dieser Aspekt besonders relevant, da diese stark auf Wiederverwendung und permanenter Weiterentwicklung basieren und so deutlich mehr soziale Kontexte durchlaufen, als mit der isolierenden Betrachtung sichtbar werden.

Um dieses Argument nachvollziehen zu können, müssen Soziolog*innen verstehen, wie Software funktioniert. In Kapitel 2 rekonstruiere ich daher zuerst, wie

[10] Ich glaube, dass beide Annahmen grundsätzlich von der Mehrzahl moderner soziologischer Theorien geteilt werden und theoretische Debatten sich mehr um die jeweilige Relevanz von Handeln und Ordnung sowie Art und Ausmaß der gegenseitigen Abhängigkeit drehen als um die grundsätzliche Frage, ob Handlungsfreiheit oder soziale Ordnung existieren (vgl. Giddens et al. 1999, S. 614–617).

Software aufgebaut ist, in welchen Prozessen sie wie von welchen Akteuren entwickelt und genutzt wird und was die Grundlagen ihrer Funktionen sind. Dabei wird dargestellt, welche soziologischen Fragen die Prinzipien und Prozesse aufwerfen, die für den Umgang mit Software relevant sind. Die allgemeinen Ausführungen zu Software werden dabei um Einschübe ergänzt, in denen ich mich vertiefend mit den Besonderheiten standardisierter Organisationssoftware auseinandersetze. Diese Form der Software wird später in Kapitel 4 Gegenstand der empirischen Untersuchung sein. Im Anschluss an die Darstellung des Gegenstands Software gebe ich einen Überblick darüber, welche der im ersten Teil des Kapitels aufgeworfenen Fragen in den bestehenden soziologischen Diskursen zu Software aufgegriffen und wie sie beantwortet werden. Die Erkenntnisse dieser Arbeiten werden zum Abschluss des Kapitels so integriert, dass die Frage, wie Software im Sozialen relevant wird, abstrakt beantwortet werden kann.

Kapitel 3 ist der Frage gewidmet, wie dieses abstrakte Konzept für konkrete Forschungsvorhaben fruchtbar gemacht werden kann. Dazu werden zuerst das Konzept der Expert*innensysteme und die Forschungsperspektive der strategischen Organisationsanalyse zusammengeführt, um auszuarbeiten, auf welche Art von sozialen Ordnungen Software verweist und wie Software die sozialen Ordnungen ergänzt, die das soziale Handeln beim Umgang mit Software beeinflussen. Darauf aufbauend wird beschrieben, wie Forschende einen Analyserahmen entwickeln können, der an konkrete Forschungsfragen und Untersuchungsgegenstände angepasst werden kann. Ich demonstriere die Vorgehensweise durch die Entwicklung eines eigenen Forschungsrahmens, den ich nutze, um zu untersuchen, welche Rolle das standardisierte Krankenhausinformationssystem eMed für die Praxis der OP-Planung in einer großen deutschen Universitätsklinik spielt. Die empirische Untersuchung und die Analyse der Fallstudie folgen in Kapitel 4. Zum Abschluss der Arbeit werde ich in Kapitel 5 rekapitulieren, wie die Zusammenführung der bestehenden Diskurse zu einem Verständnis der Rolle von Software im Sozialen beiträgt, und die Grenzen des von mir skizzierten Vorgehens diskutieren.

2 Die Komplexität von Software aus soziologischer Perspektive: Modelle, Unsicherheiten und Rekursivität

„Größere Software ist in der Regel nicht ein wenig, sondern immer extrem komplex. Merkwürdigerweise ist sich aber fast niemand dieser Komplexität bewusst, weshalb die Erstellung und die Wirkung von Software ständig unterschätzt wird. […] Die Undurchschaubarkeit der Systeme macht Fehler prinzipiell unvermeidlich." (Claus und Schwill 2003, S. 603 f.)

In diesem Kapitel nutze ich die Komplexität von Software, die in diesem Zitat aus dem „Duden Informatik" thematisiert wird, als Ausgangspunkt, um die Probleme zu skizzieren, die im Kern einer Soziologie der Software stehen, wie sie in der vorliegenden Arbeit verstanden wird. Zu diesem Zweck lege ich erstens die Grundprinzipien dar, auf denen Software basiert. Ich stelle zweitens verallgemeinert vor, wie die sozialen Prozesse ablaufen, in denen Software entwickelt und genutzt wird, welche Akteur*innengruppen daran beteiligt sind und wie diese Prozesse durch die sozialen Systeme, also Organisationen, Projekte o.ä., beeinflusst werden, in denen sie stattfinden. Drittens zeige ich, welche Wechselwirkungen und Abhängigkeiten durch Software zwischen diesen Prozessen geschaffen werden und erläutere, wie diese Wechselwirkungen aus den Grundprinzipien folgen, auf denen Software basiert. Dieser dritte Punkt ist in der soziologischen Forschung zu Software bisher nur selten behandelt worden. Er wird in meinem Beitrag zur Soziologie der Software im Mittelpunkt stehen.

Über Software wird die Vieldeutigkeit, die soziale Phänomene aus konstruktivistischer Perspektive charakterisiert, mit der eindeutigen Funktionslogik vermittelt, nach der Mikroprozessoren arbeiten. Diese Vermittlung ist aufwändig und gelingt nur, weil Softwareentwickler*innen und andere, die sich an der Softwareentwicklung beteiligen, dabei eine Vielzahl an Einzelentscheidungen treffen:

D. Ametowobla, *Zur Soziologie der Software*,
https://doi.org/10.1007/978-3-658-37256-9_2

Wozu soll die Software dienen? Wer soll sie unter welchen Bedingungen nutzen? Welche Funktionen soll sie haben? Wie sollen diese Funktionen erzeugt werden? Was darf die Entwicklung kosten? Wer soll sie durchführen? Wann soll sie abgeschlossen sein? etc. Auch wenn Software fertig programmiert ist, müssen Entscheidungen getroffen werden, bevor und während sie genutzt werden kann: Welche Software soll für die Nutzung beschafft werden? Wer darf oder muss sie wozu nutzen? Welche Daten sind nötig, damit sie funktionieren kann? Wie muss mit ihr umgegangen werden? etc. Wer mit einer soziologischen Perspektive auf Software blickt, dem springt ins Auge, dass die Umstände, unter denen diese Einzelentscheidungen getroffen werden, und die Prozesse, in denen sie getroffen werden, Themen für soziologische Untersuchungen sind. Im Folgenden gehe ich von der Annahme aus, dass Software technisch und sozial komplex ist. Ich stelle diese Komplexität in drei Dimensionen vor und diskutiere dabei jeweils, wie diese Dimensionen aus Sicht von Akteur*innen erscheinen, die sich vor allem für die technische Seite der Software interessieren, und welche Fragen diese Dimensionen für diejenigen aufwerfen, die an den sozialen Aspekten dieser Komplexität interessiert sind. Ich zeige, wie einzelne dieser Fragen in der bestehenden soziologischen Forschung zu Software aufgegriffen werden. Ich schließe die Skizze einer soziologischen Perspektive auf die Rolle von Software im Sozialen an, in der diese Fragen gemeinsam berücksichtigt werden können.

Software entsteht aus einer Vielzahl an Einzelentscheidungen, die in Wechselwirkung miteinander stehen und von denen jede durch das Verhältnis von sozialer Ordnung und sozialem Handeln im Entscheidungsprozess beeinflusst wird. Diese Tatsache hat Konsequenzen, die ich hier vereinfacht unter dem Oberbegriff „Komplexität“ zusammenfasse. Die Komplexität von Software und die Folgen, die diese Komplexität für den Umgang mit Software hat, bilden den Rahmen, in dem ich meinen Forschungsgegenstand in diesem Kapitel beschreibe. Als Umgang mit Software wird dabei jede Art des Handelns verstanden, bei der Software eine Rolle spielt; insbesondere ist damit nicht nur die Nutzung gemeint, sondern auch das, was Softwareentwickler*innen tun, wenn sie Software herstellen.

Nach Arthur sind alle Technologien durch drei grundlegende Prinzipien charakterisiert: Sie basieren auf Phänomenen, bestehen aus einer Kombination von Einzelteilen, und sind rekursiv strukturiert (Arthur 2011).[1] In Anlehnung an diese

[1] Technologien entstehen nach Arthur durch die Kombination von Einzelteilen, um einen Zweck zu erfüllen, den Menschen gesetzt haben (Arthur 2011, S. 32 f.). Der Zweck gibt die Struktur der Technologie vor, die Einzelteile erfüllen Teilaufgaben, die zur Erfüllung des Zwecks notwendig sind oder diese unterstützen. Ihre Struktur ist rekursiv (a.a.O., S. 38 f.): Jedes der Einzelteile ist eine Technologie, da es zur Erfüllung eines eigenen Zwecks dient, der wiederum dessen Struktur vorgibt. Die Grundelemente dieser rekursiven Struktur sind

Prinzipien identifiziere ich drei Dimensionen der Komplexität von Software, die ich im Folgenden nutzen werde, um die Fragen, die Software für die Soziologie aufwirft, aufzuspannen. Die Dimensionen lassen sich auf die Art und Weise zurückführen, in der sich die technologischen Prinzipien im Fall von Software ausprägen. Die erste Dimension ist aus dem Phänomen abgeleitet, das durch Software ausgenutzt wird: Software basiert auf der Nutzung von Mikroprozessoren, die diskrete und eindeutige Eingaben auf deterministische Weise verarbeiten, bildet aber kontinuierliche und mehrdeutige Prozesse ab, die vor der Verarbeitung so übersetzt werden müssen, dass die Prozessoren damit etwas anfangen können. Die zweite Dimension entsteht aus dem Konstruktionsprozess: Software entsteht aus einer Vielzahl einzelner Modelle, die miteinander verbunden werden, um gemeinsam einen übergeordneten Zweck zu erfüllen. Die dritte Dimension lässt sich auf die Rekursivität der Konstruktionsprozesse zurückführen: Die Elemente, die bei der Softwareentwicklung verbunden werden, basieren alle auf dem gleichen Phänomen und entstehen in ähnlich strukturierten Konstruktionsprozessen, wodurch sich die Komplexität der Software insgesamt über Wechselwirkungen steigern kann.

Im Umgang mit Software stehen die Akteure vor einer Reihe typischer Handlungsprobleme, die sich ebenfalls diesen Dimensionen zuordnen lassen: Wie lässt sich eine unbekannte Anzahl mehrdeutiger Gegenstände in eindeutige Anweisungen für eine deterministische Verarbeitung übersetzen? Welche Einzelteile ergeben zusammen das gewünschte Ganze, woher kann man sie bekommen und wie müssen sie kombiniert werden, um den übergeordneten Zweck zu erfüllen? Und wie ist mit den Unsicherheiten umzugehen, welche die Einzelteile und ihre Kombination mit sich bringen? Im Rahmen dieses Kapitels werden die hier sehr allgemein bestimmten typischen Handlungsprobleme und etablierte Strategien[2] zum Umgang mit ihnen ausgearbeitet. Diese liegen der sozialen Komplexität von Software zugrunde. Es wird gezeigt, wie und in welchem Ausmaß diese Handlungsprobleme und Strategien bisher in der Soziologie thematisiert werden und welche soziologisch relevanten Fragen dabei nicht berücksichtigt werden. Am

sogenannte Phänomene, d. h. natürlich auftretende Effekte, die nutzbar gemacht werden, um einen Zweck zu erfüllen (a.a.O., S. 46).

[2] Zu etablierten Strategien zum Umgang mit den Handlungsproblemen bei der Entwicklung, Einführung und Nutzung von Software zählen Best Practices und Werkzeuge der Softwareentwicklung, die Orientierung an technischen und gesetzlichen Standards oder die Entwicklung spezieller Formen der (temporären) Organisation ebenso wie die rein zeremonielle Beschwörung oder gezielte Ablehnung solcher formalen Lösungsansätze.

Ende des Kapitels wird ein Konzept vorgeschlagen, mit dem der sozialen Komplexität von Software besser Rechnung getragen werden kann, als dies bisher in der Soziologie geschieht.

Im folgenden Abschnitt 2.1. wird vorgestellt, wie die Komplexität von Software entsteht, welche Handlungsprobleme sich daraus ergeben und wie Akteur*innen mit den Unsicherheiten umgehen, die diese Handlungsprobleme erzeugen. Dadurch wird der Raum aufgespannt, in dem die Forschungsfragen einer Soziologie der Software liegen. In Abschnitt 2.2. wird dargestellt, welche dieser Fragen bisher in der soziologischen Forschung aufgenommen und wie sie bearbeitet worden sind. Dabei zeigt sich, dass der Fokus bisher auf einzelnen Prozessen im Lebenszyklus von Software liegt, während die Wechselwirkungen zwischen diesen Prozessen, aus denen die soziale Komplexität entsteht, eine Forschungslücke darstellen. In Abschnitt 2.3. entwickle ich einen Vorschlag dafür, wie in der soziologischen Forschung mit dieser Komplexität umgegangen werden kann. Dabei steht die Erkenntnis im Mittelpunkt, dass die sozialen Prozesse, in denen Software konstruiert wird, in den Einzelteilen der untersuchten Artefakte Spuren hinterlassen. Forschende können daher durch die Analyse von Elementen der Software etwas über die Konstruktionsprozesse herausfinden, ebenso wie sie durch die Untersuchung der Prozesse Schlüsse über die Artefakte und deren Einfluss auf die Nutzung ziehen können.

2.1 Dimensionen der Komplexität von Software

Die Komplexität von Software wirkt sich zwar auf jeden sozialen Prozess aus, in dem mit Software umgegangen wird, entsteht jedoch erst aus den Wechselwirkungen zwischen all diesen Prozessen und lässt sich nicht auf einzelne zurückführen. Während einige Aspekte der Entwicklung, Einführung oder Nutzung von Software soziologisch gut untersucht sind, gibt es nur wenige Arbeiten, in denen die unterschiedlichen sozialen Prozesse, in denen ein und dieselbe Software generiert und in der mit ihr umgegangen wird, im Zusammenhang betrachtet werden.[3] Eine solche Betrachtung macht Software zum zentralen Untersuchungsgegenstand, da sie das einzige verbindende Element der verschiedenen sozialen Prozesse ist. Die in den *Software Studies* verbreitete Herangehensweise, bei der Software als Träger einer abgeleiteten oder erweiterten Handlungsfähigkeit („secondary agency“, Kitchin und Dodge 2011; Mackenzie 2006) behandelt wird, erkennt das Problem

[3] Eine Ausnahme bilden v. a. die Arbeiten zur Artefaktbiographie, die in 2.2.3 vorgestellt und diskutiert werden.

der Komplexität an, hilft aber nicht dabei, es zu bearbeiten (vgl. Einleitung). In dieser Arbeit werde ich die Rolle von Software im Sozialen stattdessen als Performativität beschreiben. *Performativität* beschreibt das folgende Verhältnis zwischen einer Theorie und der von ihr beschriebenen Praxis:[4] Eine Theorie wirkt auf einen Prozess performativ[5], wenn sie den Prozess beschreibt, in diesem zur Anwendung kommt und der Prozess sich durch diese Anwendung der Theorie verändert (MacKenzie 2006). Performativ können nicht nur Theorien, sondern allgemein Modelle wirken, die soziale Prozesse beschreiben.

Softwareentwicklung wird in der Informatik unter anderem als eine Form der Modellbildung verstanden (Ludewig und Lichter 2013). Anders als bei wissenschaftlichen Theorien ist Performativität bei den Modellen der Softwareentwicklung intendiert – sie sollen die sozialen Prozesse verändern, die sie beschreiben (zur Performativität von Software vgl. auch Orlikowski 2005). Moderne Software lässt sich jedoch nicht auf ein einziges performatives Modell zurückführen, sondern entsteht aus einer Kombination solcher Modelle, wie die folgende Rekonstruktion der Komplexität von Software zeigen wird. Dabei zeichne ich nach, wie die Prozesse der Modellbildung ineinandergreifen und dadurch Kontrolle über deren Ergebnis innerhalb der Software verteilt wird. Aus soziologischer Perspektive ist dieses Ineinandergreifen der Prozesse von Bedeutung, weil dadurch die Art und Weise, wie Handeln und Ordnung in jedem einzelnen Prozess aufeinander abgestimmt sind, mit allen anderen Prozessen in Verbindung gebracht wird. Für die Rekonstruktion der Komplexität von Software muss zuerst der Begriff des Modells definiert und aufgezeigt werden, welche Fragen Modellbildung für die Soziologie aufwirft.

Informationstechnisch betrachtet ist ein Modell das Ergebnis einer Transformation, bei der ein Teil der Eigenschaften (Reduktion) eines Urbilds oder Originals so abgebildet wird (Abbildung), dass das Modell für einen bestimmten Zweck eingesetzt werden kann (Nützlichkeit) (Ludewig 2003; Stachowiak 1973).

---

[4] Ursprünglich stammt das Konzept der Performativität aus der Sprechakttheorie, wo eine Aussage performativ genannt wird, wenn durch ihre Äußerung eine Handlung vollzogen wird (Austin 2007). Vor allem im Rahmen der Untersuchung von Finanzmärkten wurde der Begriff in die Soziologie übertragen und dort weiterentwickelt. Für die allgemeine Debatte s. v. a. Callon et al. (2007), MacKenzie (2006), MacKenzie et al. (2007).

[5] MacKenzie entwickelt in seinen Studien zur Rolle ökonomischer Theorien bei der Entwicklung moderner Finanzmärkte eine Hierarchie der Performativitätsformen. Die Differenzierungen sind für das Argument in dieser Arbeit ohne Bedeutung. Mit Performativität ist hier immer effektive oder starke Performativität nach MacKenzie gemeint (MacKenzie 2006, S. 17 ff.).

Die Reduktion entfernt einige Eigenschaften des Originals, die Abbildung erweitert jedoch das Modell um sogenannte *abundante Attribute*, also Eigenschaften, die das Original nicht besitzt und die durch das Verfahren der Transformation hinzugefügt werden (Ludewig und Lichter 2013). Soziologisch lassen sich die Prozesse der Modellbildung, die bei der Softwareentwicklung stattfinden, als Prozesse betrachten, in denen die Beteiligten versuchen, durch Entscheidungen über Eigenschaften des Modells Unsicherheit zu reduzieren, wobei sie neue Unsicherheiten generieren. Die Unsicherheiten sind eine unvermeidliche Folge der Möglichkeiten, die die Modellierung mit sich bringt: Unsicherheit besteht hinsichtlich der Frage, welches Original zu welchem Zweck und mit welchem Verfahren modelliert werden soll. Dies ist nicht festgelegt, sondern muss ausgehandelt werden. Das Ergebnis dieser Aushandlungen hängt von den sozialen Bedingungen ab, unter denen sie stattfinden, und von den Akteur*innen, die an den Aushandlungen beteiligt sind. Jede Entscheidung für eine bestimmte Interpretation des Originals, einen bestimmten Teil aller möglichen Eigenschaften, eine bestimmte Form der Abbildung oder einen bestimmten Zweck, kann einen Teil der Unsicherheit (zeitweise) verstecken. Denn: Im Modell sind die Alternativen nicht mehr sichtbar, gegen die sich die Akteur*innen in den Aushandlungen entschieden haben. Durch das Verstecken wird die Unsicherheit kontrolliert, d. h. sie kann (vorerst) von Akteur*innen, die mit dem Modell umgehen, ignoriert werden. Wie die folgenden Ausführungen zeigen werden, kann die so kontrollierte Unsicherheit aber jederzeit wieder auftauchen, wenn Entscheidungen in Frage gestellt werden.

Eine Gemeinsamkeit aller Aushandlungsprozesse liegt in den technischen Grundlagen der Programmierung, die dafür sorgen, dass jedes für Software entwickelte Modell einige der abundanten Attribute teilt. Diese werden zuerst vorgestellt. Im Anschluss werden der Softwarelebenszyklus und die Aushandlungsprozesse behandelt, mit denen die Akteur*innen in den einzelnen Phasen versuchen, Unsicherheit zu reduzieren. Die in beiden Abschnitten vorgestellten Prozesse bauen jeweils analytisch linear aufeinander auf, auch wenn sie in der Praxis zeitlich oft verschränkt stattfinden. Im letzten Abschnitt geht es dann um die rekursiven Aspekte der Softwareentwicklung. Diese verknüpfen die Ergebnisse der vorher als linear beschriebenen Aushandlungen so miteinander, dass Software für Informatiker*innen wie Soziolog*innen zum komplexen Gegenstand wird.

2.1.1 Modellierung des Prozessors

Die Entwicklung der Informatik als eigene Disziplin und die Entwicklung von Software als zentrales Element heutiger Gesellschaften beruht auf einer grundlegenden Annahme: Jeder Prozess, der als Algorithmus beschrieben werden kann, lässt sich im Prinzip durch eine universelle Turingmaschine ausführen (Davis 1995). Eine solche Maschine kann zu jedem Zeitpunkt nur eine von wenigen sehr einfachen deterministischen Operationen ausführen, bei denen ein Datum (das Eingabedatum) durch ein anderes (das Ausgabedatum) ersetzt wird. Die Frage, welche Operation wann mit welchem Ergebnis ausgeführt wird, hängt von der Operation, dem Eingabedatum und den Ergebnissen aller vorangegangenen Operationen ab. Die Maschine ist dadurch theoretisch frei von Unsicherheit für jeden, der ihre Operationen kennt, allerdings nur, wenn auch alle Eingaben und die Reihenfolge bekannt sind, in der die Operationen aufgerufen werden.

Auch moderne Computer sind auf der Grundlage universeller Turingmaschinen konstruiert. Die Annahme, dass sich alle algorithmischen Prozesse mit solchen Maschinen beschreiben lassen, bildet die Grundlage dafür, dass Software heutzutage in beinahe allen Lebensbereichen zum Einsatz kommt. Die Informatik ist durch diese Annahme darauf ausgerichtet, den Anwendungsbereich von Software als formalisierbaren Gegenstand zu behandeln, der durch systematische Analyse und Abstraktion auf ein mathematisches Modell abgebildet werden kann (Mahoney 2002). Soziologisch stellt sich die Frage, wie soziale Phänomene, die voller Unsicherheit sind – aus informationstechnischer Sicht die „Originale" der Modellierung – in solche unsicherheitsbefreiten Modelle übersetzt werden und welche Auswirkungen die Verarbeitung solcher Modelle auf die sozialen Phänomene und auf die Unsicherheit der sie tragenden sozialen Prozesse hat.

In der Informatik werden häufig Anwendungs- und Systemsoftware voneinander unterschieden. Zweck[6] der Anwendungssoftware ist die Bearbeitung menschlicher Handlungsprobleme in sozialen Prozessen. Systemsoftware stellt die Verbindung zwischen Anwendungssoftware und Hardware her (Patterson und Hennessy 2016). Sie ermöglicht es damit unter anderem, dass die statischen

[6] Zweck wird hier im Sinne von Arthur verwendet für die primäre Funktion, für die die Technik entwickelt wird und an welcher sich auch ihre Struktur orientiert (Arthur 2011, S. 29,38). Damit wird weder behauptet, dass Software (oder andere Technik) (nur) einem Zweck dienen würde, dass dieser Zweck unabhängig von Betrachter*in oder Zeitpunkt sei oder irgendwie „außersozial" im Artefakt stecken würde. Eine Auseinandersetzung mit der Mittel-Zweck-Reduktion von Technik findet sich z. B. in Rammert (1998). Da diese grundsätzliche techniksoziologische Frage am Thema dieser Arbeit vorbeiführt, gehe ich darauf im Weiteren nicht ein.

Codezeilen der Anwendungsprogramme in die dynamischen Funktionen der Software überführt werden. Für die Soziologie ist Anwendungssoftware mit Abstand der bedeutsamere Gegenstand, weil diese deutlich häufiger und direkter in soziale Prozesse eingebunden ist als Systemsoftware. Auch in dieser Arbeit wird sie im Mittelpunkt stehen. Ein grobes Verständnis der Schnittstelle zwischen Hard- und Software ist jedoch notwendig, um zu verstehen, welche Folgen die Annahme, jeder algorithmische Prozess lasse sich durch einen mathematischen Formalismus beschreiben, für den Umgang mit Software im Sozialen hat.

2.1.1.1 Übersetzung von Maschinensprache in Softwarecode

Der Übergang zwischen Hard- und Software lässt sich als eine Übersetzung betrachten, bei der ein Mechanismus zum Einsatz kommt, der die Verbindung zwischen beiden Seiten herstellt. Dieser Übersetzungsmechanismus und die eine Seite der Übersetzung werden in diesem Abschnitt kurz eingeführt, um den Leser*innen das notwendige Hintergrundwissen für ein Verständnis der Prinzipien zu vermitteln, auf denen Software aufbaut. In allen weiteren Abschnitten der Arbeit werden diese grundlegenden, aber für die soziologische Betrachtung von Software nicht weiter bedeutsamen Themen keine Rolle mehr spielen.

Als Hardware werden alle physischen Elemente von Informationstechnik bezeichnet.[7] Das zentrale Element ist der Prozessor. Ein Prozessor besteht aus Transistoren, elektronischen Bauteilen, die Elektronenströme ein- und ausschalten und durch komplexe Kombinationen dieser Vorgänge vordefinierte Operationen verursachen können. Die Operationen des Prozessors überführen als Binärzahlen codierte Eingabedaten auf deterministische Weise in ebensolche Ausgabedaten oder transformieren sie anderweitig in vorgegebener Weise. Ebenfalls binär codierte Befehle aktivieren bestimmte Elemente des Prozessors und verursachen so die Operationen. Alle möglichen Operationen des Prozessors können in einer Sprache beschrieben werden, die als Maschinensprache bezeichnet wird (Herold et al. 2011). Die Maschinensprache ist ein mathematischer Formalismus, mit dem Programme als universelle Turingmaschinen beschrieben werden können. Jedes Programm muss vor der Ausführung in diese Maschinensprache überführt werden. Alles was die Funktion der Software in irgendeiner Weise beeinflussen soll, muss durch Wörter der Maschinensprache dargestellt werden, egal ob es sich um Anweisungen handelt, die in den Codezeilen direkt beschrieben werden,

[7] Zur Hardware zählen zum Beispiel Ein- und Ausgabegeräte wie Bildschirme, Drucker, Kartenleser etc., ebenso Speicherchips, Kommunikationschips, Platinen, auf denen diese befestigt sind, Leitungen, die Daten zwischen ihnen transportieren, Sensoren und anderes. Eine Übersicht über übliche Hardwarekomponenten, ihre Funktionen und ihr Zusammenwirken findet sich in Herold et al. (2011, S. 87–128).

um Ergebnisse vorangegangener Arbeitsschritte der Software, die vom Prozessor selbst erzeugt wurden, oder um Eingaben aus der Umwelt des Computers, die durch andere Elemente der Hardware aufgenommen und weitergeleitet werden.

Aus soziologischer Perspektive werden mit der Maschinensprache die Unsicherheiten, die in der Funktion der Hardware liegen (z. B. Verschleiß, Schwankungen der Stromstärke, …) hinter der absoluten Gewissheit einer formalen Sprache versteckt.[8] Die Maschinensprache ist ein mathematisches Konstrukt. Sie besteht vollständig aus wohldefinierten Elementen. Das bedeutet: Das Ergebnis jeder Anweisung, die in Maschinensprache ausgedrückt wird, ist durch die Daten determiniert, auf denen die Anweisung operiert. Formale Sprachen wie die Maschinensprache sind damit in ihrer Funktionsweise frei von Unsicherheit, da sie genau dafür konstruiert werden. Sie werden jedoch in sozialen Prozessen mit ihren Unsicherheiten generiert und verwendet.

Der Umgang mit Maschinensprache also ist keineswegs frei von Unsicherheit: Übertragen Menschen Anweisungen und Eingabedaten für den Prozessor in Maschinensprache, ist zwar sichergestellt, dass dieser genau das definierte Verhalten zeigen wird. Aber welche Elemente dabei wie kombiniert werden, ist nicht durch das Ergebnis festgelegt. Das bedeutet, es gibt eine Vielzahl an verschiedenen Möglichkeiten, die Anweisungen zusammenzustellen, ebenso eine Vielzahl an sozialen Rahmenbedingungen (professionelle oder organisationale Prozeduren, Entscheidungsstrukturen u.ä.), die die Zusammenstellung beeinflussen. Hinzu kommt: Der Code, also die in Maschinensprache formulierte Übersetzung von Anweisungen und Daten, ist für Menschen extrem schwer zu verstehen, da er nur aus einer Reihe von Nullen und Einsen besteht. Bei praktisch allen Kombinationen von Anweisungen und Daten ist damit für Verfasser*innen und Leser*innen gleichermaßen unsicher, ob der Code genau das enthält, was die Verfasser*innen damit ausdrücken wollten.

Die Maschinensprache ermöglicht es, Computer zu steuern, ohne sich mit ihren materiellen Bedingungen auseinandersetzen zu müssen.[9] Die Nutzer*innen der ersten Computer, die gleichzeitig deren einzigen Programmierer*innen waren,

[8] Eine formale Sprache ist eine künstliche Sprache mit wohldefinierter Syntax: Sowohl die einzelnen Wörter als auch die Möglichkeiten, wie diese innerhalb der Sprache kombiniert werden dürfen, sind eindeutig festgelegt (Claus und Schwill 2003, S. 627).

[9] Obwohl die Hardware und die Prinzipien, auf denen sie basiert, entscheidend für die Funktionen von Software sind, können die mit der Entwicklung von Hardware verbundenen Unsicherheiten in den sozialen Prozessen der Softwareentwicklung ausgeblendet werden. Bei der Entwicklung von Hardware spielen zwar die physikalischen und chemischen Eigenschaften von Materialien eine große Rolle (vgl. Patterson und Hennessy 2016, S. 26 ff.), ebenso wie soziale Prozesse. Diese sind aber mit den Prozessen der Softwareentwicklung nur am Rande verbunden (vgl. Sydow et al. 2012 zur Pfadkreation in Netzwerken der Halbleiterindustrie).

konnten mathematische Aufgaben in Maschinensprache formulieren und Computer so als Rechenmaschinen verwenden. Dadurch ersetzten sie ein Handlungsproblem – die Kontrolle der Funktionen des physischen Geräts – durch ein anderes – die Formulierung von Programmen in einer formalen Sprache. In der Maschinensprache sind solche Programme jedoch umständlich zu verfassen und schwer zu durchschauen. Die Binärdarstellung bringt Unsicherheiten mit sich: können die Codezeilen vom Computer ausgeführt werden (syntaktische Unsicherheit) und tut der Computer dabei das, was sich die Programmierer*innen vorgestellt haben (semantische Unsicherheit)? Die Strategien zum Umgang mit diesen beiden Unsicherheiten unterscheiden sich ebenso wie ihre Folgeprobleme und deren Relevanz für eine soziologische Perspektive auf moderne Software.

Betrachtet man Software als Mittel zur Übergabe von Informationen an die Hardware, so stellt sich bei der syntaktischen Unsicherheit die Frage, ob diese Informationen ankommen und wie sie interpretiert werden. „Versteht" der Computer, was die Programmierer*innen mit der Software sagen? Eine Strategie zur Kontrolle dieser Unsicherheit im Softwareentwicklungsprozess ist die Automatisierung der Übersetzung: Statt in Maschinensprache schreiben Programmierer*innen Code in einer für Menschen einfacher verständlichen, aber immer noch formalen Sprache. Ferner nutzen sie einen Compiler[10], also ein Programm, das in der neuen Sprache verfasste Software in Maschinensprache übersetzt. Die automatisierte Übersetzung soll sicherstellen, dass die Software syntaktisch korrekt ist und von der Hardware akzeptiert wird. Selbst in den Frühphasen des Computers haben nicht alle Softwareentwickler*innen auch Compiler entwickelt. Aus soziologischer Perspektive trennen solche Programmiersprachen soziale Prozesse, in denen Software (Compiler) entwickelt und gestaltet wird, von solchen, in denen sie nur genutzt wird. Ein Teil der Unsicherheit darüber, ob der Code Fehler hat, wird aus dem Softwareentwicklungsprozess ausgelagert und in den Prozess der Entwicklung des Compilers verschoben.

Während Leistungsparameter der Hardware wie Geschwindigkeit oder Stromverbrauch bei der Softwareentwicklung berücksichtigt werden müssen, spielt der materielle oder soziale Ursprung solcher Parameter keine Rolle, da dieser (und die damit verbundenen Unsicherheiten) ja durch den Einsatz der Maschinensprache ausgeblendet wird.

[10] Eine Unterscheidung zwischen maschinen- und problemorientierten Sprachen sowie zwischen Compilern, die statische Maschinenprogramme generieren, und Interpretern, die den Code direkt vor der Ausführung übersetzen, wird an dieser Stelle noch nicht getroffen. Handlungsproblem (Vermeidung von Syntaxfehlern bei der Codeproduktion) und Strategie (Automatisierung der Übersetzung) sind jeweils identisch. Die Darstellung folgt Herold et al. (2011, S. 131 f).

So wie die Maschinensprache es erlaubt, Details der Hardware auszublenden, erlauben es andere Programmiersprachen, von den Details der Maschinensprache zu abstrahieren. Programmiersprachen sind Modelle für das, was mit Software ausgedrückt (und damit vom Computer getan) werden kann. Programmierer*innen können mit Hilfe dieser Programmiersprachen Probleme auf einer höheren Abstraktionsebene bearbeiten, weil die Details auf niedrigerer Abstraktionsebene bereits bei der Entwicklung der Programmiersprache selbst bearbeitet worden sind.[11] Auch ein Teil der semantischen Unsicherheit wird dadurch aus dem Softwareentwicklungsprozess ausgelagert: Das Vokabular der Sprachen ist für Menschen leichter verständlich als das der Maschinensprache. Programmierer*innen können den Code, den sie in diesen Sprachen verfasst haben, im Nachhinein besser verstehen und damit darauf vertrauen, dass die Grundbausteine ihrer Programme genau die Funktionen verursachen, die sie erwarten. Sie können dadurch komplexere Probleme bearbeiten, was sowohl in der Frühphase des Computers als auch bei der späteren Entwicklung sogenannter problemorientierter oder höherer Programmiersprachen geschehen ist (Herold et al. 2011). Höhere Programmiersprachen sind wie die Maschinensprache formale Sprachen, also künstliche Konstrukte mit wohldefinierten Elementen (Claus und Schwill 2003). Die Übersetzung zwischen höheren Programmiersprachen und der Maschinensprache basiert auf mathematischen Transformationen, die ein Programm in einer Programmiersprache auf Basis bekannter Regeln in ein Programm in Maschinensprache überführen. Die Frage, wie einfache Details (wie z. B. die Ablage von Listen im Speicher) in Code umgesetzt werden können, wird über eine mathematische Modellierung dieser Probleme gelöst. Diese Modellierung ist Gegenstand des Compilerbaus.

Die Funktionen von Compilern und anderer Systemsoftware werden bei der Entwicklung von Anwendungssoftware in der Regel als gegeben betrachtet. Aus der Perspektive der Informatik ist dies eine notwendige Bedingung für die Entwicklung von Software, die komplexe Probleme bearbeiten kann. Aus der Perspektive der Soziologie werden damit die sozialen Prozesse, in denen Systemsoftware entwickelt wird, bei der Entwicklung von Anwendungssoftware ausgeblendet. Es stellt sich die Frage, inwieweit die Akteur*innen dabei Annahmen über die Eigenschaften der Maschinensprache in Annahmen über

[11] Solche Details sind zum Beispiel in Programmiersprachen enthaltene Datentypen, die dem Prozessor anzeigen, wie die Daten in bestimmten Speicherbereichen zu interpretieren sind (z. B. als Liste von Textzeilen statt als Gleitkommazahlen) oder Funktionen, die komplexe, aber häufig genutzte Rechenschritte ausdefinieren (z. B. das Sortieren der Liste von Textzeilen nach dem Alphabet).

die Eigenschaften der Anwendungssoftware übertragen. Eine soziologische Perspektive auf Software erlaubt es, solche unhinterfragten Annahmen aufzudecken und zu untersuchen, wie bei der Konstruktion von Systemsoftware Unsicherheit reduziert wird, die beim Umgang mit Anwendungssoftware eine Rolle spielt.[12]

2.1.1.2 Übersetzung von Daten in Informationen über soziale Prozesse

Die Entwicklung von Systemsoftware bringt zwar Unsicherheit mit sich, die gelegentlich bedeutsame Auswirkungen auf den Umgang mit Anwendungssoftware hat, bildet aber aus soziologischer Perspektive nicht die Hauptgrundlage der Komplexität von Software. Die Tatsache, dass die Funktionen von Anwendungssoftware auf Systemsoftware basieren, ist eine Folge der Modellierung des Prozessors (oder allgemeiner: der Hardware) durch mathematische Abstraktionen. Programmiersprachen sind jedoch nicht nur Modelle des Prozessors, sondern vor allem Werkzeuge zur Modellierung von Phänomenen, die mit Hilfe der Funktionen der Software verändert werden sollen[13]. Frühe Programmierung diente vor allem der Bearbeitung von mathematischen Problemen, deren Modellierung anspruchsvoll sein mochte, die aber an sich wenig interpretationsoffen waren. Ganz anders sieht dies bei der Modellierung von sozialen Phänomenen aus. Soziale Phänomene sind grundsätzlich komplex; weder ist es möglich, eindeutig nicht weiter unterteilbare Grundelemente zu identifizieren, aus denen alle sozialen Phänomene bestünden, noch lassen sie sich vollständig abbilden, ohne dass dabei Informationen verloren gingen. Soziale Phänomene enthalten immer Unsicherheit, da sie auf unterschiedliche Art und Weise wahrgenommen werden können und ihre Wahrnehmung daher Entscheidungen verlangt (Luhmann 2009). Die Modellierung sozialer Phänomene mit Hilfe von Programmiersprachen stellt damit die Softwareentwickler*innen grundsätzlich vor die Frage, welche Eigenschaften das Phänomen eigentlich besitzt, welche davon für die intendierten Funktionen der Software relevant sind und wie sie in das Modell übertragen

[12] Dass diese Frage auch gesellschaftlich relevant werden kann, zeigt das Beispiel des Y2K-Problems: Vor der Jahrtausendwende war die Befürchtung verbreitet, dass Anwendungssoftware, aufgrund von Entscheidungen bei der Entwicklung von Systemsoftware, nicht in der Lage sein könnte, vierstellige statt der bis dahin häufig verwendeten zweistelligen Jahreszahlen zu verarbeiten. Dies führte zu wissenschaftlichen und gesellschaftlichen Debatten, der Entwicklung einer Vielzahl an Methoden zur Lösung des vermuteten Problems und immensen finanziellen Investitionen, mit denen der Zusammenbruch eines großen Teils der Computerinfrastruktur verhindert werden sollte (eine Betrachtung des Problems aus zeitgenössischer Perspektive findet sich z. B. bei Edwards 2008).

[13] Programmiersprachen sind damit Metamodelle für die Modelle, die mit ihnen programmiert werden (Kühne 2006).

werden sollen. Je komplexer das Phänomen, desto mehr Entscheidungen müssen getroffen werden, um den Code in Programmiersprachen zu produzieren, die wenig Raum für diese Unsicherheit bieten.

Die Funktionen der Software, die mit Programmiersprachen beschrieben werden können, basieren zwar auf deterministischen Operationen des Prozessors, ihre konkreten Ergebnisse sind bei der Programmierung aber nur schwer vorhersehbar. Dies hängt damit zusammen, dass die Funktionen hauptsächlich von den Daten bestimmt werden, die der Prozessor verarbeitet. Daten sind zum größten Teil nicht im Code festgelegt, sondern entstehen erst zur Laufzeit der Software, also dann, wenn sie genutzt wird. Wird Code als statischer und Funktionen als dynamischer Aspekt von Software betrachtet, so bilden Daten das Element, das zwischen beiden Aspekten vermittelt. Code wird ausschließlich in Prozessen der Softwareentwicklung hergestellt. Die Produktion von Daten basiert ebenfalls zum Teil auf diesen Prozessen, und zwar insofern dabei (nämlich im Code) die Struktur der Daten festgelegt sowie bestimmt wird, wann und wie diese von der Software verarbeitet werden. Zum Teil basiert sie aber auch auf dem, was während der Nutzung der Software geschieht, und zwar insofern die vordefinierten Datenstrukturen erst bei der Nutzung mit konkreten Werten gefüllt werden. Ein einfaches Beispiel: Bei der Programmierung einer Software zur Verwaltung von Krankenakten wird im Code festgeschrieben, dass Patient*innen über ihren Name und ihr Geburtsdatum identifiziert werden. Diese (und viele andere) statische Festlegungen bilden die Datenstruktur. Zur Laufzeit, also bei der Nutzung der Software, wird diese Struktur mit den Namen und Geburtsdaten konkreter Patient*innen gefüllt. Diese Namen und Geburtsdaten bilden die Daten, auf denen die Software dann operiert.

In den Nutzungsprozessen entstehen eine Unmenge an Informationen über die sozialen Phänomene, in denen die Software zum Einsatz kommt. Um bei dem Beispiel zu bleiben: Patient*innen sind Menschen mit Haar- und Augenfarbe, Berufen, Hobbies und Vorlieben, Körpergeruch, besonderen Fähigkeiten, Stimmungen und unzählbaren weiteren Eigenschaften. Das informationstechnische Modell sorgt dafür, dass nur der Teil der Informationen über das soziale Phänomen von der Software aufgenommen wird, der bei der Entwicklung als für die intendierten Funktionen relevant festgelegt wurde, und dass dies in genau einer der vielen möglichen Interpretationen geschieht. Für die Identifikation der Patientin in der Krankenakte in unserem Beispiel wären dies zum Beispiel Vor- und Nachname sowie Geburtsdatum wie auf der Versichertenkarte vermerkt, nicht Beruf und Haarfarbe und auch nicht ein selbstgewählter Spitzname und das Alter in Jahren. Nur durch diese Festlegungen lassen sich die Informationen als wohldefinierte Elemente der Programmiersprache ausdrücken. Damit aus

den ausgewählten Informationen Eingabedaten werden können, die der Prozessor verarbeiten kann, müssen sie anschließend ebenso wie der Code selbst die automatischen Übersetzungen durchlaufen, die die Systemsoftware bereitstellt. Aus soziologischer Perspektive wirkt der Code als Filter, der die Vielfalt der möglichen Informationen reduziert, mit der das soziale Phänomen beschrieben werden kann, und damit Unsicherheit (temporär) verringert. Mit Hilfe des Filters wird kontrolliert, wie das soziale Phänomen in Daten übersetzt wird.

Software übernimmt jedoch nicht nur Daten, die direkt aus Informationen über den Nutzungsprozess übersetzt worden sind, sondern produziert aus diesen auch neue Daten, die dann Grundlage neuer Verarbeitungsschritte werden. Auf der Ebene des Prozessors lassen sich beide Arten von Daten nicht voneinander unterscheiden. Auch für die Funktionen der Software ist dieser Unterschied nicht relevant. Wird Software in sozialen Prozessen eingesetzt, ist dadurch für die Nutzenden nicht mehr ohne weiteres nachvollziehbar, worauf die Daten zurückzuführen sind, auf denen die Funktionen der Software beruhen. Geben die Funktionen Daten an den Nutzungsprozess zurück (zum Beispiel, indem sie auf dem Bildschirm angezeigt werden), gewinnen sie wieder Unsicherheit, da sie nun erneut interpretiert werden müssen. Die Ergebnisse dieser Interpretation gehen wieder als Informationen in den Nutzungsprozess ein. Dieser Ablauf ist in Abbildung 2.1 dargestellt.

Soziologisch stellen sich dazu vor allem zwei Fragen, nämlich

1. Inwiefern verändert die Software, wie das Soziale interpretiert wird? und
2. Wie, wann und von wem wird Einfluss auf die Interpretationsmöglichkeiten genommen?

Die erste Dimension der Komplexität von Software wird hier als *Übersetzungsdimension* bezeichnet. Software übersetzt soziale Prozesse, die voller Unsicherheiten sind, in die deterministischen, aber für Menschen undurchschaubaren Operationen des Prozessors. Durch den Umgang mit Software werden die Ergebnisse dieser Operationen wieder zum Teil sozialer Prozesse und gewinnen neue Unsicherheiten. Die Software wird im Sozialen performativ, weil die Nutzenden mit den Daten, die sie produziert, konfrontiert werden und sie als Informationen über soziale Prozesse interpretieren. Die Akteur*innen, die mit Software umgehen, wissen in der Regel nichts (oder nur sehr wenig) über all die sozialen Prozesse, die diese Übersetzung möglich machen und übertragen somit Erwartungen über die Funktionsweise des Prozessors unhinterfragt auf die Funktionen

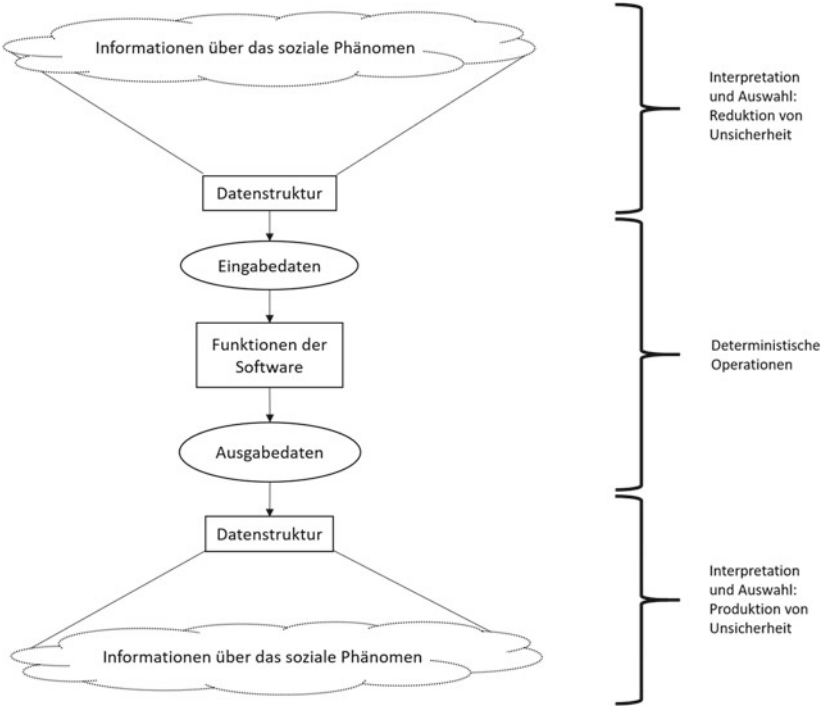

Abbildung 2.1 Reduktion und Produktion von Unsicherheit bei der Nutzung von Software

der Software. Für eine soziologische Perspektive auf Software sind daher die Prozesse bedeutsam, die die Übersetzung zwischen der Maschine und dem Sozialen ermöglichen. Diese Prozesse werden im nächsten Abschnitt beschrieben.

2.1.2 Software-Lebenszyklus

Im vorigen Abschnitt wurden Programmiersprachen als Modelle des Prozessors betrachtet. Die Funktionen des Prozessors (Original) werden mit Hilfe von Programmiersprachen durch mathematische Transformationen (Abbildungen) so umgeformt, dass sie von Menschen einfacher verstanden werden können (Zweck). Im Folgenden wird dargestellt, wie die sozialen Prozesse modelliert werden, die durch Software beeinflusst werden sollen. Programmiersprachen stehen aus

diesem Blickwinkel nicht mehr als Modelle im Fokus, sondern als Abbildungsverfahren, mit denen ein soziales Phänomen in mehreren Schritten in ein Modell, den statischen Code der Software, überführt wird.

Aus der Perspektive von Informatiker*innen besteht der Softwareentwicklungsprozess aus einer Kette von Abbildungen, in denen unterschiedliche Verfahren nacheinander kombiniert werden und so ein soziales Phänomen auf ein Modell in Form von Softwarecode abgebildet wird. Gegenstand der Forschung und Ausbildung im Bereich der Softwareentwicklung ist die Frage, wie die Verfahren gestaltet und auf welche Art und in welcher Reihenfolge sie am besten kombiniert werden müssen, um ein soziales Phänomen so abzubilden, dass die Funktionen, die der Code hervorruft, den Zweck erfüllen, der Anlass für die Modellierung war. Aus soziologischer Perspektive stellen sich hingegen andere Fragen: Wie, von wem und unter welchen Bedingungen wird festgelegt, was der Zweck der Modellierung ist, welches soziale Phänomen in Software abgebildet werden soll, welche Eigenschaften für die Abbildung ausgewählt werden und wann die Software gut funktioniert? In anderen Worten: Wie, wann und wo werden die Unsicherheiten kontrolliert, die der Umgang mit Software aufwirft? Im Folgenden stelle ich dar, an welchen Stellen im Softwareentwicklungsprozess diese Fragen aufkommen, wie die Akteure mit ihnen umgehen und welche Auswirkungen dieser Umgang auf die Software und ihre Nutzung hat.

2.1.2.1 Rationalistische Tradition der Softwareentwicklung und Softwarelebenszyklus

Trotz eines lange etablierten, von den Sozialwissenschaften beeinflussten und eher konstruktivistisch orientierten Diskurses in der Informatik (vgl. z. B. Floyd 1989; Rolf 1990; Floyd 1992; Rönkkö et al. 2005; Wulf et al. 2018) folgt Softwareentwicklung in der industriellen Praxis größtenteils einer rationalistischen Tradition (Floyd 1989, S. 4). Diese basiert auf der Grundannahme, dass eine objektiv gegebene Realität existiert, die zumindest in der Theorie richtig und vollständig erkannt werden kann, auch wenn dies in der Praxis möglicherweise nicht immer gelingt. In dieser Realität stellen sich eindeutig kommunizierbare Probleme, die von Softwareentwickler*innen analysiert, modelliert und durch die Konstruktion eines Informationssystems gelöst werden können (a.a.O.). Diese Tradition spiegelt sich auch in den klassischen Verfahrensanweisungen für den Softwareentwicklungsprozess, den sogenannten Vorgehensmodellen der Softwareentwicklung. Diese Vorgehensmodelle beschreiben, welche Aufgaben in welcher Reihenfolge erfüllt und welche Methoden eingesetzt werden sollen, um möglichst effizient möglichst fehlerfreie Software herzustellen. Sie sollen Softwareentwickler*innen dabei helfen, die Unsicherheiten des Entwicklungsprozesses

durch Regeln so weit unter Kontrolle zu bekommen, dass sie die zu lösenden Probleme richtig identifizieren und dafür die unter den gegebenen Rahmenbedingungen bestmögliche Lösung konstruieren können. Klassische Vorgehensmodelle unterscheiden sich untereinander vor allem darin, wie die vorgesehenen Aktivitäten zeitlich arrangiert sind, welche Akteur*innen wann wie stark beteiligt werden und wie und wann Übergänge zwischen den Aktivitäten vorgesehen sind. Alle Verfahren *„presuppose a pre-given object domain (including organizations, functions and processes) which can be represented through the neutral (observation) language of conceptual modelling, data flow diagramming or data modelling and the like […]“* (Bloomfield 1995, S. 493).[14] Eine für soziologische Fragen offenere Haltung kommt in den innerhalb der vergangenen Jahre populärer gewordenen agilen Ansätzen zur Softwareentwicklung zum Ausdruck. Diese Alternativen zu klassischen Vorgehensmodellen basieren auf der Annahme, dass Unsicherheit im Sozialen unvermeidbar ist und in Softwareentwicklungsprozesse integriert werden muss, weil alle Versuche, sie durch Regeln möglichst auszuschließen, zu einem schlechten Entwicklungsprozess und schlechter Software führen. Zentrale Unterschiede zwischen der klassischen und der agilen Perspektive werden im Anschluss an die klassischen Verfahren kurz vorgestellt.

Der Zeitraum vom Beginn der Planung bis zum Ende der Nutzung einer Software an irgendeinem Einsatzort wird in der Informatik als Softwarelebenszyklus[15] bezeichnet (Herold et al. 2011). Der Lebenszyklus unterteilt diesen Zeitraum in verschiedene Phasen, die durch die jeweils zentralen Aufgaben für Informatiker*innen bestimmt werden. Während über die Anzahl der Phasen und die genaue Grenzziehung keine Einigkeit besteht,[16] liegt in allen Varianten deutlich mehr Gewicht auf den Aktivitäten der Produktion vor der ersten Nutzung. Wie Nutzer*innen, Administrator*innen oder Entwickler*innen nach der Fertigstellung der ersten funktionsfähigen Version mit der Software umgehen, wird wenig differenziert betrachtet. Die Phasen des Lebenszyklus und

[14] Auch Methoden, die anerkennen, dass unterschiedliche Perspektiven auf einen Gegenstandsbereich für die Gestaltung von Software relevant sind, sind darauf ausgerichtet, im Laufe des Entwicklungsprozesses eine einheitliche Beschreibung aus dieser Vielfalt zu generieren (Bloomfield und Vurdubakis 1997).

[15] Der Begriff Lebenszyklus wird hier rein metaphorisch verwendet, da er im Zusammenhang mit Software allgemein geläufig und anschaulich ist, obwohl Software dadurch unglücklich naturalisiert wird (problematisch, aber schwer ersetzbar finden den Begriff auch Ludewig und Lichter 2013, S. 155).

[16] Die OECD-Definition (nach Ludewig und Lichter 2013, S. 156) benennt z. B. 8–9 typische Phasen, während Herold et al. (2011, S. 130 f.) nur fünf beschreiben.

die Aufgaben, mit denen klassische Vorgehensmodelle die Arbeit im Softwareentwicklungsprojekt strukturieren, sind daher zu großen Teilen identisch (eine typische Darstellung des Softwarelebenszyklus zeigt Abbildung 2.2). Da diese Aufgaben in modernen Softwareentwicklungsprojekten in der Regel zeitlich nicht eindeutig getrennt werden können (a.a.O., S. 56), werden sie in den meisten Vorgehensmodellen nicht als Phasen bezeichnet. Für die soziologische Perspektive ist das Konzept der Phasen jedoch hilfreich, da es erlaubt, die Abhängigkeiten zwischen den einzelnen Modellen des sozialen Phänomens in linearer Reihenfolge sichtbar zu machen. Diese Abhängigkeiten entstehen, da in jeder Phase das Modell, das im Rahmen der vorangegangenen Phase geschaffen wurde, als Vorlage für eine weitere Modellierung genutzt wird, in deren Rahmen einige der noch interpretationsoffenen Eigenschaften der Vorlage (z. B. „Schriftfarbe Blau“) in wohldefinierte Eigenschaften des Modells (z. B. „Schriftfarbe #5F9EA0“) übersetzt werden. Der Softwarelebenszyklus wird daher hier anhand der Phasen vorgestellt, auch wenn die Aktivitäten sich in der Praxis oft zeitlich überlappen.

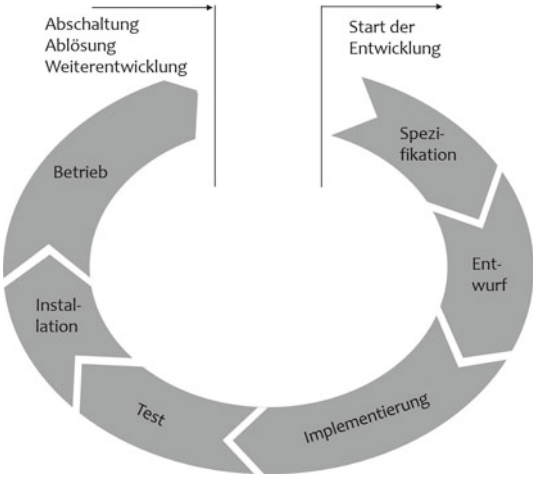

Abbildung 2.2 Software-Lebenszyklus. (Eigene Darstellung in Anlehnung an Balzert 2011, S. 1)

2.1.2.2 Softwareentwicklung im Projekt

Aus soziologischer Perspektive können Softwareentwicklungsprojekte als temporäre Organisationen betrachtet werden (Lundin und Söderholm 1995). Sie können

innerhalb der Grenzen einer Organisation gegründet werden oder auch durch Kollaboration mehrerer Organisationen entstehen. Sie besitzen eine im Voraus festgelegte Lebenszeit, innerhalb derer eine bestimmte Aufgabe (hier die Produktion von Software oder die Modellierung eines sozialen Phänomens für den Computer) von einem dezidiert dazu abgestellten Team erfüllt werden soll. Wie alle Projekte sind auch Softwareentwicklungsprojekte auf vielfältige Weise mit den Organisationen verbunden, aus denen sie hervorgehen: Diese setzen die Projekte in Gang und geben initial Aufgaben und Lebenszeit vor, sie stellen eigene Mitglieder für das Team ab und finanzieren die Mitarbeit von Externen, sie legitimieren und unterstützen die Arbeit in den Projekten und bewerten, ob die Aufgabe erfüllt wurde oder noch erfüllt werden kann (a.a.O., S. 439–444). Zentrale Fragen für die Modellierung im Softwareentwicklungsprozess stellen sich damit bereits vor der Initiierung des Projekts. Um die Modellierung zu ermöglichen, müssen Organisationen verschiedene Unsicherheiten kontrollieren: Der Zweck, den die neue Software für die Organisationen erfüllen soll, muss festgelegt, das zu modellierende Phänomen ausgewählt und beides ebenso wie die dafür aufzuwendenden Ressourcen legitimiert werden. Die Initiator*innen müssen dazu innerhalb ihrer Organisation(en) durchsetzen, dass der Zweck als relevantes organisatorisches Problem und das Projekt als realistische Lösung für dieses Problem anerkannt wird. Mitarbeiter*innen müssen freigestellt, Zeitrahmen und Budget festgelegt und (unter Umständen) Verträge mit anderen Organisationen ausgehandelt und unterschrieben werden. Die Organisationssoziologie zeigt, dass solche Fragen nicht unproblematisch anhand übergeordneter Organisationsziele beantwortet werden können, sondern Gegenstand von Auseinandersetzungen sind, die von den Interessen und Machtressourcen der Mitglieder (Perrow 1986; Crozier und Friedberg 1979), gesellschaftsweiten oder feldspezifischen Institutionen (Scott 2014; Friedland und Alford 1991) und der Historie der Organisation (Luhmann 2000) abhängen, in denen sie stattfinden. Die Kontrolle über Zweck, Rahmenbedingungen des Softwareprojekts, Mitglieder und Bewertung wird in sozialen Prozessen ausgeübt, die vor der Initiation des eigentlichen Modellierungsprozesses stattfinden. Welche Software entsteht, hängt damit unmittelbar davon ab, wie diese vorgelagerten Prozessen aussehen.

2.1.2.3 Spezifikation

In der Logik klassischer Vorgehensmodelle wird zu Beginn der Softwareentwicklung in der Phase der *Spezifikation* (s. Abbildung 2.2, S. 40) festgelegt, welche Leistungen die Software erbringen und unter welchen Rahmenbedingungen sie arbeiten soll. Dazu wird zuerst ein Systemmodell erstellt, wobei als System in diesem Fall nicht allein das technische Artefakt, sondern das

gesamte sozio-technische System bezeichnet wird, in dem dieses zum Einsatz kommen soll (Pressman 1997). Als sozio-technisches System kann je nach Einsatzzweck eine Abteilung, eine Organisation oder auch ein Zug mit Passagieren betrachtet werden. Anhand des Systemmodells sollen die Probleme des sozio-technischen Systems identifiziert und festgelegt werden, welche davon durch Software gelöst werden sollen. Elemente, Eigenschaften und Leistungen der Software sowie ihre technische Umgebung (z. B. Vorgaben zur Hardware, bereits vorhandene Software) sollen dann in einem Spezifikationsdokument (kurz: Spezifikation) möglichst genau beschrieben werden. Die Spezifikation ist damit ein Modell, das die für die Funktionen der Software als relevant erachteten Eigenschaften des sozialen Phänomens enthält, die anhand des Systemmodells und der Vorgaben der Initiationsphase des Projekts festgelegt wurden. Bei der Entwicklung von Individualsoftware[17] bildet die Spezifikation häufig die Basis des Vertrags zwischen Kund*innen- und Entwickler*innenorganisation.

Technisches und sozio-technisches System werden in der Informatik als „zweckgerichtete Sammlung von miteinander verbundenen Komponenten" betrachtet, „die zusammenwirken, um ein bestimmtes Ziel zu erreichen" (Sommerville 2012). Diese Sichtweise verweist darauf, dass die Funktionen der Software und der Kontext, in dem sie zum Einsatz kommt, einander beeinflussen und dass auch einzelne Veränderungen auf einer Seite auf die andere Seite zurückwirken. Sie blendet aus, dass das Soziale in sozio-technischen Systemen nicht in klar umrissene Komponenten zerlegbar ist, deren Zweck feststeht, sondern Unsicherheiten mit sich bringt. Widersprüchliche und ambivalente Ziele, Konflikte und wechselnde Koalitionen zwischen Akteur*innenn und Abteilungen, routiniertes statt zweckorientiertes Handeln und differierende Interpretationen dessen, was Ziel, Zweck oder angezeigtes Zusammenwirken in konkreten Situationen bedeuten, kurz, ein großer Teil der sozialen Eigenschaften des sozio-technischen Systems werden bei der Abbildung auf das Systemmodell reduziert. Diese Reduktion ist zum Teil Ergebnis der Vorgaben der Initiator*innen, die vor Projektbeginn ausgehandelt wurden, zum Teil das Produkt des Modellierungsprozesses selbst. Mögliche Auseinandersetzungen über viele der Fragen, die sich aus solchen Facetten der Unsicherheit des Sozialen ergeben, können in diesem Prozess durch einen Verweis auf das Projektziel, den Zeitplan oder die verfügbaren Ressourcen beendet werden (Lundin und Söderholm 1995). Damit werden Unsicherheiten des

[17] Als Individualsoftware bezeichnet man Software, die speziell für eine Kund*in, in der kommerziellen Softwareentwicklung in der Regel eine bestimmte Organisation, entwickelt wird. Anders als Standardsoftware entsteht Individualsoftware auf Auftrag von und in Zusammenarbeit mit der Kund*in. Sie kann sowohl von der Auftraggeber*in selbst (bzw. der internen IT-Abteilung) als auch von externen Dienstleister*innen entwickelt werden.

aktuellen Prozesses gezielt an den vorangegangenen ausgelagert. Aufgelöst sind sie dadurch aber nicht! Es kann nicht ausgeschlossen werden, dass die gleichen Fragen zu einem späteren Zeitpunkt oder in einem anderen Kontext wieder aufgegriffen werden. In so einem Fall kommt die scheinbar an den vorgelagerten Prozess abgeschobene Unsicherheit in einem nachfolgenden Prozess unerwartet wieder zum Vorschein. Auch wenn solche Spätfolgen möglich sind: Die immer temporäre Kontrolle, die durch solche Verweise auf den Rahmen des Projekts ausgeübt wird, verhindert, dass der Modellierungsprozess unmittelbar zum Erliegen kommt, und sichert damit das Fortbestehen des Projekts.[18]

In klassischen Vorgehensmodellen des Software Engineerings setzt sich das Projektteam in dieser Phase aus Repräsentant*innen der zukünftigen Benutzer*innen[19], den *Stakeholder*innen*, und Informatiker*innen, den *Systemanalyst*innen*, zusammen. Nach der Logik der Vorgehensmodelle gibt es eine klare Arbeitsteilung zwischen den Gruppen: Die Stakeholder*innen sollen Wissen über das soziale Phänomen und den Zweck der Modellierung bereitstellen. Die Systemanalyst*innen sollen dieses Wissen erheben, es in Anforderungen an die Software und ein Systemmodell übersetzen und daraus die Spezifikation entwickeln. In der Spezifikation sollen die Anforderungen der Stakeholder*innen in Form von eindeutig messbaren Eigenschaften dargestellt sein, deren Einhaltung später bei der Validierung überprüft werden kann (Pfleeger und Atlee 2006).

Unsicherheiten im Modellierungsprozess, die zum Beispiel durch widersprüchliche und unvollständige Aussagen der Stakeholder*innen über das System oder die Funktionen, die von der Software erwartet werden, entstehen, sind in der Informatik vorgesehen (Herold et al. 2011). Sie werden aber als Probleme betrachtet, die sich mit Hilfe geeigneter Methoden auf Dauer kontrollieren lassen (Sommerville 2012).[20] Ebenso problematisiert wird, dass die in der Spezifikation enthaltene Beschreibung der Anforderungen immer einen gewissen Grad an Interpretationsspielraum lässt, der von Akteur*innen in späteren Phasen

[18] Nach Lundin und Söderholm sind Fortschritte im Prozess für das Überleben temporärer Organisationen zentral, weswegen sie stark darauf angewiesen sind, dass einmal getroffene Entscheidungen nicht mehr in Frage gestellt werden. Sie weisen darauf hin, dass dieses In-Frage-Stellen eine erfolgversprechende Strategie ist, um Projekte aufzuhalten oder komplett zum Scheitern zu bringen (Lundin und Söderholm 1995, S. 453).

[19] Bei der Entwicklung von Individualsoftware werden diese Repräsentant*innen oft direkt als „Endnutzer*innen“ (z. B. Sommerville 2012, S. 65) oder „Kund*innen“ (z. B. Herold et al. 2011, S. 464) bezeichnet.

[20] Der Umgang mit neu aufkommenden Anforderungen, den Sommerville als Anforderungsmanagement bezeichnet (Sommerville 2012, S. 145–148), wird im Abschnitt Evolution diskutiert.

unterschiedlich ausgelegt wird (z. B. a.a.O., S. 118). Aus soziologischer Perspektive ist im Zusammenhang mit diesen Fragen vor allem interessant, dass die Auswahl des Projektteams zur Kontrolle einiger dieser Unsicherheiten beiträgt, andererseits aber dazu führt, dass andere verstärkt werden. Einer dieser Kontrollmechanismen ist die Einbettung der Projektmitglieder in weitere soziale Kontexte. Projektmitglieder sind in aller Regel nicht nur gleichzeitig Mitglieder der Mutterorganisation, sondern gehören auch einer Abteilung, Hierarchiestufe, informellen Koalition u.ä. an. Die Auswahl der Teammitglieder macht es wahrscheinlich, dass bestimmte Fragen und Deutungen im Projekt (nicht) zum Vorschein kommen. Dies vereinfacht zunächst, dass das Modell überhaupt entsteht. Es begünstigt auch, dass einige Eigenschaften des sozialen Phänomens als relevanter eingeschätzt und damit eher in das Modell übernommen werden als andere. Dadurch wird nicht nur Unsicherheit im Modellierungsprozess kontrolliert, sondern auch für alle weiteren Phasen im Lebenszyklus der Software – so zumindest die implizite Annahme. Ein anderer Mechanismus, der einige Unsicherheiten kontrolliert und andere verstärkt, ist die Ausbildung in den Methoden des Software Engineering. Die Methoden des Software Engineering verhindern viele Unklarheiten und mögliche Auseinandersetzungen von vorneherein, indem sie Stakeholder*innen von Systemanalyst*innen systematisch voneinander trennen,[21] allen bestimmte Aufgaben zuweisen und für jede Gruppe Handlungsvorlagen bereitstellen. Die in der Ausbildung vermittelte rationalistische Grundhaltung sorgt andererseits aber auch dafür, dass z. B. Unterschiede in den Perspektiven der Stakeholder*innen für die Systemanalyst*innen systematisch ausgeblendet werden. Soziologisch lässt sich in diesem Zusammenhang sowohl fragen, wie sich die unterschiedlichen Zugehörigkeiten der Akteur*innen auf ihre Interpretation des Modellierungsgegenstands auswirken, als auch, welche Möglichkeiten sie dadurch haben, diese Interpretationen im Modellierungsprozess zur Geltung zu bringen.

Bei der Entwicklung von Standardsoftware können Mitglieder der Organisation, die die Software herstellt, die Rolle der Stakeholder*innen übernehmen. Es stellen sich ähnliche Fragen wie bei Individualsoftware, selbst wenn das Projekt hauptsächlich von einer einzigen Organisation getragen wird. Häufig werden in diesem Fall jedoch Repräsentant*innen der zukünftigen Kund*innengruppe in irgendeiner Form einbezogen (Pollock et al. 2007) oder die Softwarehersteller*in erweitert ein als Individualsoftware entwickeltes Produkt zum Standardprodukt weiter (Schmidt 2008). In solchen Fällen ist aus soziologischer Perspektive nicht nur interessant, wie Teammitglieder innerhalb ihrer Organisationen konkret ausgewählt werden, sondern auch, welche Gruppen oder Organisationen

[21] Ausnahmen, in denen beide Seiten Softwareentwickler*innen sind, existieren natürlich.

als repräsentativ für die Kund*innengruppe definiert werden und dadurch Einfluss auf die Softwareentwicklung nehmen können. Zudem stellt sich die Frage, wie, von wem und zu welchem Zeitpunkt im Modellierungsprozess die Anforderungen von spezifischen Erfahrungen der Teammitglieder zu repräsentativen Eigenschaften eines generischen Modells werden. Gerade bei Standardsoftware, die in vielen unterschiedlichen sozialen Kontexten eingesetzt wird, ist später nur schwer nachvollziehbar, welche möglichen Alternativen durch solche Entscheidungen ausgeblendet worden sind und an welchen sozialen Prozess die Kontrolle über diese Unsicherheiten damit ausgelagert worden ist. Bei der Nutzung ist damit häufig unmöglich festzustellen, welcher Logik konkrete Eigenschaften der Software folgen, welche Zwecke damit verbunden sind und an wen Fragen zu oder Kritik an den Entscheidungen, die zu diesen Eigenschaften geführt haben, überhaupt adressiert werden könnten.

2.1.2.4 Entwurf

Im ersten Schritt des Modellierungsprozesses wird laut klassischer Vorgehensmodelle festgelegt, *was* die Software tun soll, bevor im zweiten Schritt die Frage geklärt wird, *wie* dies zu geschehen hat. In dieser zweiten Phase des Softwarelebenszyklus (Abbildung 2.2, S. 40), dem *Entwurf,* soll ein neues Modell konstruiert werden, in welchem alle Eigenschaften aus dem Spezifikationsdokument abgebildet sind, die in Softwarecode überführt werden können. Diese Eigenschaften aus dem Spezifikationsdokument werden als Systemanforderungen bezeichnet (Sommerville 2012).[22] Ergebnis dieses Modellierungsschritts soll eine sowohl fachlich als auch technisch detailreiche und möglichst eindeutige Beschreibung der Software sein, die dann programmiert (also in Code überführt) oder aus bereits bestehenden Teilen zusammengesetzt werden kann. Zwar gibt es unterschiedliche Paradigmen und Methoden für den Entwurf von Software, sie teilen sich aber die zentrale Grundannahme, dass sich alle Systemanforderungen

[22] Die Spezifikation enthält neben den Systemanforderungen auch Benutzer*innenanforderungen, die nicht direkt in das nächste Modell (den Entwurf) übernommen werden, sondern erst bei der Validierung wieder Bedeutung im Prozess bekommen. In der Literatur zum Software Engineering finden sich verschiedenste Kategorisierungen von Anforderungen sowie eine Auseinandersetzung mit den Voraussetzungen und Folgen für die Modellierung (z. B. Sommerville 2012, S. 123 ff.; Herold et al. 2011, S. 366 ff.). Dabei zeigt sich, dass die Informatik nicht blind für soziologische Fragen ist, ihre Perspektive es aber verlangt, solche nicht vollständig lösbaren Probleme als unvermeidbare Abweichungen vom idealen Vorgehen zu konstituieren.

in irgendeiner Form durch das Verhältnis von statischen (Daten) und dynamischen (Funktionen) Elementen beschreiben lassen (Rolf et al. 1998).[23]

Der Umgang mit verschiedenen Aspekten der Unsicherheit des Sozialen wird auch in klassischen Vorgehensmodellen der Softwareentwicklung problematisiert. Dies betrifft allerdings nur die Spezifikationsphase. Mit der Fertigstellung des Spezifikationsdokuments verschwindet der Umgang mit diesen Unsicherheiten aus dem offiziellen Aufgabenbereich der Softwareentwickler*innen. In der Entwurfsphase geht es nur noch um die Modellierung der Spezifikation, nicht mehr um die des sozio-technischen Systems (Sommerville 2012). Unsicherheiten im Entwurf beruhen damit auf Ungenauigkeiten in der Spezifikation. Soziologisch ist dies anders zu bewerten: Für die Überführung der Anforderungen in den Entwurf gibt es keine eindeutig definierte Vorgabe, es sei denn, man bezeichne das Fehlen einer solchen als eben diese. Vielmehr gilt der Entwurf im Software Engineering als kreativer Prozess, der vor allem Erfahrung verlangt (Sommerville 2012). Damit entstehen zusätzlich zu den in der Entwurfsphase ausgeblendeten (aber nicht aufgelösten) Unsicherheiten aus der Phase der Spezifikation neue Unsicherheiten darüber, auf welche Weise die Spezifikation am besten in den Entwurf zu überführen ist. Wie das erfolgen soll, muss daher neu ausgehandelt werden. Wer diesen Prozess gestalten kann, ist bedeutsam. Soziologisch interessant ist daher, wie das alles zwischen den Organisationen, die das Projekt tragen, zwischen den Organisationen und dem Projektteam und zwischen den Mitgliedern innerhalb des Teams ausgehandelt wird und wie dies den weiteren Prozess und damit auch die nächsten Modelle beeinflusst.

Eine der grundlegenden Fragen, über die im Rahmen des Entwurfsprozesses verhandelt wird ist, aus welchen Komponenten sich die Software zusammensetzen soll und welche Verbindungen zwischen diesen bestehen sollen. Diese als *Software-Architektur* bezeichnete Grundstruktur wird im Entwurfsprozesses als erstes modelliert. Alle weiteren Modellierungsschritte werden auf dem Architekturmodell aufgebaut. Die Architektur kann später, wenn überhaupt, nur mit großem finanziellen, zeitlichen und häufig auch mikropolitischen Aufwand verändert werden, da mit ihr auch ein großer Teil des Codes geändert oder zumindest

[23] Der funktionsorientierte Ansatz geht von den Abläufen aus und konstruiert das Datenmodell entlang der Prozesslogik, der datenorientierte Ansatz beginnt mit der Analyse der anfallenden Daten und modelliert erst in einem nachgeordneten Schritt die Funktionen, die diese Daten miteinander in Verbindung setzen. Der objektorientierte Ansatz verbindet beide Ideen, indem er Daten und Funktionen in einem möglichst realitätsnahen und für Außenstehende verständlichen Modell vereint. Im Mittelpunkt stehen hier die Elemente des Systems mit ihren Eigenschaften (Daten) und Fähigkeiten (Funktionen). Die zeitlichen und sachlichen Beziehungen der Entitäten sind (im Modellierungsprozess) nachgeordnet.

auf seine Funktion überprüft werden müsste. Aus diesem Grund bleibt sie üblicherweise auch in ansonsten auf Flexibilität ausgelegten Entwicklungsprojekten dauerhaft stabil (Sommerville 2012). Soziologisch betrachtet ist die Modellierung der Software-Architektur in ihrer Bedeutung für alle folgenden Phasen des Lebenszyklus darin dem Prozess der Initiation des Softwareprojekts gleichzusetzen. Durch die Architekturentscheidung zu Beginn wird der Teil der Aufgabe des Softwareprojekts, der noch nicht als gelöst angesehen wird, in Einzelaufgaben unterteilt, die im weiteren Verlauf von Entwurf und Implementierung größtenteils unabhängig voneinander bearbeitet werden sollen. Der Verweis auf die Software-Architektur, die bei ihrer Modellierung getroffenen Entscheidungen und die Kosten, die Veränderungen daran mit sich bringen würden, kann damit in allen späteren Phasen des Lebenszyklus genutzt werden, um Auseinandersetzungen über die Softwareentwicklung abzuwehren. Gelingt dies, werden Unsicherheiten (temporär) kontrolliert, indem sie an den Entwurfsprozess ausgelagert werden – wie immer ohne dass sie dadurch dauerhaft aufgelöst werden könnten.

2.1.2.5 Implementierung[24]

Auf den Entwurf, Phase 2 im Softwarelebenszyklus, folgt die Phase der *Implementierung* (s. Abbildung 2.2, S. 40). In dieser Phase soll aus dem Modell, das im Entwurfsdokument festgehalten ist, die funktionsfähige Software programmiert werden. Die Programmierung ist damit der letzte Schritt der Modellierung eines sozialen Phänomens für den Computer, der nicht vollständig automatisiert ist.[25] Das Ergebnis der Programmierung ist ein Modell in einer Programmiersprache. In der Phase der Implementierung müssen die letzten am Ende der Entwurfsphase noch im Modell verbliebenen Unsicherheiten geschlossen werden, weil es in den formalen Programmiersprachen nicht möglich ist, Unsicherheiten auszudrücken. Alles Programmierte wird eindeutig in Operationen des Prozessors übersetzt. Ähnlich wie der Entwurfsprozess gilt jedoch auch die Übersetzung des Entwurfs in Programmcode als nicht vollständig formalisierbar: *„Die Programmierung ist eine individuell gestaltete Aktivität und es gibt dafür keinen allgemeingültigen Prozessverlauf“* (Sommerville 2012, S. 67). Die einzelnen Programmierer*innen

[24] Der Begriff „Implementierung“ wird in dieser Arbeit immer entsprechend dem hier dargelegten Begriffsverständnis der Informatik verwendet und hat nichts mit der Einführung von Standardsoftware in Organisationen zu tun, die in betriebs- und sozialwissenschaftlichen Studien häufig ebenfalls als Implementierung (vgl. z. B. Robey et al. 2002) bezeichnet wird.

[25] Alle weiteren Schritte, die nötig sind, bevor die Software vom Prozessor ausgeführt werden kann, werden automatisch durch Übersetzungsprogramme erledigt. Details dazu werden in Abschnitt 2.1.1.1, S. 26, beschrieben.

besitzen bei der Implementierung damit trotz aller in den vorangegangenen Phasen erfolgten Festlegungen immer noch einige individuelle Handlungsspielräume zur Mitgestaltung des Endergebnisses. Obwohl jede einzelne Programmierer*in eigene Teile des Codes schreibt, ist die Implementierung primär eine kollaborative Aufgabe. Denn je nach Umfang der Software definieren ein Team oder mehrere Teams von Softwareentwickler*innen zusammen das Gesamtergebnis und leiten aus diesem dann immer kleinere Teilaufgaben ab, die schließlich von Einzelpersonen ausgearbeitet werden.

Soziologisch ist in der Phase der Implementierung vor allem interessant, wie Programmierer*innen mit den Unsicherheiten aus dem Entwurf umgehen und welche Rolle ihre Einbindung in verschiedene soziale Zusammenhänge dabei spielt. Nach Einflussfaktoren kann dabei im Team des (Teil)projekts (Button und Sharrock 1996), innerhalb der Organisation, die Programmierer*innen beschäftigt (Barrett 2004) oder in der Profession (Bloomfield und Vurdubakis 1997; Funken 2001) gesucht werden. Wie in allen vorangegangenen Phasen bringt die Einbettung in diese sozialen Zusammenhänge einerseits Mechanismen mit sich, über die sich Unsicherheit verringern lässt, andererseits aber auch solche, die neue Unsicherheit in den Prozess hineinbringen. Ein Beispiel für ersteres sind Stellenbeschreibungen, die verschiedenen Teammitgliedern eindeutige Aufgaben und Entscheidungsbefugnisse zuweisen. Beispiel für zweiteres ist die Mitgliedschaft in einem Professionsverband, der Prinzipien für gute Programmierung festlegt und seine Mitglieder dazu animiert, ihre eigene Arbeitspraxis zu hinterfragen. Während im ersten Fall das Auftreten von Unsicherheit (bei der Aufgabenverteilung) im Voraus verhindert wird, erzeugt die Kritik an den Programmierprinzipien im Projekt neue Unsicherheit durch den Anstoß von außen. In beiden Fällen versuchen Akteur*innen aber, Unsicherheit an andere soziale Prozesse auszulagern, indem sie sich auf die sozialen Zusammenhänge beziehen, in die sie zusätzlich zum Projekt eingebettet sind.

Moderne Software wird in der Implementierungsphase praktisch nie vollständig neu programmiert. Stattdessen werden mehr oder weniger bereits vorhandene Programme oder Programmbausteine mit neu programmiertem Code verbunden (Claus und Schwill 2003). Die Folgen der Wiederverwendung für die soziale Komplexität der Software werden im Abschnitt 2.1.3 diskutiert. Sie bilden ein wesentliches Moment eines soziologischen Verständnisses von Softwareentwicklung.

2.1.2.6 Test

Am Ende der Implementierungsphase ist aus dem Entwurf eine Sammlung an Teilmodellen entstanden. Diese müssen nun für ein komplettes Modell in Form

des vollständigen Softwarecodes noch zusammengefügt werden. In die Phase des *Tests* (s. Abbildung 2.2, S. 40) fallen die Aktivitäten nach der Implementierung und vor der Fertigstellung der Software. In dieser Phase wird geprüft, ob die Modellierung des sozialen Phänomens für den Computer die Aufgabenstellung des Softwareprojekts erfüllt. Tut sie das nicht, werden Modell oder Aufgabenstellung so lange verändert, bis beides übereinstimmt. Gelingt es nicht, diese Übereinstimmung herzustellen, gilt das Projekt als gescheitert. Klassisch wird beim Test lapidar gesagt festgestellt, ob die Software das tut, was sie tun soll. Dazu müssen zwei Fragen beantwortet werden, die zumindest theoretisch klar voneinander getrennt werden können: Hat die Software Fehler? Und stimmt sie allgemein mit den Erwartungen überein? (Sommerville 2012).[26] Als Fehler gilt dabei etwa, wenn Funktionen nicht ihrer Beschreibung in einem der Modelle entsprechen, z. B. wenn die Software unter bestimmten Bedingungen anders funktioniert, als es in der Spezifikation festgelegt wurde. Fehler bilden damit eine relevante Teilmenge aller unerfüllten Erwartungen, die an die Software gerichtet werden. Andere unerfüllte Erwartungen sind solche, die absichtlich oder unabsichtlich nicht in das Modell aufgenommen worden sind, aber beim Ausführen der Software gleichwohl als Erwartungen adressiert werden. Für diese Restkategorie unerwarteter Funktionen gibt es in der Informatik keinen einheitlichen Begriff. In klassischen Vorgehensmodellen steht stehen das Auffinden und Beheben von Fehlern im Mittelpunkt der Testphase.[27]

Aus soziologischer Perspektive ist nicht nur der Umgang mit Fehlern und anderen unerfüllten Erwartungen interessant, sondern auch die Frage, wie zwischen den beiden Formen unerwarteten Verhaltens unterschieden wird. Beim Test werden die Funktionen der Software wieder und wieder ausgelöst, wobei die Bedingungen systematisch verändert werden. Die Testenden versuchen dabei, Rand- und Extrembedingungen der Anwendung nachzustellen und dadurch Unterschiede zwischen erwarteten und beobachteten Funktionen zu entdecken, die dann, so als relevant und behebbar eingestuft, von den Entwickler*innen beseitigt

[26] Fehlerfreiheit ist bis auf extrem seltene Ausnahmefälle nicht Ziel des Tests, denn eine gewisse Menge an Fehlern gilt als unvermeidlich (Claus und Schwill 2003). Auch ist das Nichtvorhandensein von Fehlern schon bei mäßig komplexen Softwaresystemen nicht nachzuweisen (Myers et al. 2012).

[27] Andere nicht erfüllte Erwartungen werden sehr wohl thematisiert, in den klassischen Vorgehensmodellen aber nicht als Problem in der Testphase, sondern als Problem bei der Anforderungsermittlung. Unerfüllte Erwartungen, die im Test (oder später) adressiert werden, lassen sich so auf Fehler in den Anforderungen zurückführen oder als unvermeidliche Restkategorie aus dem Problembereich der Informatik auslagern.

werden sollen. Es stellt sich dabei nicht nur die Frage, ob die beobachtete Funktion ein Fehler ist, sondern auch, ob der Fehler überhaupt behoben werden muss, damit das Projekt erfolgreich abgeschlossen werden kann. Bei den Verhandlungen darüber, wer was wann und warum verändern muss, um Erwartungen und Funktionen in Einklang zu bringen, können die Unsicherheiten, die in vorangegangenen Phasen bereits bearbeitet worden sind, erneut zum Thema gemacht werden.

In klassischen Vorgehensmodellen wird zwischen verschiedenen Arten von Tests unterschieden, die sich auf unterschiedliche Ebenen des Modells beziehen. Dazu werden die Modellierungsschritte rückwärts nachvollzogen und überprüft, ob die produzierten Codeteile und, nach deren Integration, die gesamte Software die Funktionen zeigt, die sie dem Modell zu Folge zeigen soll (Abbildung 2.3 zeigt diese Logik vereinfacht am Beispiel des V-Modells).

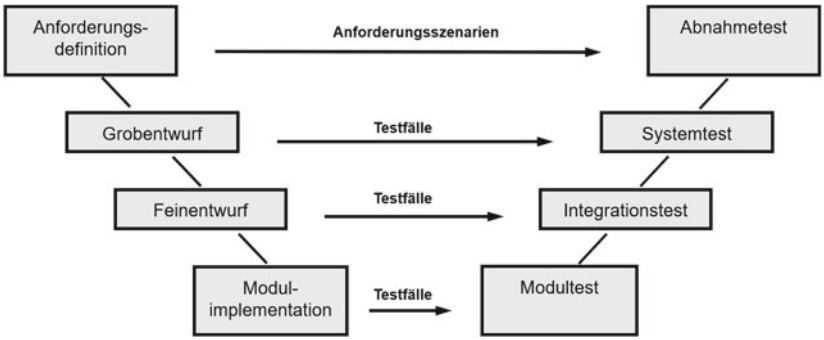

Abbildung 2.3 Beziehung von Modellebenen und Testebenen nach dem V-Modell (Brandt-Pook und Kollmeier 2015, S. 24)

Programmierer*innen oder Softwaretester*innen testen einen großer Teil der Software schon während der Programmierung der Komponenten oder während ihrer Integration. Softwaretester*innen sind so Teil des Entwicklungsteams. Sie haben aber weder den Code verfasst, dessen Funktionen sie überprüfen, noch sind sie Stakeholder*innen der fertigen Software. Letztere kommen beim klassischen Vorgehen erst ganz am Ende der Testphase zum Zug. Die Akteur*innen, die die Software testen, sind also alle vorher in unterschiedlichem Ausmaß an den Prozessen der Entwicklung beteiligt gewesen. Die nun zu testende Software bildet also zunächst die Ergebnisse früherer Modellierungsprozesse ab. Bedeutsam

ist ferner: Fehler zu beheben kostet Zeit, die im Projekt immer knapp ist (Lundin und Söderholm 1995). Der Verweis auf die Zeit kann damit auch strategisch genutzt werden um durchzusetzen, dass statt des Codes die Erwartung angepasst und eine Funktion nicht als Fehler, sondern als akzeptables Ergebnis der Modellierung angenommen wird. Das Modell bleibt gleich, die Vorlage wird verändert, und schon stimmen beide doch noch überein. So kann die bestehende Software als „Zwang des Faktischen" im Nachhinein Kontrolle zwischen den Phasen des Lebenszyklus verschieben: Im Entwicklungsprozess werden Unsicherheiten des Sozialen durch Entscheidungen über die Eigenschaften des Modells kontrolliert. Die Entscheidungen sind das Ergebnis von Aushandlungen, die von den beteiligten Akteur*innen und den Bedingungen abhängen, unter denen verhandelt wird. Wenn eine Entscheidung, die in der Spezifikationsphase getroffen worden ist, in der Testphase in Frage gestellt wird, beginnt die Aushandlung erneut. Die fertig programmierte Software und die verstrichene Zeit bilden dabei neue Bedingungen für die Aushandlung. Selbst wenn die Akteur*innen und ihre Verhandlungsziele gleichbleiben, kann sich dadurch das Kräfteverhältnis so verschieben, dass die Entscheidung nun anders ausfällt als beim ersten Mal. Ein Beispiel dafür ist, wenn Mitglieder der Finanzabteilung, die bei der Erhebung der Anforderungen nicht verhindern konnten, dass eine Funktion nach dem Wunsch der Fachabteilung in das Modell aufgenommen wird, nun im Test keine weiteren Mittel für die Behebung des Fehlers bewilligen und so die unbeliebte Funktion doch noch verhindern. Wenn auf diese Weise eine Entscheidung aus der Spezifikationsphase revidiert wird, so verschiebt sich damit auch die Kontrolle über die Unsicherheit aus der Spezifikations- in die Testphase.

2.1.2.7 Übergang in die Nutzung

Mit dem Abschluss der Testphase endet das Softwareentwicklungsprojekt.[28] Die soziologisch bedeutsamen Fragen lassen sich nun immer weniger an die Phasen des Softwarelebenszyklus anlehnen, weshalb die weitere Betrachtung nun nicht mehr an Hand der Phasen aus Abbildung 2.2 strukturiert wird. Klassischen Vorgehensmodellen zu Folge kann die Software nun in die Nutzung übergehen. Soziologisch ist dagegen auch die Evaluation des Projekts durch die initiierende(n) Organisation(en) bedeutsam. Dabei werden die Funktionen der Software, Dauer und Kosten des Projekts sowie andere Ergebnisse der

[28] Dies ist vereinfacht dargestellt. In der Praxis wird Software häufig inkrementell entwickelt, so dass die Fertigstellung der ersten Version der Software nur einen Meilenstein im Rahmen eines länger bestehenden Projekts bildet. Mit der Übergabe in die Nutzung kommen jedoch grundsätzlich neue Akteur*innen, Umgangsformen und Kontexte hinzu, weshalb diese Vereinfachung mir hier zulässig erscheint.

Projektarbeit (z. B. die Dokumentation der Software oder eventuelle Schulungsunterlagen) begutachtet, bewertet und mit den vor Beginn des Projekts getroffenen Festlegungen verglichen. Genau genommen sind die Grenzen zwischen Softwaretest und Evaluation des Softwareentwicklungsprojekts fließend: Die am Prozess der Softwareentwicklung beteiligten Akteur*innen antizipieren während der Testphase, welche Auswirkungen ihre Handlungen auf die Evaluation des Projekts haben. Die nicht am Projekt, aber an der Evaluation beteiligten externen Stakeholder*innen antizipieren, wie und in welcher Qualität ihre Anforderungen im Projekt aufgenommen worden sind – oder entwickeln zumindest diesbezüglich Erwartungen. Dies kann – wie üblich – strategisch genutzt werden, wenn Akteur*innen die Unsicherheit über die zukünftige Evaluation des Gesamtprojekts ausnutzen, um eine Einigung zum Umgang mit Fehlern herbeizuführen.

Entscheidend ist nun: Für Akteur*innen außerhalb des Softwareprojekts sind die Modelle, auf denen die Funktionen beruhen, in der Regel nicht zugänglich. Software wird als ausführbarer Code ausgeliefert, der veränderbare Quellcode verbleibt bei der Herstellerin bzw. beim (Nachfolge)Projekt.[29] Mit dem Übergang in die Nutzung entsteht eine neue Form des Umgangs mit Software: Auf der einen Seite besteht Software als Objekt der Modellierung in den Prozessen der Weiterentwicklung fort. Auf der anderen Seite wird Software als Trägerin von Funktionen Teil von sozialen Prozessen, in denen diese Modellierungsprozesse und die dabei ausgeblendeten Unsicherheiten nicht oder nur sehr eingeschränkt nachvollzogen werden können. Die Performativität der Software, die sich mit jedem Schritt der Softwareentwicklung weiter spezifiziert und in Ansätzen vor allem bereits in der Testphase aufscheint, tritt in diesen Prozessen in den Vordergrund; dies geschieht nun allerding im Kontext eines neuen Sozialsystems, nämlich der anwendenden Organisation.

Damit die neu entstandene Software genutzt werden kann, muss sie nun in ein anderes sozio-technisches System integriert werden, welches im Systemmodell antizipierend abgebildet worden ist. Im Softwarelebenszyklus wird diese Phase als *Installation* bezeichnet. Damit wird betont, dass die zentrale Aktivität für Informatiker*innen die Einbindung der ausführbaren Software in die technische Infrastruktur des Nutzungskontexts ist. Veränderungen am Code der Software – vor allem grundlegender Art – sind in dieser Phase nicht mehr vorgesehen. Eine

[29] Dies gilt nicht für Open Source Software, bei der der Quellcode frei zugänglich ist und abhängig vom Lizenzierungsmodell auch verändert werden kann. Einige der hier behandelten Fragen stellen sich bei Open Source Software nicht oder anders. Ein Vergleich würde den Rahmen dieser Arbeit sprengen, auch wenn er sich gerade aus soziologischer Perspektive anbietet.

Ausnahme stellt das unten diskutierte *Customizing* dar. Aus Sicht der Informatik müssen für die Einbindung in den Nutzungskontext lediglich die Daten, deren Verarbeitung im Code beschrieben ist, hinzugefügt oder deren Werte verändert werden, wenn diese noch auf den Kontext der Softwareentwicklung verweisen (Balzert 2011). Einige dieser Daten (z. B. Benutzernamen, Hardwareinformationen etc.) dienen primär der Integration und werden in der Regel nur im Rahmen von Installationen und Betrieb verändert. Andere (und deutlich mehr) werden von der Software verarbeitet und erzeugen ihre Funktionen. Wie, von wem und im Rahmen welcher sozialen Prozesse diese Daten entstehen, hängt von vielem ab, nicht zuletzt von der Art der Software, den Kompetenzen der zukünftigen Nutzer*innen und der Art, wie der Nutzungskontext organisiert ist.[30] Die Fragen, die sich daraus ergeben, werden hier nicht allgemein ausdifferenziert.[31] Ich konzentriere mich hier auf Software, die für Organisationen entwickelt ist und in Organisationen eingebunden werden soll.

2.1.2.8 Einführung in sozio-technische Systeme

Bei der Einführung von Organisationssoftware in Organisationen werden Vorgaben dafür entwickelt, wie im Alltag der Organisation mit ihr umzugehen ist. Nach den Vorstellungen, die die klassische Softwareentwicklung prägen, soll durch die Integration von Organisationssoftware in Organisationen aus der bestehenden Organisation ein neues sozio-technisches System entstehen (vgl. die Ausführungen zum Systemmodell in 2.1.2.3). Aus dieser Perspektive sind die Vorgaben nötig um sicherzustellen, dass das neue System möglichst mit dem Systemmodell übereinstimmt und die Software damit den Zweck erfüllen kann, der zu Beginn der Entwicklung festgelegt wurde. Solche Vorgaben sind Modelle der Praktiken der Organisation in Form von Texten in natürlicher Sprache (z. B. in Handbüchern) und Bildern (z. B. Diagramme, Organigramme). Sie beschreiben, welche Tätigkeiten die Nutzer*innen innerhalb der neuen Praktiken ausführen sollen und

[30] Welche Akteur*innengruppen an der Integration beteiligt und welche Prozesse damit verbunden sind, hängt grundsätzlich davon ab, ob die Software von Organisationen betrieben wird oder von Individuen, ob sie in Organisationen betrieben wird oder nicht, und ob sie technisch verändert werden muss, bevor sie eingesetzt werden kann. Im ersten Fall sind in der Regel Informatiker*innen, meist Administrator*innen, für Installation, Betrieb und Wartung verantwortlich. Dies gilt auch für den Fall von Anwendungsprogrammen im Internet, die Anbieter*innen von Onlinediensten selbst entwickeln und betreiben. Bei Software, die für Individualnutzer konzipiert ist, ist die Frage, welche Akteur*innen und Prozesse Integration, Betrieb und Wartung der Software charakterisieren, stark abhängig von den individuellen Kompetenzen und Beziehungen der Nutzer*innen.

[31] Ähnliche Fragen stellen sich bei der Kombination von Softwareelementen, welche im folgenden Abschnitt diskutiert wird.

welche durch Funktionen der Software ersetzt (oder im Vergleich zu den alten Praktiken ergänzt) werden. Soziologisch können die Prozesse, in denen diese Vorgaben entstehen, wie andere Prozesse der Regelbildung in Organisationen als Aushandlungen betrachtet werden, bei denen einige Akteur*innen innerhalb der Organisation versuchen, ihre Kontrolle über die Praktiken der Organisation in formalen Regeln festzuschreiben (Ortmann et al. 1990). Software wirkt in solchen Prozessen insofern performativ, als die Modelle im Code festlegen, welche Funktionen unter welchen Umständen erwartet werden können, welche Eingabedaten diese Funktionen verlangen und welche Ausgaben sie produzieren. Damit wird vorstrukturiert, welche Tätigkeiten durch Software ersetzt oder ergänzt werden können (und welche nicht), welche Art von Daten die Nutzer*innen eingeben müssen, wenn die Software erwartungsgemäß funktionieren soll (und welche nicht), und welche Ergebnisse innerhalb der Organisation weiterbearbeitet werden können (und welche nicht). Die Gegenstände, über die bei der Modellierung der Praktiken zum Umgang mit der Software verhandelt werden kann, sind damit durch die sozialen Prozesse bei der Modellierung der Software vorstrukturiert, sobald die Entscheidung für die Einführung dieser Software gefallen ist. Ob die Kontrolle des Nutzer*innenverhaltens gelingt, ist damit noch nicht entschieden.

Bevor ich mich der Softwarenutzung zuwende, werde ich im folgenden Abschnitt standardisierte Organisationssoftware vorstellen und erläutern, welche Besonderheiten diese Software im Hinblick auf Entwicklung und Integration aufweist. Diese Besonderheiten sind für die vorliegende Arbeit wichtig, da die standardisierte Organisationssoftware SAP Gegenstand der empirischen Analyse in Kapitel 4 ist. Für ein Verständnis dieser Analyse ist Grundwissen über die spezifischen Prozesse bei der Entwicklung und Integration von standardisierter Organisationssoftware daher unabdingbar. Besonders die Integration dieser Form der Software, das sogenannte Customizing, muss eigens erläutert werden.

2.1.2.9 Sonderfall standardisierte Organisationssoftware

In der Informatik wird standardisierte Organisationssoftware als Werkzeug betrachtet, welches Organisationen einsetzen, um ihre Arbeitsprozesse zu optimieren (folgende Zusammenfassung dieser Perspektive nach Rolf et al. 1998). Die Funktionen standardisierter Organisationssoftware sollen Organisationen in die Lage versetzen, die informationstechnische Bearbeitung möglichst aller betriebswirtschaftlich relevanten Vorgänge zu vereinheitlichen und nebenbei sicherstellen, dass diese Vorgänge unterschiedlichen regulatorischen Anforderungen genügen. Standardisierte Organisationssoftware ersetzt einzelne Programme für unterschiedliche Aufgaben (z. B. Lagerhaltung, Produktionsplanung, Rechnungslegung, Personalmanagement etc.) durch spezialisierte Module, die die

gleichen Aufgaben erfüllen, aber über eine gemeinsame Datenbank, aufeinander abgestimmte Funktionen und ein einheitliches Erscheinungsbild verfügen. Die Software modelliert eine informationstechnisch integrierte Organisation: Informationen, die in den verschiedenen Bereichen der Organisation produziert werden, können als einheitlich formatierte Eingaben zu Daten der Software werden. Eine Anforderung an alle Komponenten der Software ist, dass die Daten aus verschiedenen Arbeitsprozessen ohne Eingriff der Nutzer*innen von einer Komponente an die andere übergeben werden können. Soziologisch ist vor allem bedeutsam, wie sich die Rahmenbedingungen, Akteur*innen und Prozesse bei der Entwicklung, Integration und Nutzung von standardisierter Organisationssoftware von den bisher beschriebenen unterscheiden. Die grundlegende Aufgabe des Softwareentwicklungsprozesses, Unsicherheit aus dem Sozialen durch Entscheidungen über Eigenschaften des Modells zu kontrollieren, bleibt bestehen. Hinzu kommt aber noch die Anforderung, dabei ein Modell zu konstruieren, welches eine große Bandbreite an konkreten sozio-technischen Systemen möglichst genau abbildet. Diese Anforderung erhöht die Unsicherheit aller Modellierungsschritte deutlich, ebenso wie die Anzahl der (potentiell) beteiligten Akteur*innen und sozialen Zusammenhänge, relevanten Rahmenbedingungen und Aushandlungen.

Besonderheiten der Softwareentwicklung

Die Modellierung standardisierter Organisationssoftware wird primär von einer Softwareherstellerin getragen, die als Initiatorin des Entwicklungsprojekts entscheidet, auf welche Organisationen die Funktionen ausgerichtet sein sollen. Im Rahmen der Initiation des Entwicklungsprojekts wirbt die Herstellerin bestimmte Organisationen, sogenannte Pilotorganisationen, an, die Mitarbeiter*innen entsenden, die als zusätzliche Stakeholder*innen an den Modellierungsprozessen beteiligt werden. In der Logik der klassischen Softwareentwicklung ergänzen die Repräsentant*innen der Pilotorganisationen das Wissen der Softwareherstellerin über typische Anforderungen der Organisationen im anvisierten Markt durch eigene Erfahrungen (Pollock und Williams 2009; Scott und Kaindl 2000). Soziologisch stellt sich die Zusammenarbeit von Softwareherstellerin und Pilotorganisationen weniger eindeutig dar: Softwareherstellerinnen müssen einen bestimmten Ausschnitt des Markts als Zielgruppe auswählen, mögliche Pilotorganisationen identifizieren und sie überzeugen, eigene Mitarbeiter*innen für die Beteiligung am Entwicklungsprojekt bereitzustellen. Die Kooperation muss in beiden Organisationen durchgesetzt, ihr Zweck bestimmt und die Zusammenarbeit während der Entwicklungszeit aufrechterhalten werden. Das Entwicklungsprojekt muss gegenüber Pilotorganisationen und Softwareherstellerin unterschiedlich legitimiert werden: In ersteren soll die Software nach ihrer Fertigstellung angewandt

werden, während letztere sie verkaufen will. Für Pilotorganisationen lohnt sich die Beteiligung an der Kooperation daher vor allem, wenn dadurch möglichst viele ihrer konkreten Anforderungen in der entstehenden Software berücksichtigt werden. Für die Softwareherstellerin geht es in der Kooperation hingegen darum, möglichst wenig auf die spezifischen Unterschiede zwischen den Pilotorganisationen einzugehen, sondern herauszufinden, wie sich die konkreten Anforderungen in ein generisches Modell übertragen lassen, welches in möglichst vielen unterschiedlichen Organisationen eingesetzt werden kann (Pollock et al. 2007). Diese unterschiedlichen Ziele führen dazu, dass konkrete Anforderungen einzelner Pilotorganisationen und der Wunsch der Softwareherstellerin, mit dem Produkt einen möglichst großen Markt zu erreichen, während des Modellierungsprozesses in systematischem Konflikt stehen (a.a.O.). Gelingt die Modellierung trotz dieser Schwierigkeiten, so entsteht eine generische Grundform der Software. Diese enthält Vorlagen für Organisationsstrukturen, Arbeitsabläufe, Berechtigungen und ähnliches, die vor der Inbetriebnahme konkretisiert werden müssen, indem Daten über den Nutzungskontext (z. B. die Namen der Mitarbeitenden, Bezeichnungen der Abteilungen etc.) eingegeben und bestimmte Modellelemente (z. B. die für die Organisation relevante Variante der Bilanzierung) aktiviert werden (Davenport 1998). Erst nach dieser Konkretisierung kann die Software in einer konkreten Organisation funktionieren.

Besonderheiten der Integration: Customizing

Diese Konkretisierung der generischen Grundform der Software findet beim Customizing statt. Das Customizing kommt nicht als Phase im allgemeinen Softwarelebenszyklus vor, da es nur für standardisierte Organisationssoftware relevant ist. Customizing kann als eine eigene Form der Modellierung von Software verstanden werden, die den Akteur*innen weniger Kontrolle als Softwareentwicklung, aber mehr als reine Einführung ermöglicht. Der Modellierungsprozess beim Customizing lässt sich insofern mit dem bei der Softwareentwicklung vergleichen, da auch hier ein Modell des sozio-technischen Systems (der Organisation, die die Standardsoftware nutzen will) erstellt und in Software abgebildet werden soll. Customizing wird ebenso im Rahmen eines Projekts ausgeführt, das von der Organisation, in der die Software genutzt werden soll, sowie mindestens einer weiteren Organisation getragen wird. In der Regel ist dies ein IT-Beratungsunternehmen, das auf das Customizing des ausgewählten Standardprodukts spezialisiert ist, oder eine Beratungsabteilung der Softwareherstellerin selbst. Eine der Aufgaben (Lundin und Söderholm 1995) des Customizingprojekts ist es, aus der standardisierten Organisationssoftware in ihrer nicht ausführbaren generischen Grundform eine konkrete Version der Software zu schaffen, die in

jener Organisation zum Einsatz kommt, die am Customizing beteiligt ist. Nach Angaben von Herstellerinnen wie SAP geht es im Customizing darum, die Software an die Organisation anzupassen (Boeder und Groene 2014). Tatsächlich geht es beim Customizing jedoch eher darum festzulegen, wann Eigenschaften des Modells in der Software an die entsprechenden Eigenschaften der Organisation angepasst werden und was umgekehrt an der Organisation verändert werden soll, damit diese mehr dem Modell in der Software entspricht. Aus informationstechnischer Sicht handelt es sich dabei um eine Gratwanderung:

> *„Die Übernahme der angebotenen Funktionen und Prozesse ist softwaretechnisch zwar die einfachere Lösung, sie ist allerdings häufig mit organisatorischen Verwerfungen und Unruhe bei den Beschäftigten verbunden. Anpassungen, die zu stark auf die Betriebshistorie abstellen, verwässern in der Regel das Prozeßmodell und können softwaretechnisch aufwendig werden." (Rolf et al. 1998, S. 112)*

Soziologisch können bei der Integration und beim Customizing selbst die gleichen Prozesse wie bei der Softwareentwicklung beobachtet werden. Die Frage, welche Eigenschaften der Softwaremodelle übernommen werden und wann stattdessen Eigenschaften der Organisation verändert werden sollen, ist Gegenstand vielfacher Aushandlungen. Da standardisierte Organisationssoftware in der Regel in mehreren (oder allen) Bereichen einer Organisation zum Einsatz kommen soll, treffen in diesen Aushandlungen Akteur*innen mit sehr unterschiedlichen sozialen Einbettungen, Zielen und Machtressourcen aufeinander. Die komplexe und in der Regel zu hohen Kosten eingekaufte Software beeinflusst den Ausgang dieser Aushandlungen erheblich (Pollock und Cornford 2004). Der zentrale Unterschied zwischen Customizingprojekt und allgemeinem Softwareentwicklungsprojekt liegt darin, dass das Customizingprojekt unauflösbar an eine spezifische Software gebunden ist. Deren Modell bietet bestimmte Elemente zur Konkretisierung an (z. B. Art der Organisationseinheiten) und beinhaltet Optionen, zwischen denen im Customizingprozess ausgewählt werden kann (z. B. Abteilung, Arbeitsgruppe oder Team, vgl. SAP AG 2001, D8.16.2.1, S. 15 f.). Andere Elemente des Modells sind als Standardoptionen gesetzt und können, müssen aber nicht bearbeitet werden, damit die Software ausgeführt werden kann. Wieder andere Elemente sind nicht zur Bearbeitung vorgesehen. Um sie zu ändern, muss der Code bearbeitet werden. Selbst wenn die Akteur*innen im Customizing Zugriff auf eine so bearbeitbare Version der Software haben, besitzen sie keine vollständige Beschreibung der Modelle, die den einzelnen Codestellen unterliegen, und ihrer Abhängigkeiten zu einander. Diese Beschreibungen sind geistiges Eigentum der Softwareherstellerin und werden in aller Regel als Teil des Geschäftsgeheimnisses behandelt. Veränderungen im Code

von Standardsoftware bringen daher das Risiko mit sich, auch unvorhersehbare Veränderungen an den Funktionen der Software zu verursachen (Light 2001).

Die Ziele und Handlungsoptionen der Akteur*innen beim Customizing werden in vielfacher Hinsicht von ihrer Einbindung in die Organisationen vorgeformt, die sie in das Projekt geschickt haben. Repräsentant*innen der Kundenorganisation treten als Expert*innen für das sozio-technische System auf, unabhängig davon, mit welchen Ausschnitten dieses Systems sie faktisch Erfahrung haben und inwieweit ihre Ziele bei der Modellierung mit den Zielen der Akteur*innen übereinstimmen, die mit der Software später im Alltag Umgang haben werden (Wagner et al. 2006). Auf die gleiche Weise können Berater*innen sich auf überlegene Kenntnis der Modelle in der Standardsoftware sowie Erfahrungen mit vergangenen Customizingprojekten berufen, unabhängig davon, wie akkurat ihr Wissen über die Software oder wie vergleichbar die vergangenen Prozesse mit den aktuellen sein mögen (Mormann 2016). Zukünftige Nutzer*innen können bestimmte Veränderungen mit Verweis auf die Praxis einfordern, Vertreter*innen der Organisationsleitung können andere mit Verweis auf die vorgesehenen Optionen, die Kosten einer Abweichung von Standardoptionen (Benders et al. 2006) oder die Risiken von Codeveränderungen (Light 2001) ablehnen. Die Vorgaben der generischen Form der Standardsoftware dazu, welche Modellelemente wie konkretisiert werden können, strukturieren die Handlungsspielräume der Akteur*innen in diesen Prozessen (Pollock und Cornford 2004). Sie verbinden verschiedene Modellierungsoptionen mit unterschiedlichen Aufwänden und Risiken und definieren, welche Akteur*innengruppen bei welchen Fragen als Entscheider*innen vorgesehen sind. Auch wenn die Akteur*innen im Customizingprozess von diesen Vorgaben abweichen können, müssen solche Abweichungen aufwändig legitimiert werden. Insgesamt wird es dadurch wahrscheinlich, dass die Modelle des sozio-technischen Systems, die im Customizingprozess entstehen, den Modellen der Organisation ähneln, die in der Standardsoftware angelegt sind (Gosain 2004). Die Standardsoftware wirkt damit im Customizingprozess performativ.

Bei der soziologischen Untersuchung von Customizingprojekten stellen sich einerseits Fragen dazu, wie die Akteur*innen im Projekt mit den ihnen von ihren Organisationen zugewiesenen Rollen und mit den von der Software vorgesehenen Handlungsspielräumen umgehen. Wie gehen sie in den Modellierungsprozessen im Projekt und in den Aushandlungen im Außenverhältnis vor, um ihre eigenen Vorstellungen davon, wie Organisation und Software aneinander angepasst werden sollen, durchzusetzen? Andererseits ist zu untersuchen, in welcher Phase und unter Mitwirkung welcher Akteur*innen im Softwareentwicklungsprozess die Festlegungen getroffen werden, die den Akteur*innen im Customizing als

unveränderliche Eigenschaften der Software gegenübertreten. Über diese Eigenschaften, also z. B. die Optionen für mögliche Einheiten der Organisation, werden Unsicherheiten, die die konkrete Organisation betreffen, aus dem Customizingprozess in die Softwareentwicklung verlagert. Damit verschiebt sich auch ein Teil der Kontrolle über diese Unsicherheiten hin zu den sozialen Prozessen, in denen Softwarehersteller*in, Pilotorganisationen und andere Akteur*innen Einfluss auf die Entwicklung der Standardsoftware nehmen.

2.1.2.10 Softwarenutzung

Im Softwarelebenszyklus (s. Abbildung 2.2, S. 40) folgt auf die Installation der Software der *Betrieb*, also eine Phase der technischen Betreuung der Software im veränderten sozio-technischen System.[32] Soziologisch stellt jedoch die *Nutzung* den relevanteren Untersuchungsgegenstand in dieser Phase dar. Bei der Nutzung ist die Software technisch und organisatorisch in das sozio-technische System integriert, in das sie in der Phase der Integration eingeführt worden ist. Sie kann nun zum Einsatz kommen. Ob dies tatsächlich geschieht, also ob sich an die Integration eine Phase der Nutzung oder der Nicht-Nutzung anschließt, ist – wie schon vieles in vorangegangenen Phasen – unsicher. Es muss ausgehandelt werden, ob und wie die neue Software in bestehende Prozesse eingebaut wird. Bei diesen Aushandlungen treffen die Modelle eines sozialen Phänomens, die im Laufe des Entwicklungsprozesses geschaffen worden sind, nun zum ersten Mal auf eine reale Version des Phänomens.[33] Diese Modelle sind in den Datenstrukturen und Funktionen des Softwarecodes enthalten. Auch die Anleitungen und Vorgaben für die Nutzung, die parallel zur Softwareentwicklung und während der Phase der Integration geschaffen worden sind, stellen Modelle der Anwendung dar. Der Code (und die Vorgaben zur Nutzung der Software) kommen nun innerhalb des Phänomens zur Anwendung, welches sie beschreiben. Nach der Definition der Performativität, die zu Beginn von Abschnitt 2.1. erläutert wurde,

[32] Als zweite definierende Aktivität in dieser Phase wird häufig die Wartung genannt. Betrieb und Wartung werfen aus soziologischer Perspektive die gleichen Fragen auf wie Weiterentwicklung und Wiederverwendung und werden daher in dieser Arbeit nicht eigens behandelt. Diese Betrachtung ist zumindest anschlussfähig an die Informatik: Sommerville (2012) fasst alle Aktivitäten nach der Übergabe in die Nutzung unter dem Begriff Evolution zusammen.

[33] Selbst in Fällen, in denen Software speziell für einen bestimmten Kontext entwickelt worden ist, liegen zwischen dem Original (dem Anwendungskontext) und dem fertigen Modell (dem Softwarecode) all die Vereinfachungen der Modellierung, die der Softwareentwicklungsprozess mit sich bringt und die in den vergangenen Abschnitten beschrieben worden sind. Meist kommt jedoch gekaufte Software zum Einsatz. Vorlage für die Modellierung ist in diesem Fall entweder ein konkreter anderer oder ein vollständig ausgedachter Anwendungskontext.

wird die Software dann in den sozialen Prozessen, die dieses Phänomen auszeichnen, performativ, wenn diese durch die Nutzung den Modellen ähnlicher werden, durch die sie in der Software beschrieben sind. Dass Software die sozialen Prozesse, in denen sie eingesetzt wird, zumindest in gewissem Ausmaß auf diese Art beeinflusst, dürfte weder in der Soziologie noch in der Informatik ein überraschendes Untersuchungsergebnis darstellen. Interessanter ist, wie diese Veränderungen mit den sozialen Prozessen zusammenhängen, in denen die Software entwickelt und eingeführt wurde, und mit den sozialen Bedingungen, unter denen die Nutzung stattfindet.

Für Informatiker*innen steht bei der Nutzung die Frage im Mittelpunkt, wie gut die Software ihren Zweck erfüllt. In einem guten Modellierungsprozess sind hinreichend viele der Anforderungen, die auf Basis der Zweckbestimmung erhoben worden sind, hinreichend genau in Code übersetzt worden. Durch eine erfolgreiche Integration sind die Nutzer*innen soweit auf die notwendigen Veränderungen in ihren Praktiken eingestimmt worden, dass sie so mit der Software umgehen, dass der Zweck, der zu Beginn des Modellierungsprozesses festgelegt worden ist, hinreichend gut erreicht werden kann. Was genau „hinreichend" bedeutet, wird vor, während und nach dem Softwareprojekt in immer neuen Akteur*innenkonstellationen und unter sich ständig verändernden Rahmenbedingungen ausgehandelt. Diese Aushandlungen bestimmen sowohl, wie die Software am Ende aussieht, als auch, welche Erwartungen bei der Nutzung an sie gerichtet werden. Diese Aushandlungen, die Softwareentwicklung für Soziolog*innen so interessant machen, werden in der Informatik kaum thematisiert. In einem aber sind sich Vertreter*innen beider Disziplinen einig: Keine Software kann alle Erwartungen erfüllen. Soziolog*innen können dies mit unterschiedlichen Standpunkten und widersprüchlichen Interessen, mehrdeutigen Interpretationen und sich wandelnden Ansichten begründen, also kurz: indem sie auf die Unsicherheiten des Sozialen verweisen. Informatiker*innen gehen davon aus, dass es immer Anforderungen gibt, die im Modellierungsprozess nicht aufgenommen oder darin falsch abgebildet werden, und dass andere Anforderungen erst durch den Umgang mit der Software zu Tage treten. Auch berücksichtigen sie, dass sich das soziotechnische System oder sein Umfeld im Laufe der Zeit verändern und dadurch neue Anforderungen entstehen oder die Software einen neuen Zweck erfüllen soll (Sommerville 2012). Unerkannte und sich verändernde Anforderungen werden in klassischen Vorgehensmodellen als Problem aufgenommen, das vielleicht nicht vollständig aufgelöst, aber dessen Lösung zumindest angestrebt werden soll. Aus der Praxis der Softwareentwicklung heraus haben sich dazu Ergänzungen entwickelt: Methoden, die die Verständlichkeit der Software für die Nutzenden und das Verständnis der Softwareentwickler*innen für die modellierten Gegenstände

verbessern sollen.[34] Diese Methoden nehmen auch einige der Fragen auf, die in dieser Arbeit als soziologische Fragen vorgestellt worden sind. Besonders nah an der Thematik dieser Arbeit liegt die sogenannte agile Softwareentwicklung, eine Sammlung von Ansätzen, die auf der Annahme aufbauen, dass Unsicherheit nicht dauerhaft reduziert werden kann. Diese Ansätze verbreiten sich seit Mitte der 2000er Jahre immer stärker in der Informatik (Hoda et al. 2018). Bei großen Softwareprojekten, wie sie für die Entwicklung standardisierter Organisationssoftware nötig sind, kommen sie jedoch erst seit kurzem und keineswegs flächendeckend zum Einsatz. Für die Fragen der vorliegenden Arbeit ist das Verständnis der klassischen Vorgehensmodelle daher deutlich relevanter. Aus diesem Grund stelle ich die Grundideen der agilen Softwareentwicklung hier nur in einem Exkurs vor.

Exkurs: Agile Softwareentwicklung

In der *agilen Softwareentwicklung* stehen ein früher Kontakt der Nutzer*innen mit der Software und ständige Weiterentwicklung im Mittelpunkt. Der agilen Perspektive liegt die Annahme zu Grunde, dass Softwareentwicklung sich mit veränderbaren Gegenständen befasst und sich ihren Zielen grundsätzlich nur annähern kann. Damit stehen agile Ansätze im Gegensatz zur rationalistischen Tradition der Softwareentwicklung, die auf der Annahme basiert, dass die Unsicherheiten des zu modellierenden Phänomens durch ausgiebige Planung und das Befolgen strukturierter Prozessvorgaben systematisch reduziert werden können. Informatiker*innen, die die agile Perspektive einnehmen, lehnen das Konzept grundsätzlich getrennter Modellierungsphasen mit davon abgeleiteten Plänen, Aktivitäten und Rollenverteilungen ab und setzen auf selbstorganisierte Teams, aus deren Zusammenarbeit gute Software „emergieren“ soll (Beck et al. 2001). Zentral für agile Ansätze sind daher möglichst kurze Entwicklungszyklen, um Funktionen der Software möglichst früh in die Nutzung zu

[34] Mit einer verbesserten Gestaltung der Schnittstellen zwischen Mensch und Computer befasst sich vor allem das Usability-Engineering (z. B. Nielsen 1993), mit den soziotechnischen Systemen beschäftigt sich primär die Wirtschaftsinformatik. Von dieser werden Organisationen in der Regel als Systeme betrachtet, die vom Management durch die Wahl passender Strategien gesteuert werden können. Charakteristisch dafür sind Studien aus dem Bereich Information Systems Research, bei denen kritische Faktoren identifiziert werden, welche Projektleiter beachten müssen, um sicherzustellen, dass die Einführung eines ERP-Systems in ihre Organisation zu einer im Sinne des Managements erfolgreichen Nutzung führt (z. B. Holland und Light 1999).

bringen und damit herauszufinden, inwieweit sie die intendierten Zwecke erfüllen. Die Beteiligung von Stakeholder*innen beschränkt sich nicht auf frühe und späte Phasen des Entwicklungsprozesses wie in klassischen Vorgehensmodellen. Stattdessen sind sie dauerhaft notwendiger Teil des Entwicklungsteams und sollen, ebenso wie dessen übrige Mitglieder, jederzeit neue oder geänderte Anforderungen an die Software formulieren und sich an der Erarbeitung von Lösungen beteiligen, mit denen diese Anforderungen erfüllt werden können. Die ideale Steuerung agiler Softwareprojekte fokussiert darauf, die Arbeitsumgebung so einzurichten, dass die Teammitglieder motiviert bleiben, Raum für direkte Interaktionen, gemeinsame Entscheidungen und Reflexion und stetige Weiterentwicklung ihrer Arbeitsprozesse haben. Auf diese Weise soll aus der von außen möglichst wenig kontrollierten Interaktion im Team durch ständiges Nachsteuern Software entstehen, die die Zwecke der Nutzung dauerhaft möglichst gut erfüllt.

Agile Ansätze der Softwareentwicklung sensibilisieren die Beteiligten insofern für soziologische Fragen, als sie den Unsicherheiten sozialer Prozesse gezielt bei der Modellierung selbst Raum geben.[35] Trotzdem bleiben auch agile Projekte hochgradig abhängig von den sozialen Bedingungen, unter denen sie stattfinden (Schulz-Schaeffer und Bottel 2018).[36]

Aus soziologischer Perspektive werden bei der Nutzung zwar auch Aspekte betrachtet, die in der soziologisch inspirierten Informatiker*innenperspektive berücksichtigt werden, es werden aber auch Fragen gestellt, die darüber hinaus gehen. Ob Nutzer*innen mit Software so umgehen, wie es in ihren Modellen vorgesehen ist, wird nicht primär als Verständnisproblem thematisiert. Stattdessen steht im Vordergrund, wodurch die Handlungsspielräume beeinflusst werden, die Nutzer*innen beim Umgang mit der Software haben. Dabei entstehen einerseits Fragen nach den Zusammenhängen zwischen der Nutzung und den Prozessen im sozio-technischen System, in der sie stattfindet: Wie handeln Nutzer*innen

[35] Auf Seiten der Forschung kann das in der Einleitung eingeführte Feld der Computer Supported Cooperative Work (CSCW) als sozialwissenschaftlich informierte Informatik bezeichnet werden.

[36] In der zitierten Studie wird zum Beispiel dargestellt, wie ein Entwickler, dessen Bewertung vom Mitarbeiter der Fachabteilung, dem internen Kunden, abhängt, und nicht von seinem direkten Vorgesetzten, dem Leiter des Entwicklungsteams, regelmäßig Qualitätsanforderungen an seine Arbeit ignoriert, um möglichst schnell neue Funktionen zu programmieren, weil die Fachabteilung nur letzteres beurteilen kann. (Schulz-Schaeffer und Bottel 2018).

miteinander neue Umgangsformen mit der Software aus? Wie gehen sie dabei mit den Modellen neuer Praktiken um, die bei der Einführung entwickelt worden sind? Und welche Rolle spielt die Tatsache, dass sie nicht nur Nutzer*innen der Software sind, sondern auch an anderen Prozessen (in- und außerhalb des soziotechnischen Systems) teilnehmen? Andererseits liegen auch Fragen dazu nahe, wie durch die Software soziale Prozesse, die während der Entwicklungsphasen des Softwarelebenszyklus stattfinden, Einfluss auf die Nutzung nehmen: Welche Handlungsspielräume werden den Nutzer*innen in den Modellen der Software zugedacht, wie sind diese zugeschnitten, welche versuchen die Modelle zu versperren? Welche Aspekte der Unsicherheit des Nutzungsprozesses werden bei der Modellierung vorweggenommen, in welchen sozialen Prozessen werden sie auf welche Gewissheiten reduziert und welche Art von Kontrolle wird dadurch aus der Nutzung in die Entwicklung verlagert?

Die Phase des Betriebs und der Nutzung schließt den Softwarelebenszyklus. Das „Leben“ der Software endet damit jedoch nicht: Wie bereits angedeutet, wird Software in aller Regel über einen längeren Zeitraum weiterentwickelt und verwendet, es werden also neue Versionen der Software erstellt, die bereits genutzt wird. Diese Weiterentwicklung wirft eigene soziologische Fragen auf, weil dabei die bestehende Version der Software ein Element der Modellierungsprozesse wird, in denen die neue Version der Software entsteht. Software kann dadurch bei der Weiterentwicklung Performativität entwickeln. Ebenso gilt dies bei der Wiederverwendung von Softwareelementen, wenn in einem Softwareentwicklungsprojekt einzelne Teile des Modells bereits fertig aus anderen Projekten übernommen werden. Beides sind Aspekte der Rekursivität von Software, mit der sich der nächste Abschnitt beschäftigt.

2.1.3 Weiterentwicklung und Wiederverwendung

Die bisherigen Ausführungen haben zwei Charakteristika von Software vorgestellt, die zu ihrer Komplexität beitragen: Das erste Charakteristikum ist die Tatsache, dass Software soziale Phänomene, die voller Unsicherheit sind, also nicht vollständig und eindeutig beschrieben werden können, weil jede Beschreibung von der Interpretation der Beschreibenden abhängt, in mathematische Formalismen übersetzt, die eindeutig und damit frei von Unsicherheit sind. Durch diese Übersetzung ist für Nutzende schwer nachvollziehbar, dass die Möglichkeiten, das Soziale zu interpretieren, durch den Umgang mit Software verändert werden: Alternative Interpretationsmöglichkeiten und nicht in der

Software berücksichtigte Aspekte des Sozialen sind in den mathematischen Formalismen nicht mehr erkennbar. Das zweite sind die Prozesse, in denen Software entsteht und genutzt wird. In jedem dieser Prozesse werden einzelne Aspekte des Sozialen ausgewählt und über soziale Aushandlungen in eindeutige Eigenschaften der Software umgewandelt, die es den Akteur*innen ermöglichen, mit der Software umzugehen. Aspekte der Unsicherheit des Sozialen werden damit aus der Nutzung in die Entwicklung und innerhalb der Entwicklung zwischen den Prozessen verschoben. Welche Unsicherheiten dies sind und wie sie verschoben werden, hängt davon ab, wer sich in den konkreten Aushandlungen der einzelnen Prozesse durchsetzen kann. Selbst wenn Nutzende davon ausgehen, dass der Umgang mit Software Einfluss auf ihre Möglichkeiten hat, Soziales zu interpretieren, wird es durch diese Prozesse schwer nachzuvollziehen, wie, von wem und aus welchen Gründen diese Möglichkeiten beeinflusst werden. Beide Charakteristika können als Schichtungen verstanden werden: Die oberste Schicht, das soziale Phänomen, ist voller Unsicherheiten, voller Alternativen und Handlungsspielräume für die Akteur*innen, die mit Software umgehen. Die unterste Schicht, der Code, ist frei von Unsicherheit,[37] Alternativen und Handlungsspielräume gibt es nicht, weil nicht gehandelt wird, nur Operationen ablaufen.

Das dritte Charakteristikum ist die Rekursivität der Softwareentwicklung, also die Tatsache, dass neue Software aus der Kombination und Veränderung bestehender Software entsteht. Dieses Charakteristikum lässt sich am ehesten als Faltung begreifen: Wenn der Umgang mit Software neue Software produziert, entstehen dadurch auf allen Schichten des Produktes Abhängigkeiten zwischen den Elementen, die selbst geschichtete Unsicherheit beinhalten. Dies geschieht, wenn Software weiterentwickelt oder in andere Software eingebaut wird.

2.1.3.1 Bedingungen und Folgen der Weiterentwicklung

Software wird in der Regel über längere Zeiträume weiterentwickelt. Dabei werden zu unterschiedlichen Zeitpunkten einzelne Codezeilen oder ganze Komponenten ergänzt, entfernt oder gegen andere ausgetauscht, um der Software neue Funktionen zu geben oder sicherzustellen, dass ihre Funktionen unter veränderten Bedingungen erhalten bleiben. Der Weiterentwicklungsprozess enthält alle Aktivitäten des ursprünglichen Entwicklungsprozesses (vgl. Abbildung 2.4).

Die Entwickler*innen agieren jedoch unter anderen Bedingungen, weil sie die bestehende Software verstehen müssen, um zu entscheiden, welche Änderungen am Code nötig sind und welche Auswirkungen diese Änderungen auf

[37] Die Unsicherheit, die die Hardware als physisches Objekt mit sich bringt, wird in der Software ausgeblendet.

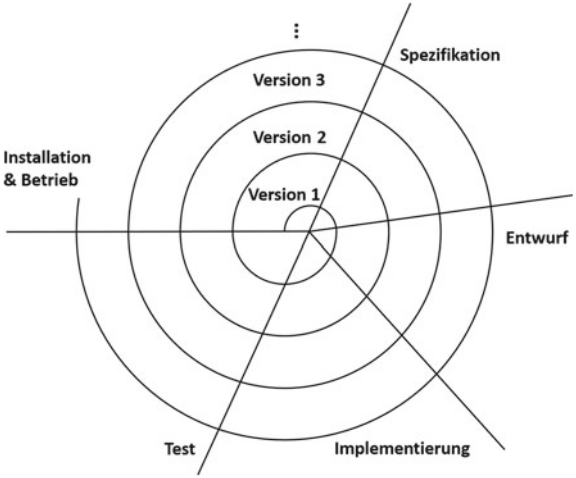

Abbildung 2.4 Software-Lebenszyklus bei der Weiterentwicklung. (Eigene Darstellung)

das Gesamtsystem haben (Sommerville 2012). Anders als beim ursprünglichen Entwicklungsprozess ist es bei der Weiterentwicklung deutlich schwieriger, grundlegende Entscheidungen über zu erfüllende Anforderungen oder Architektur der Software zu verändern, da jede Änderung in ein komplexes Ganzes eingreift, das zudem nicht vollständig bekannt oder transparent ist. Grundlegende Änderungen sind daher mit deutlich größerem zeitlichen und finanziellen Aufwand verbunden als bei Neuentwicklungen. Die Veränderungen müssen sich also in der Regel in bestehende Geflechte von Strukturen einfügen. Software wird durch die Weiterentwicklung daher mit der Zeit komplexer, fehleranfälliger und weniger leistungsfähig. Dies wird als Software-Alterung bezeichnet (Ludewig und Lichter 2013).[38]

Anders als bei der primären Entwicklung ist die Software, die verändert wird, nun selbst Teil eines der sozio-technischen Systeme, die in ihr modelliert sind. Das bedeutet vor allem, dass die Nutzung der Software entscheidenden Anteil

[38] Solche Effekte können vermieden oder zumindest verringert werden, wenn kontinuierlich Aufwand in die Wartung des Codes investiert wird. Solchem Wartungsaufwand steht allerdings kein Gewinn an Funktionalität gegenüber, ihr Nutzen ist kaum berechenbar, weil er vor allem in der Abwendung möglicher zukünftiger Probleme liegt, deren Eintreten und Ausmaß unbekannt ist. Software-Alterung ist damit kein grundsätzliches Phänomen wie biologische Alterung, sondern ein Ergebnis von Kosten-Nutzen-Kalkülen.

an der Weiterentwicklung hat: Probleme der Nutzer*innen, neue Formen der Regulation, Bedingungen der Anwendung und mögliche Lösungen werden in Auseinandersetzung mit vorliegenden Erfahrungen in sozio-technischen Systemen definiert. Daher gilt Weiterentwicklung in inkrementellen Vorgehensmodellen und vor allem in der agilen Softwareentwicklung als der beste Weg zur Modellierung sozialer Phänomene in Code (Sommerville 2012; Beck et al. 2001).

Aus soziologischer Perspektive wirkt Software damit in zweifacher Hinsicht performativ auf die Prozesse der Weiterentwicklung. Weiterentwicklung kann als Modellierungsprozess verstanden werden, bei dem einige Glieder der vorhandenen Kette an Modellen (Spezifikation, Entwurf, Code) so verändert werden sollen, dass sich daraus eine kontrollierte Veränderung des Umgangs mit der Software ergibt. Im Modellierungsprozess gilt es daher, zu verstehen, welche Aspekte des Sozialen bei der Entwicklung der bestehenden Software wie in Eigenschaften des Modells übersetzt worden sind. Daraufhin müssen dann die Unsicherheiten, die aus den neuen Anforderungen entstehen, für die Veränderung des Modells wieder in Eindeutigkeiten umgewandelt werden. Dies alles spielt sich wie bei der initialen Entwicklung im Rahmen von sozialen Prozessen ab. An diesen sind bei der Weiterentwicklung zwar häufig weniger und weniger divers sozial eingebettete Akteur*innen beteiligt als an der ursprünglichen Entwicklung.[39] Oft nehmen aber an der Weiterentwicklung andere Akteur*innen Teil als an der Entwicklung früherer Versionen. Diese neuen Teilnehmer*innen können zwar die Modelle früherer Versionen zum Teil einsehen, können aber in der Regel nicht nachvollziehen, unter welchen Bedingungen welche Unsicherheiten wann und von wem reduziert wurden, um diese Modelle zu erschaffen. Unsicherheiten aus dem Prozess der Weiterentwicklung werden damit in den Prozessen der ursprünglichen Entwicklung kontrolliert. Wechselwirkungen zwischen einzelnen Elementen dieser Modelle, die bei der initialen Entwicklung noch nachvollziehbar sind, werden in den Prozessen der Weiterentwicklung mit der Zeit verdeckt (Schmidt 2012).

Die bestehende Version der Software wirkt bei der Weiterentwicklung aber auch dadurch performativ, dass neue Anforderungen aus den Erfahrungen entstehen, die Akteur*innen mit ihrer Nutzung im sozio-technischen System gemacht haben. Diese Nutzung hängt, wie im vorangegangenen Abschnitt dargestellt, nicht nur von der Software selbst ab, sondern auch von den Aushandlungen, in denen darum gerungen wird, das Verhalten der Nutzer*innen im Umgang mit ihr zu kontrollieren. Die Weiterentwicklung wird dadurch auch von Auseinandersetzungen

[39] Dies gilt vor allem für kleinere Veränderungen, die nur wenige Funktionen der Software betreffen. Ein großer Teil der Aufgaben des Betriebs und der Wartung wird innerhalb von Organisationen zum Beispiel fast ausschließlich von IT-Administrator*innen erledigt, die Anweisungen von Softwareherstellern einfach umsetzen.

zwischen Berater*innen und Organisationsmitgliedern in Customizingprojekten, zwischen Vorgesetzten und Untergebenen, verschiedenen Abteilungen und unterschiedlichen Nutzer*innen darüber beeinflusst, wie die Software in der Organisation am besten genutzt werden sollte. Mit der Weiterentwicklung können Nutzer*innen Kontrolle über Aspekte ihres Umgangs mit der Software zurückgewinnen, die ihnen durch die alte Version der Software entzogen worden sind. Bei der Weiterentwicklung von Standardsoftware stellen sich zusätzliche Fragen: für die Softwareherstellerin zum Beispiel die, wie mit den Anforderungen verschiedener Kund*innen umzugehen ist, für die Kund*innen, ob und wie sie versuchen, Veränderungen in der Software, die nicht von ihnen selbst angestoßen wurden, in die eigenen Praktiken einzubinden.

Bei der Betrachtung der Performativität von Software bei der Weiterentwicklung zeigt sich, dass sich die Komplexität von Software durch die Weiterentwicklung potenziert, weil immer neue Prozesse der Unsicherheitsreduktion stattfinden, die auf den Ergebnissen vergangener Prozesse aufbauen. Unsicherheit wird damit in einem immer dichteren Geflecht sozialer Prozesse weitergereicht, wodurch undurchschaubare Abhängigkeiten zwischen diesen Prozessen entstehen.

2.1.3.2 Wiederverwendung vorgefertigter „Softwarebausteine“

Den gleichen Effekt erzeugt die Wiederverwendung von Softwarebausteinen. Wiederverwendete Softwarebausteine bilden den Hauptteil moderner Software: *„The modern developer no longer builds applications from scratch. Instead, most developers essentially glue different libraries together to perform a task.“* (Wurster und van Oorschot 2008, S. 89). Aus Sicht vieler Informatiker*innen ähnelt die Konstruktion von Software der Konstruktion eines Wolkenkratzers, bei der Gruppen von Spezialist*innen vorgefertigte Bauteile unterschiedlicher Komplexität mit speziell für diesen Bau angefertigten Einzelteilen zu einem neuen Ganzen zusammenfügen.[40] Bei der Softwareentwicklung wird dazu das Modell des sozialen Phänomens spätestens ab der Entwurfsphase in Teilmodelle zerlegt, die theoretisch unabhängig voneinander in Code überführt werden können. Zwischen den Teilmodellen werden Schnittstellen definiert, die sicherstellen sollen, dass sich

[40] Ähnlich wie beim physischen „Bau“ können dadurch Spezialist*innen die Produkte anderer Spezialist*innen in ihre eigene Arbeit integrieren, ohne dass sie verstehen müssten (oder könnten), wie diese Produkte funktionieren. Eine soziologische Reflexion dieses Phänomens findet sich in Abschnitt 3.1.1.; allgemein lässt sich darin auch der für soziale Systeme beschriebene Effekt beobachten, dass Systeme ihre Fähigkeit zur Verarbeitung von Komplexität steigern, indem sie interne Komplexität reduzieren (vgl. Luhmann 2018). Im Beispiel der Software sind komplexere soziale Phänomene modellierbar, indem fertige Bausteine einfach genutzt und damit die Komplexität des Modellierungsprozesses selbst reduziert wird.

die Einzelteile später wieder zu einem Gesamtmodell verbinden lassen. Auf diese Weise entstehen Modellierungsaufgaben, die denen in anderen Softwareprojekten auf einer bestimmten Abstraktionsebene gleichen. Statt solche Aufgaben immer wieder neu zu lösen, können die entsprechenden Modelle aus einem anderen Projekt übernommen werden, wodurch Teile des eigenen Entwicklungsprozesses entfallen. Dadurch sinken Kosten, Risiken und Dauer von Softwareentwicklungsprojekten, es entstehen aber auch zahlreiche Probleme, die gegen die Vorteile abgewogen werden müssen (Sommerville 2012). Bei dieser Abwägung spielen die Auswirkungen, die Wiederverwendungen auf die Weiterentwicklung haben, eine zentrale Rolle: Die wiederverwendeten Elemente werden unabhängig von der eigenen Software weiterentwickelt. Neue Versionen der Elemente müssen in die Software aufgenommen werden, wenn sichergestellt sein soll, dass Fehlerkorrekturen, Lösungen für neu entdeckte Sicherheitsprobleme und ähnliches übernommen werden. Aus der Weiterentwicklung der Elemente entstehen damit Anforderungen für die Weiterentwicklung der eigenen Software.[41]

Soziologisch betrachtet wird bei der Wiederverwendung ein Teil der Entscheidungen über die Eigenschaften der Modelle, auf denen die eigene Software aufbaut, durch eine Entscheidung ersetzt, nämlich durch die für das zu nutzende wiederverwendete Element. Damit wird auch die Kontrolle über die Unsicherheiten, die diese Entscheidungen darstellt, von einem Softwareentwicklungsprozess in einen anderen verschoben. Oft kennen die Akteur*innen des ersten Prozesses diesen anderen nicht, zumindest nicht im Detail. In der Regel können sie keinen Einfluss auf die Aushandlungen ausüben, in welchen festgelegt wird, wie die Unsicherheit des Sozialen bei der Modellierung reduziert werden soll. Die Entwickler*innen, die fertige Bausteine in ihren eigenen Projekten einsetzen, geben damit nicht nur Kontrolle über die Inhalte des aktuellen Modells ab, sondern (zum Teil) auch über die Prozesse der Weiterentwicklung. Damit schaffen sie mit der Wiederverwendung Unsicherheiten für die Zukunft, die zukünftige Entwickler*innen nicht kontrollieren können.

[41] Der dadurch zusätzlich entstehende Wartungsaufwand wird nicht immer aufgebracht (Derr et al. 2017). Fehler und Unzulänglichkeiten der verwendeten Elemente führen zu Fehlern und unerwartetem Verhalten der eigenen Software (vgl. für Bibliotheken z. B. Zibran et al. 2011, Linares-Vásquez et al. 2013). Wollen Softwareentwickler*innen diese Probleme vermeiden, müssen sie solche Abhängigkeiten zwischen Elementen in einem Softwareprojekt daher zuverlässig identifizieren und, falls sie unerwünscht sind, auch beheben. Ansätze zur Identifikation solcher Interdependenzen zwischen Elementen eines Modells gibt es zwar seit langem (z. B. a.a.O., Opdyke 1992 für Fälle, in denen sich die Interdependenzen innerhalb einer Software befinden). Sie kommen in der Praxis jedoch häufig nicht zum Einsatz.

Auch die Produktion wiederverwendbarer Elemente wird im Software Engineering als eigenes Problem behandelt. Solche „Software-Bausteine“ sollen zwar stabil sein (Ludewig und Lichter 2013), müssen aber auch weiterentwickelt werden. Sind die Nutzer*innen der wiederverwendeten Elemente nicht bekannt, stellen *„Ripple effects“* (Yau et al. 1978) Produzent*innen vor Probleme (Robbes et al. 2012). Mit diesem Begriff werden die Auswirkungen bezeichnet, die Veränderungen eines Softwareelements auf andere Elemente haben, die sich in einem gemeinsamen Modell befinden. Soziologisch stellt sich in solchen Fällen die Frage, inwieweit die Produzent*innen der wiederverwendeten Elemente reflektieren, welche Unsicherheiten sie der Kontrolle der Nutzer*innen entziehen, und inwieweit die Reduktionen, bei der Entwicklung der wiederverwendbaren Komponenten vorgenommen worden sind, bei der Nutzung nachvollzogen werden können.

Die grundlegendsten Elemente, die bei der Softwareentwicklung wiederverwendet werden, sind die höheren Programmiersprachen mit ihren Bibliotheken, also Sammlungen von kurzen oder längeren Codebausteinen, die häufig genutzte allgemeine Funktionen erbringen. Elemente von Bibliotheken ermöglichen zum Beispiel die grafische Darstellung von statistischen Daten in Form eines Kreisdiagramms oder eine Eingabeaufforderung in einem Bildschirmfenster. Auch werden sie in aller Regel genutzt, um Daten und Anweisungen mit Ein- und Ausgabegeräten (wie Tastaturen und Bildschirme) oder Datenbanken auszutauschen. Software-Bausteine, die spezifische, aber allgemein anwendbare Funktionen erfüllen, sind in der Regel in so genannten Programmbibliotheken zusammengefasst, die von Softwareentwickler*innen wie Baukästen genutzt werden (Claus und Schwill 2003, S. 603).[42]

Auch technische Standards gehören, soziologisch betrachtet, zu den Bausteinen, die bei der Softwareentwicklung wiederverwendet werden. Zwar sind solche Standards selbst keine Software, aber auch in ihnen werden bestimmte Aspekte eines sozialen Phänomens ausgewählt und festgelegt, wie diese ausgeprägt sein müssen, damit ein bestimmter Zweck erreicht werden kann. Wenn zum Beispiel in einem Standard zur IT-Sicherheit vorgegeben wird, dass Emails mit bestimmten Verfahren verschlüsselt werden müssen, ist dadurch festgelegt, dass der Inhalt der Email ein für die Sicherheit der Kommunikation relevanter Aspekt ist, und nicht etwa Eigenschaften des Computers, von dem die Email geschrieben wird. Standards wie dieser sind abstrakte Modelle eines sozialen Phänomens (z. B. der

[42] Die Nutzung von Systemsoftware kann allgemein als eine Form der Wiederverwendung betrachtet werden: auch wenn Systemsoftware nicht in den Code der Anwendungssoftware übernommen wird, erfüllt sie doch Teilaufgaben, die notwendig sind, um die Funktionen der Anwendungssoftware hervorzurufen.

IT-Sicherheit), die in die Modelle der Softwareentwicklung übernommen werden. Sie reduzieren einige Aspekte der Unsicherheit des Sozialen radikal und beeinflussen dadurch erheblich, wie die Akteur*innen das Phänomen im Softwareentwicklungsprozess modellieren können. Am Beispiel: Durch den Standard müssen Entwickler*innen nicht aushandeln, was genau unter IT-Sicherheit zu verstehen ist, sondern nur, mit welchem der festgelegten Verfahren die Emails von ihrer Software verschlüsselt werden sollen.

Programmiersprachen und technische Standards sind bei der soziologischen Betrachtung der Wiederverwendung Extremfälle: Programmiersprachen tragen sehr wenig zur Reduktion von Unsicherheit des Sozialen bei, weil sie vor allem Modelle für wohldefinierte mathematische Phänomene enthalten, kaum für soziale Phänomene.[43] Die Aufgabe von technischen Standards ist es hingegen, möglichst viele unterschiedliche soziale Phänomene gleichzeitig zu erfassen. Wenn sie erfolgreich sind, verbreiten sie sich weit und können so die Aushandlungen in vielen Folgeprozessen beeinflussen. Die sozialen Prozesse, die zur Entwicklung von Standards führen, sind damit oft komplex und bringen viele Akteur*innen mit sehr unterschiedlichen und oft widersprüchlichen Interessen zusammen. Die Untersuchung von Standardisierungsprozessen ist daher für eine Soziologie der Software ebenso bedeutsam wie die Frage, wie Akteur*innen bei der Softwareentwicklung mit Standards umgehen.

Neben Programmiersprachen, die wenige, und technischen Standards, die viele soziale Phänomene modellieren, existieren verschiedene Arten von Elementen, die in Software wiederverwendet werden. Das in der modernen Softwareentwicklung weit verbreitete Paradigma der Objektorientierung zum Beispiel basiert darauf, dass Programme aus wiederverwendbaren Elementen zusammengebaut werden. Diese sogenannten Objekte fassen Daten und Anweisungen zu deren Bearbeitung zusammen. Bestimmte Eigenschaften und Funktionen dieser Objekte sind bekannt, die genaue Konstruktionsweise bleibt aber allen außer den Konstruierenden verborgen. Diese Kombination von sichtbaren Funktionen und verborgener Funktionsweise, das sogenannte Kapseln von Funktionalität, soll die Wiederverwendung vereinfachen (Herold et al. 2011).[44] Wiederverwendet werden

[43] Compiler, also die Programme, die höhere Programmiersprachen in maschinennähere Sprachen übersetzen, sind allerdings ebenfalls Software, ihre Funktionen sind daher nicht vollständig mit den mathematischen Transformationen gleichzusetzen, die sie modellieren. Die Unsicherheit, die dadurch auch Programmiersprachen enthalten, wird in dieser Arbeit nicht weiterverfolgt, da sie die Argumentation unübersichtlicher machen würde, ohne neue relevante Punkte hinzuzufügen.

[44] Ein Beispiel: Um die Namen der Benutzer*innen eines Programms zu speichern, zu sortieren, zu zählen oder zu löschen, können Programmier*innen ein Listenobjekt benutzen, das

aber nicht nur einfache Objekte oder Bausteine mit wohldefinierten Funktionen wie die grafische Darstellung statistischer Daten. Auch strukturierte Kombinationen solcher Bausteine mit eigener Funktionslogik für einen Anwendungsbereich, je nach Ausprägung Frameworks oder Referenzarchitekturen genannt (Ludewig und Lichter 2013), und komplexe Komponenten, die eigenständig spezialisierte Aufgaben (wie Terminplanung, Verwaltung von Kundendaten) erfüllen können, werden bei der Entwicklung neuer Software genutzt (a.a.O., S. 604 ff.).

Da Wiederverwendung ein Grundprinzip aller Softwareentwicklungsprozesse ist, stellen die Aushandlungen rund um diesen Aspekt gute Ansatzpunkte für die soziologische Untersuchung der Software im Sozialen dar. Die hier dargelegten Fragen können dabei helfen zu rekonstruieren, welche Aspekte der Unsicherheit aus dem Nutzungskontext in welchen sozialen Prozessen wie reduziert worden sind, unabhängig davon, ob dies bei der Entwicklung der genutzten Software direkt oder bei der Entwicklung eines ihrer Elemente geschehen ist. Auf der anderen Seite wird durch diese Fragen auch ein Licht auf die Aushandlungen im Softwareentwicklungsprozess geworfen, die durch die Wiederverwendung angestoßen werden: Vor allem bei der Wiederverwendung komplexerer Elemente orientiert sich bereits der Entwurf an deren Eigenschaften; auch die Anforderungen werden zumindest zum Teil daran angepasst (Sommerville 2012). Was für welche Zwecke neu programmiert und was wozu wiederverwendet wird, ist damit von Beginn an ein relevanter Faktor in den sozialen Prozessen, in denen Zweck, Kosten und Bedingungen der Softwareentwicklung festgelegt werden. In diesen Aushandlungen geht es nicht nur um die „offiziellen" Rahmenbedingungen der Softwareentwicklung, sondern auch darum, welche Handlungsmöglichkeiten sich die Akteur*innen jeweils erhalten wollen und welche Formen von Kontrolle sie dafür bereit sind aufzugeben.

Besonders bedeutsam ist die soziologische Auseinandersetzung mit den Folgen der Wiederverwendung bei der Untersuchung agiler Softwareprojekte. Deren Anspruch ist es, durch dauerhaft enge Zusammenarbeit von Nutzer*innen,

---

Funktionen wie „hinzufügen", „sortieren", „entfernen" und „Länge ausgeben" bereitstellt. Wie die Daten im Speicher des Computers abgelegt und bearbeitet werden, ist den Nutzer*innen des Listenobjekts, also den Programmierer*innen, nicht bekannt. Der Code, der die Speicherverwaltung betrifft, muss nicht bei jedem Softwareprojekt neu geschrieben werden, das Listen verwendet. Stattdessen wird im neuen Code nur das Objekt „Liste" mit den benötigten Funktionen referenziert; erst auf einer dem Prozessor näheren Ebene der Codierung (s. 2.1.1) wird der Code, der das grundsätzliche Verhalten der Liste beschreibt, mit dem neu erstellten Code, in dem die Liste angewandt wird, zu einer maschinenlesbaren Variante der fertigen Software verbunden.

Entwickler*innen und anderen Stakeholder*innen und permanente Weiterentwicklung Software zu produzieren, deren Funktionen den Bedürfnissen der Nutzer*innen besonders gut gerecht werden. In der hier entwickelten Perspektive versuchen die Softwareentwickler*innen bei agilen Ansätzen, Nutzer*innen möglichst an den Aushandlungen über die für das soziale Phänomen bedeutsamen Aspekte der Unsicherheit und deren Modellierung zu beteiligen. Besteht ein relevanter Teil der Software aus wiederverwendeten Elementen, wie dies zunehmend der Fall ist, lässt sich dieser Anspruch nicht mehr erfüllen.

Ein ähnlicher Effekt entsteht bei standardisierter Organisationssoftware: Bei langlebigen Produkten dieser Art erzeugen Weiterentwicklung und Wiederverwendung eine charakteristische interne Struktur, die ich in dieser Arbeit als Struktur konzeptioneller Schichten bezeichne. Diese Struktur, schematisch dargestellt in Abbildung 2.5, wird im Folgenden genauer beschrieben.

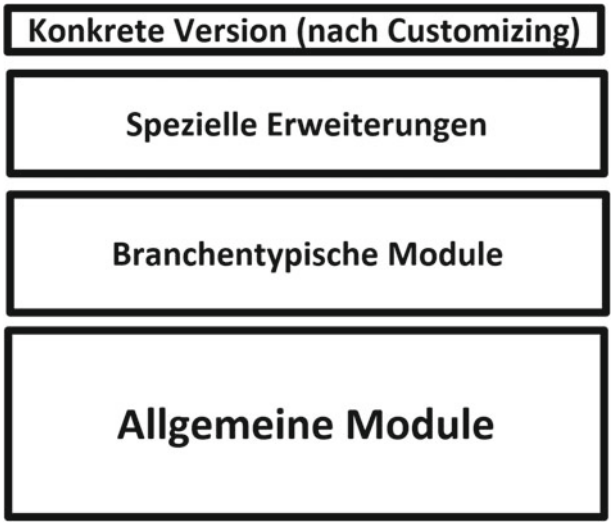

Abbildung 2.5 Konzeptionelle Schichten standardisierter Organisationssoftware. (Eigene Darstellung)

2.1.3.3 Konzeptionelle Schichten standardisierter Organisationssoftware

Standardisierte Organisationssoftware existiert häufig in verschiedenen Varianten, die für bestimmte Branchen oder Wirtschaftsbereiche entwickelt wurden. Diese Varianten unterstützen zusätzlich zu „allgemein typischen" Arbeitsabläufen (z. B. Rechnungsstellung) auch „speziell typische" (z. B. Prüfungsverwaltung im Bildungsbereich). Sie bestehen aus einer Reihe an allgemeinen Modulen, die allgemein typische Prozesse modellieren, und einer Reihe an branchentypischen Modulen, die wiederum in die allgemeinen integriert sind (Davenport 1998). Bei der Entwicklung und Weiterentwicklung branchenspezifischer Varianten der Software werden nur die Aspekte bearbeitet, die die branchenspezifischen Phänomene betreffen. Phänomene, die die Softwareherstellerin als allgemeine Organisationsphänomene einstuft, werden in den Entwicklungsprozessen der allgemeinen Module modelliert. Im Ergebnis kann die Struktur der Modelle in standardisierter Organisationssoftware als geschichtet betrachtet werden: Module, die allgemeinere Phänomene modellieren, bilden das Fundament, auf dem die Module aufbauen, die speziellere Phänomene modellieren. In den Entwicklungsprozessen werden Aspekte bearbeitet, die auf der eigenen und darüber liegenden (also spezielleren) Schichten verortet werden, nicht jedoch solche, die auf niedrigeren (also allgemeineren) Schichten liegen.

Für populäre Produkte wie das in dieser Arbeit untersuchte von SAP gibt es darüber hinaus spezielle Erweiterungen. Diese werden von anderen Herstellerinnen entwickelt und bieten weitere Funktionen an. Die Abhängigkeiten zwischen den Entwicklungsprozessen der Erweiterungen und denen des Standardprodukts entsprechen denen zwischen spezielleren und allgemeineren Schichten.

Standardisierte Organisationssoftware bildet eine informationstechnische Infrastruktur für die Organisationen, die sie einsetzen, da sie einen großen Teil der Prozesse in der Organisation modelliert. Soll sie in der Organisation ihren offensichtlichen Zweck (die weitgehende informationstechnische Integration der Arbeitsabläufe) erfüllen, muss der Rest der Software auf diese Infrastruktur abgestimmt werden. Die Einführung von standardisierter Organisationssoftware wirkt damit performativ auf den Umgang mit (beinahe) jeglicher Software in der Organisation, da die Akteur*innen dadurch gezwungen werden, ihre Arbeitsabläufe mit den Modellen organisatorischer Prozesse in Einklang zu bringen, die das Standardprodukt vorgegeben hat.

Auch bei standardisierter Organisationssoftware können einmal erfolgte Reduktionen in späteren Phasen hinterfragt und die geschlossenen Aspekte der Unsicherheit damit wieder aufgegriffen werden. Bei der Entwicklung branchentypischer Varianten können zum Beispiel Vorschläge für Veränderungen an den

Modellen der allgemeinen Module auftauchen. Die Entscheidung darüber, ob und wie solche Vorschläge aufgegriffen werden, verbleibt jedoch in aller Regel in den sozialen Prozessen, in denen die allgemeinere Schicht weiterentwickelt wird. Darüber hinaus ist gerade für die Akteur*innen in Customizing und Nutzung nur schwer nachvollziehbar, auf welcher Schicht die Aspekte der Unsicherheit modelliert (und damit in eindeutige Lösungen übersetzt) worden sind, die in ihren eigenen Prozessen zum Thema werden. Ihre Möglichkeiten, Einfluss auf die Software auszuüben, sind auch durch diese Undurchschaubarkeit eingeschränkt.

2.1.4 Fazit: Soziale Komplexität von Software

In Abschnitt 2.1.1 habe ich dargestellt, dass Programmiersprachen als eine Reihe von Modellen betrachtet werden können, die durch mathematische Transformationen ineinander überführt werden und deren Vorlage der Befehlssatz des Prozessors ist. Dieser Befehlssatz stellt Informatiker*innen ein Modell des Prozessors zur Verfügung, in dem die Unsicherheiten, die die physischen Elemente des Computers mit sich bringen, ausgeblendet werden, so dass deren Funktionen durch Reihen binär codierter Anweisungen dargestellt werden können. Der Code der Software muss dadurch immer eine vollständige und logisch konsistente Beschreibung ihrer Funktionen enthalten. Die Konstruktion von Software wird damit zu einer komplexen Modellierungsaufgabe für Menschen, da ihnen vollständige und logisch konsistente Beschreibungen der meisten Tatbestände schwerfallen. In Abschnitt 2.1.2 habe ich dargestellt, dass Softwareentwicklung als eine Reihe von Modellierungsprozessen betrachtet werden kann, die aufeinander aufbauen. Jeder dieser Prozesse reduziert einige Aspekte der Unsicherheit des Sozialen und ist dabei durch die Reduktionen vorangehender Prozesse vorstrukturiert. Jeder Prozess besitzt als sozialer aber auch eigene Unsicherheiten und sorgt dafür, dass diese die darauffolgenden Prozesse mit strukturieren. Im Modellierungsprozess von Software wird damit die Komplexität des Modells gesteigert, weil die Akteur*innen versuchen, die Komplexität des Phänomens zu reduzieren. Das Ergebnis der Softwareentwicklung ist ein performatives Artefakt, weil die Modelle die Prozesse der Nutzung vorstrukturieren. In Abschnitt 2.1.3 wurde gezeigt, wie Software auch in den Prozessen der Softwareentwicklung performativ wird. Bei der Weiterentwicklung müssen sich Akteur*innen an dem Modell früherer Versionen orientieren, wenn sie den größten Teil der dadurch ausgelösten Funktionen für neue Versionen erhalten wollen. Ihr Verständnis der Modelle wird dabei mit zunehmendem zeitlichen und sozialen Abstand zwischen den Entwicklungsprozessen erschwert, was ihre Handlungsmöglichkeiten zusätzlich

einschränkt. Der Umgang mit früheren Versionen der Software lässt neue Anforderungen entstehen, so dass nicht nur die Software an sich, sondern auch die sozialen Prozesse, in die sie innerhalb der sozio-technischen Systeme eingebunden ist, Rückwirkungen auf die Weiterentwicklung haben. Eine andere Form von Unsicherheitsreduktion stellt die Nutzung wiederverwendbarer Softwareelemente dar. Mit diesen tauschen die Akteur*innen in Softwareentwicklungsprozessen eigene Kontrolle über den Modellierungsprozess gegen erprobte Modelle ein, die sie in das eigene Modell integrieren müssen. Dadurch verknüpfen sie ihre eigenen Entwicklungsprozesse mit denen, in denen die wiederverwendeten Modelle weiterentwickelt werden, und erzeugen so Unsicherheit für die Zukunft ihres eigenen Produkts.

Bis hierhin habe ich die Modellierungsprozesse dargestellt, die für die Softwareentwicklung und den Umgang mit Software charakteristisch sind, und auf die soziologischen Fragen hingewiesen, die in diesen Prozessen aufgeworfen werden. Im folgenden Abschnitt wird nun gezeigt, wie diese Fragen bisher in der soziologischen Forschung zu Software aufgegriffen worden sind.

2.2 Soziologische Forschung zu Software

Zwar wird die Komplexität von Software selten zum zentralen Gegenstand soziologischer Forschung gemacht, aber es gibt eine Reihe von soziologischen Arbeiten, in denen Fragen wie die, die im vorangegangenen Kapitel dargestellt wurden, aufgegriffen werden. Software kann dabei immer nur insofern von soziologischem Interesse sein, als in irgendeiner Form in sozialen Prozessen auf sie Bezug genommen wird, insofern sie also entwickelt oder genutzt oder ihre Entwicklung oder Nutzung in irgendeiner Form thematisiert wird. Letzteres gilt zum Beispiel auch für Prozesse, in denen Regulierung, Anschaffung, Bedingungen oder Konsequenzen von Softwareentwicklung oder -nutzung ausgehandelt werden, in denen die Software selbst also gar nicht zum Einsatz kommt. Durch all diese Formen der Bezugnahme entsteht eine Verbindung zwischen Software und sozialen *Praktiken*, wiederkehrenden, überindividuell geteilten, regelgeleiteten und situativ adaptierbaren Handlungsweisen (Reckwitz 2003).[45] In soziologischen Arbeiten zum Umgang mit Software werden häufig solche Praktiken untersucht, oder die Praxis, also das, was Akteur*innen tatsächlich tun, wenn sie

[45] Diese sehr knappe Bestimmung des Begriffs „Praktiken“ stellt eine Art kleinsten gemeinsamen Nenner einer Reihe unterschiedlicher soziologischer (und nicht-soziologischer, vgl. Schmidt 2018) Definitionen dar, der alle für eine Soziologie der Software relevanten Elemente beinhaltet.

sich in ihrem Handeln mit den konkreten Bedingungen der Welt um sie herum auseinandersetzen müssen.[46] Wenn die Software von Soziolog*innen als relevantes Element von Praktiken oder Praxis berücksichtigt wird, greifen diese häufig auf Erkenntnisse der Techniksoziologie zurück.

Techniksoziolog*innen haben sich verhältnismäßig wenig mit der Frage befasst, was das Besondere an der Software sein könnte, sondern allgemein untersucht, wie Technik und Gesellschaft zusammenspielen. Im Gegensatz zum rationalistisch dominierten Diskurs in der Informatik (vgl. Floyd 1989) wird in der Techniksoziologie einerseits davon ausgegangen, dass die Perspektiven, mit denen Technikentwickler*innen auf die Welt blicken, und die Bedingungen, unter denen sie arbeiten, das Ergebnis ihrer Arbeit bestimmen (Winner 1980; Pinch und Bijker 1987; Bijker et al. 1987). Da dieses Ergebnis in sozialen Prozessen eingesetzt wird, gestaltet Technikentwicklung (und die sie kennzeichnenden Perspektiven und Bedingungen) immer auch das Soziale:

> *„In der technischen Praxis werden nicht nur Probleme technisch effektiv und scheinbar politisch neutral gelöst, sondern neue Problemsichten und neue Praktiken gegenüber alten und eingelebten durchgesetzt. Praktiken der Technikproduktion sind daher so politisch wie andere Praktiken, da sie immer Partei nehmen, zumindest gegen die vorherige Praxis, und da sie immer neue Rahmen und Regeln für die Praxis setzen.“* (Rammert et al. 1998, S. 34)

Technikentwicklung muss damit sowohl der Form als auch ihren Ergebnissen nach als politisch betrachtet werden: Ersteres, weil die Entwickelnden mit der Entscheidung für eine Art und Weise, Technik zu gestalten, „Partei“ in einer Auseinandersetzung ergreifen, in der es nicht eine objektiv beste, sondern verschiedene gleichermaßen mögliche Lösungen gibt. Letzteres, weil zusammen mit der Technik auch Bedingungen des Zusammenlebens gestaltet werden.

Zur soziologischen Perspektive auf Technik gehört andererseits auch die Erkenntnis, dass die Gestaltbarkeit des Sozialen durch Technik begrenzt ist. Technik kann soziale Prozesse nur nachhaltig beeinflussen, wenn sie in diesen längerfristig genutzt wird, wodurch sich in irgendeiner Form geregelte Umgangsformen ausbilden. Wie diese geregelten Umgangsformen aussehen ist davon

[46] „In der Praxistheorie erscheint die soziale Welt der Praktiken im Spannungsfeld zweier grundsätzlicher Strukturmerkmale: der Routinisiertheit einerseits, der Unberechenbarkeit interpretativer Unbestimmtheiten andererseits. Anders formuliert, bewegt sich die Praxis zwischen einer relativen ‚Geschlossenheit‘ der Wiederholung und einer relativen ‚Offenheit‘ für Misslingen, Neuinterpretation und Konflikthaftigkeit des alltäglichen Vollzugs. Diese beiden Aspekte (die allerdings bei verschiedenen praxeologischen Autoren in unterschiedlicher Weise betont werden) markieren keinen Widerspruch, sondern zwei Seiten der ‚Logik der Praxis‘.“ (Reckwitz 2003, S. 295).

abhängig, wie Nutzer*innen die Technik interpretieren und wie sie diese konkret in ihren Alltag einbauen (Rammert 2006).

Das Wechselverhältnis zwischen diesen beiden Aspekten wird als Schlüssel zum Verständnis der Rolle von Technik im Sozialen betrachtet (Schulz-Schaeffer 1999). Schulz-Schaeffer bezeichnet dieses in Anlehnung an Giddens' Strukturationstheorie als *Dualität von Ressourcen und Routinen.* Technik[47] ist insofern Ressource des Handelns, als Akteur*innen sie in ihren Handlungsabläufen nutzen, um einen Zusammenhang zwischen Ereignissen herzustellen, der durch Regeln formuliert wird, die der Technik zu Grunde liegen: *„Ein Akteur tut A, um B zu erreichen, indem er sich auf einen Ereigniszusammenhang verlässt (seine Ressource), der seiner begründeten Erwartung nach bewirkt, dass die Handlung A mit einiger Zuverlässigkeit die Folge B zeitigt"* (Schulz-Schaeffer 1999, S. 414). Ereigniszusammenhänge im Sozialen entstehen aber durch die Praxis und sind damit immer kontingent. Sie basieren auf Regeln des Sozialen, die ebenso interpretationsoffen sind wie die Praxis, die sie regulieren. Die ausformulierten Regeln für den Ereigniszusammenhang, die der Technik zugrunde liegen, können immer nur eine der vielen möglichen Interpretation dieser sozialen Regeln festhalten (Giddens 2008 [1984]). Diese Regeln und die auf ihnen beruhende Technik können daher die Ereigniszusammenhänge, die Akteur*innen beim Einsatz der Technik erwarten, nicht allein erzeugen. Dies gelingt nur (mit relativer Wahrscheinlichkeit), wenn zusätzlich Praktiken existieren, die die Interpretationen der Akteur*innen, die am Entstehen dieser Ereigniszusammenhänge beteiligt sind, mit den Interpretationen in Einklang bringen, die den formulierten Regeln der Technik zu Grunde liegen.

> *„Es ist […] nur deshalb möglich, einen Ausschnitt des gesellschaftlichen Lebens explizit zu reglementieren, weil zugleich eine Vielzahl von Handlungspraktiken, die in der einen oder anderen Weise mit dem reglementierten Ausschnitt zusammenhängen, als selbstverständlich vorausgesetzt werden können." (Schulz-Schaeffer 1999, S. 415)*

Um komplexe Techniken nutzen zu können, müssen Akteur*innen nur einen kleinen Teil der Regeln kennen, auf denen die Technik beruht, nämlich nur den Teil, der sich direkt auf ihr Handeln und das Ergebnis bezieht, das sie erwarten können. Der Rest der Regeln muss nur einer Gruppe von Expert*innen

[47] Schulz-Schaeffer bezieht seine Aussagen streng genommen nur auf „Sachtechnik" oder „gegenständliche Technik", die er von „Handlungstechnik" (z. B. Rhetorik, autogenes Training) abgrenzt (Schulz-Schaeffer 1999, S. 410). In der vorliegenden Arbeit will ich keine allgemein techniksoziologische Argumentation führen, sondern nur über Software sprechen, die als Sachtechnik aufgefasst wird. Um die bereits ausreichend komplexe Darstellung nicht zunehmend unverständlich zu machen, spreche ich im Rahmen dieser Arbeit nur allgemein von „Technik".

bekannt sein. Er wird in von der Nutzungspraxis getrennten gesellschaftlichen Kontexten, zum Beispiel in wissenschaftlichen Disziplinen oder professionellen Gemeinschaften von Ingenieur*innen, bearbeitet. Für Außenstehende, also Akteur*innen, die sich nicht an der Bearbeitung dieser Regeln beteiligen, werden in diesen Kontexten Systeme aufeinander bezogenen Wissensbestände, Expert*innensysteme, produziert, die regelhafte Ereigniszusammenhänge ermöglichen. Alle komplexen Techniken können als Expert*innensysteme betrachtet werden. Die Entwicklung komplexer Techniken findet in sozialen Kontexten statt, in denen Expert*innen interagieren, muss aber auf die Nutzungspraxis außerhalb dieser Kontexte bezogen werden, um sicherzustellen, dass formulierte Regeln und sie stützende Handlungspraktiken ineinandergreifen. Die Praktiken von Expert*innen und Lai*innen zum Umgang mit der Technik beschränken und ermöglichen einander damit wechselseitig (a.a.O., S. 419).

Der folgende Literaturüberblick konzentriert sich auf Arbeiten aus der Soziologie und verwandter Disziplinen, die Software nicht allein als Beispiel für allgemein techniksoziologische Fragen behandeln, sondern ihre Besonderheiten im Vergleich zu anderen Formen von Technik thematisieren. Wie sich zeigen wird, teilen die meisten Arbeiten in diesem Feld prinzipiell die Grundidee der Dualität, also die Annahme, dass die in die Technik eingebetteten Regeln und die Praktiken des Umgangs mit dieser Technik wechselseitig aufeinander bezogen sind und einander sowohl beschränken als auch ermöglichen. Die wenigstens ziehen daraus jedoch Konsequenzen. In der Zusammenfassung dieses Kapitels werde ich daher auf die Idee der Dualität zurückkommen und herausstellen, welche Gemeinsamkeiten die verschiedenen Arbeiten aufweisen und wo einzelne Arbeiten einander ergänzen. Das Konzept der Expert*innensysteme wird in Abschnitt 3.1.1 weiter ausgeführt. Zunächst wird aber dargestellt, auf welche Weise Soziolog*innen die Fragen aufgegriffen haben, die die drei Dimensionen der Komplexität von Software aufwerfen. Die Darstellung der soziologischen Arbeiten, in denen dieses Fragen thematisiert werden, folgt der gleichen Struktur wie die Darstellung der drei Dimensionen selbst: Als erstes stelle ich Arbeiten vor, in denen Soziolog*innen sich mit der Übersetzung von sozialen Phänomenen in mathematische Formalismen bei der Softwareentwicklung auseinandergesetzt haben. Im Anschluss folgen Arbeiten zu soziologischen Aspekten des Softwarelebenszyklus. Die soziologische Forschung zu Weiterentwicklung und Wiederverwendung wird zuletzt vorgestellt. Der Überblick über die soziologische Forschung zu Software dient als Grundlage für die Herleitung der in dieser Arbeit forschungsleitenden soziologischen Perspektive auf Software, die ich in 2.3 ausführen werde, und der Einordnung des Forschungsrahmens, der in Kapitel 3 entwickelt wird.

2.2.1 Arbeiten zur Übersetzungsdimension

Bei der Softwareentwicklung werden Phänomene aus der sozialen Welt ausgewählt, interpretiert und in mathematische Formalismen übersetzt. In den Modellen, die im Laufe dieser Übersetzung entstehen, sind die Unsicherheiten, die die Auswahl und Interpretation sozialer Phänomene mit sich bringt, nicht mehr sichtbar. Beim Umgang mit der Software repräsentieren diese Modelle die in ihnen abgebildeten Phänomene. Die vielen Entscheidungen, die im Laufe der Modellierungsprozesse getroffen werden müssen, um eine funktionsfähige Software zu entwickeln, sind durch das Endziel der Modellierung vorstrukturiert, also dadurch, dass Mikroprozessoren nur eindeutige, vollständige und widerspruchsfreie mathematische Formalismen verarbeiten können. Zentrale soziologische Fragen, die durch dieses grundlegende Funktionsprinzip von Computern aufgeworfen werden, sind in den Arbeiten von Werner Rammert, Michael Schlese, Gerald Wagner, Josef Wehner und Rüdiger Weingarten (Rammert et al. 1998), von Christiane Funken (2001) und von Jannis Kallinikos (2009) behandelt.

2.2.1.1 Wissensproduktion durch Kombination von Repräsentationsebenen

Aus mehreren empirischen Studien zur Softwareentwicklung setzt sich das Buch von Rammert et al. (1998) zusammen. Dieses ist eine der wenigen soziologischen Arbeiten, in der die grundlegende Funktionsweise von Mikroprozessoren aus einer soziologischen Perspektive, der zu Folge das Soziale grundlegend unsicher ist, systematisch diskutiert wird. Ergebnis dieser Diskussion ist das Konzept des Computers als Wissensmaschine: als Artefakt, das Eigenschaften von Maschine und Medium verbindet. Durch seine besondere technische Struktur ist ein Computer eine Maschine, die zweckgebunden Funktionen auf vorhersehbare Weise erfüllt. Gleichzeitig ist ein Computer aber auch ein Medium, dessen Eigenheiten beeinflussen, wie Wissen im Umgang mit ihm abgebildet, transformiert und neu geschaffen wird (a.a.O., S. 10 f.). Die besondere technische Struktur des Computers, auf der dieser Doppelcharakter beruht, beschreiben Rammert et al. als System aufeinander aufbauender Repräsentationsebenen. Auf jeder Repräsentationsebene werden Zeichen entsprechend jeweils eigener Gesetze transformiert, zum Beispiel wenn Anweisungen in einer höheren Programmiersprache durch automatische Übersetzung in Anweisungen in der Maschinensprache übersetzt werden. Den in Abschnitt 2.1.1 als Basis der Funktionen von Software dargestellten Schichten Hardware, Maschinensprache, und höhere Programmiersprachen entsprechen die Ebenen der physikalischen, logischen und abstrakten Maschine. Sie sind durch deterministische Übersetzungsregeln verbunden und ermöglichen

gemeinsam das zweckgebundene und (relativ) vorhersehbare Funktionieren der Computer (a.a.O., S. 19). Bei der Softwareentwicklung entstehen komplexe *„Repräsentationsschemata für ‚Fakten'"* (a.a.O. S. 20) als oberste Ebene, durch die Wissen aus den sozialen Prozessen, in denen der Computer zum Einsatz kommt, in die von der Maschine bearbeitbaren Zeichen übersetzt wird. Über diese Ebenen verbindet der Computer *„die Determiniertheit von Kausalbeziehungen mit der Offenheit*[48] *von Repräsentationsbeziehungen"* (a.a.O., S. 19). Der Computer bringt also die durch eindeutige Regeln gesteuerten Zeichenmanipulationen der mathematischen Formalismen, auf denen die Maschine basiert, mit den hochgradig unsicherheitsbehafteten sozialen Prozessen der Wissensproduktion zusammen.

Rammert et al. entwickeln ihr Modell anhand mehrerer Studien zur Entstehung und Verbreitung von Expert*innensystemen.[49] Sie zeigen: Was beim Umgang mit diesen Systemen als Wissen gilt, hängt sowohl davon ab, welche Probleme in den sozialen Prozessen bearbeitet werden, in denen mit den Repräsentationsschemata umgegangen wird, als auch davon, welche Weltanschauungen, professionellen Orientierungen, Einflussmöglichkeiten und andere *„soziostrukturelle[n] Merkmale"* die beteiligten Akteur*innen in die Prozesse einbringen (a.a.O., S. 47 ff.). Die Autoren untersuchen die Entwicklung der informatischen Grundlagen der Expert*innensysteme, die Entwicklung konkreter Expert*innensysteme sowie deren Nutzung als aufeinander aufbauende *„Wissenswelten"* (a.a.O., S. 50). Diese Wissenswelten sind durch das dominante Problem gekennzeichnet, über dessen Lösung in jeweils charakteristischen Akteur*innenkonstellationen verhandelt wird. Rammert et al. konzentrieren sich in ihrer Untersuchung darauf, welche Folgen die Einbettung der Akteur*innen in die verschiedenen Wissenswelten für die Entwicklung der Expert*innensysteme hat. Dabei kommen sie zu dem Schluss: Während sich die Interpretationen der Akteur*innen, die informatische Grundlagen entwickeln, nach denen Wissen verarbeitet werden soll, vor allem an den Gesetzen der Maschine orientieren, haben die Akteur*innen bei der Nutzung vor allem die soziale Einbettung des Wissens vor Augen. Die Akteur*innen bei der

[48] „Offenheit" meint in diesem Zusammenhang die Offenheit für Interpretationen.

[49] Expert*innensysteme galten in den 1970er und 1980er Jahren als nächste Stufe der Software und sollten Wissen und Folgerungsfähigkeit von Expert*innen aller Art auf den Computer übertragen. In der Praxis erwiesen sich diese Hoffnungen als unbegründet; die damaligen Expert*innensysteme fanden nie große Verbreitung (Rammert et al. 1998, S. 9). Als Soziolog*in kann man die Expert*innensysteme (zumindest aus der Entfernung) insofern als Vorläufer der aktuell wieder intensiv diskutierten Künstlichen Intelligenz betrachten, als in den Auseinandersetzungen um beide Arten von Systemen sehr ähnliche (positive wie negative) Vorstellungen von Gesellschaft verhandelt werden.

Entwicklung sind durch ihre soziostrukturelle Verortung und die institutionellen Bedingungen der Aushandlungen den Regeln aller Repräsentationsebenen verbunden und müssen sie in ihrer Praxis miteinander vereinbaren (a.a.O., S. 78 ff.). Entwickler*innen und Nutzer*innen bewegen sich dabei in einem Handlungskorridor, der durch die Aushandlungen in den vorgelagerten Wissenswelten geformt, auf Grund der Interpretationsbedürftigkeit des Wissens aber niemals zum Zwang wird (a.a.O., S. 249 ff.).

In dem Konzept des Computers als Wissensmaschine setzen sich Rammert et al. direkt damit auseinander, wie und mit welchen sozialen Folgen Software die deterministische Verarbeitung von Daten mit den kontingenten Interpretationen sozialer Phänomene verknüpft. Damit weisen sie auf den in späteren Arbeiten häufig ignorierten Zusammenhang hin, der zwischen den technischen Grundlagen von Software und ihrer Bedeutung für das Soziale besteht. Dieser Zusammenhang ist für ein Verständnis der Komplexität von Software zentral. In dieser Hinsicht skizzieren Rammert et al. die Grenzen des Raums der sozialen Komplexität von Software, der in Abschnitt 2.1 aufgespannt wird. Konkret binden sie ihre Aussagen aber eng an die spezifischen Akteur*innenkonstellationen, Probleme und Prozesse von zum Untersuchungszeitraum hochgradig wissenschaftsnahen und experimentellen Expert*innensystemen, die sich nicht über einen engen Kreis von an der Entwicklung beteiligten Pilotkund*innen verbreitet haben (a.a.O., S. 259 ff.). Darüber hinaus blenden sie durch den Fokus auf die Wissenswelten aus, dass es bei der Entwicklung von Software auch um die Durchsetzung von (teilweise widersprüchlichen) Interessen geht und dass die Akteur*innen im Entwicklungsprozess nicht nur unterschiedliche Orientierungen, sondern auch unterschiedlich viel Macht haben, um ihre Interessen durchzusetzen. Der Bezug auf die jeweilige Rolle im Prozess und auf die hinter dieser Rolle stehende Wissenswelt kann eine solche Machtressource sein. Beispiele dafür finden sich in Christiane Funkens Buch „Die Modellierung der Welt“ (Funken 2001). Dieses enthält Untersuchungen zu den Auswirkungen der Idee der universellen Turingmaschine auf die Weltanschauungen und Praktiken von Informatiker*innen.

2.2.1.2 Hintergrundordnungen und Repräsentationspraktiken der Informatik

Funken (2001) konzentriert sich in diesen Untersuchungen auf die Anforderungsanalyse, also die erste Phase im Softwarelebenszyklus. Bei der Anforderungsanalyse sollen Softwareentwickler*innen und Repräsentant*innen zukünftiger Nutzer*innen gemeinsam ein Modell nach Vorlage eines sozialen Phänomens erstellen. Dabei werden die Probleme besonders deutlich, die der Versuch mit

sich bringt, unsichere soziale Phänomene in Modelle zu übersetzen, die frei von Unsicherheit sind und damit im Prozessor verarbeitet werden können. Der Gegensatz zwischen den beiden Darstellungsformen spiegelt sich ihrer Ansicht nach im Verhältnis zwischen Softwareentwickler*innen und Nutzer*innen. Das Verhalten der Softwareentwickler*innen wird dabei durch professionelle Einstellungen zu ihrer Aufgabe und den Nutzer*innen geprägt. Diese Einstellungen bezeichnet Funken als Hintergrundordnungen der Informatik (a.a.O., S. 89–104). Diese Hintergrundordnungen sind normative Orientierungen der Berufsgruppe, die sich Softwareentwickler*innen während der Ausbildung und im beruflichen Alltag aneignen. Sie finden sich in den Strukturen der Programmiersprachen und Softwareentwicklungswerkzeugen wieder. Softwareentwickler*innen verstehen sich Funken zu Folge vor allem als Ingenieur*innen (a.a.O., S. 71), deren Aufgabe die Konstruktion eines Modells ist, das aus diskreten Einheiten besteht, die in wohldefinierten Beziehungen zu einander stehen und deren Verhalten durch allgemeine Regeln beschrieben werden kann. Soziale Faktoren, die diese Aufgabe verkomplizieren könnten, werden weitgehend ausgeblendet, Nutzer*innen werden als unberechenbare Elemente des Systems betrachtet und ihr Einfluss auf die Modellierung wird möglichst beschränkt (a.a.O., S. 72). Diese formalistische Primärorientierung steht im Widerspruch zu einer Praxis, in der die Softwareentwickler*innen ihre Arbeit als kreativen Prozess wahrnehmen und hohe intrinsische Motivation aus einem als intuitiv empfundenen Arbeiten ableiten (a.a.O., S. 84). Softwareentwicklung ist damit auch aus Sicht der untersuchten Informatiker*innen ein Prozess voller Ambivalenzen.

Funken beschreibt zwar, wie Softwareentwickler*innen sich auf ihre Expertise berufen, um gegenüber Nutzer*innen ihre Interessen durchzusetzen, geht aber ebenso wie Rammert nicht systematisch darauf ein, welche Bedeutung Macht für die Modellierungsprozesse der Softwareentwicklung hat. Die Macht der Informatiker*innen wird bei Brian Bloomfield und Theo Vurdubakis zum Thema. Nicht die handlungsleitenden Orientierungen der Softwareentwickler*innen, sondern die formalisierten Methoden und Werkzeuge der Softwareentwicklung sehen Bloomfield und Vurdibakis (1997) als zentrale Mechanismen der Übersetzung an. Diese Methoden und Werkzeuge betrachten sie als Repräsentationspraktiken, die dafür sorgen sollen, dass die Unsicherheit des Sozialen im Laufe des Softwareentwicklungsprozesses verschwindet. Ziel der Repräsentationspraktiken ist die Konstruktion eines Modells, das die Essenz des sozialen Phänomens enthält (a.a.O., S. 649 f.) – eine Essenz, die von den Auftraggeber*innen der Softwareentwicklung festgelegt worden ist. Unterschiedliche Interpretationen des Phänomens werden im Entwicklungsprozess dabei nur symbolisch berücksichtigt: *„[T]hough subjectivity is accorded an important place at the start of the analysis it must*

eventually be 'supplemented' by objectivity." (Bloomfield und Vurdubakis 1997, S. 662). Die Behauptung der „Objektivität" verschleiert dabei, dass Repräsentationspraktiken dabei helfen sollen, das repräsentierte Phänomen entsprechend den Zwecken der Auftraggeber*innen umzugestalten. Die Repräsentation bildet nach Ansicht der Autoren das Phänomen nicht ab, sondern konstruiert eine idealisierte Version davon, die bei der Nutzung als Maßstab dient, an dem das tatsächliche Phänomen gemessen wird (a.a.O., S. 664).

2.2.1.3 Rekonstitution der Realität durch den Filter der Berechnung

Auch Jannis Kallinikos betrachtet die behauptete Objektivität der Repräsentationen, die bei der Softwareentwicklung entstehen, als das zentrale soziologische Problem der Übersetzung sozialer Phänomene in Software. Anders als die Autor*innen der anderen Arbeiten stellt er jedoch die Auswirkungen dieser angeblichen Objektivität in den Mittelpunkt (Kallinikos 2005, 2009). Er befasst sich also mit dem Ergebnis der Modellierung und dem Umgang mit Modellen, nicht mit dem Modellierungsprozess oder seinen Akteur*innen. Die Besonderheit von Software liegt Kallinikos Ansicht nach darin, wie Informationen über die Welt in Daten übersetzt und wie von der Software produzierte Daten von Nutzer*innen interpretiert werden. Die Modelle, die im Softwareentwicklungsprozess konstruiert worden sind, erlauben es, soziale Phänomene digital zu repräsentieren. Bei der digitalen Repräsentation werden analoge Objekte, Subjekte und Prozesse, also Elemente der Welt, die Menschen als Einheiten betrachten, automatisch mit Hilfe von Modellen in mathematische Formalismen übersetzt und damit über mehrere Modellebenen hinweg bis auf eine Binärdarstellung dekonstruiert. Diese Binärdarstellungen, die digitalen Objekte, ersetzen für diejenigen, die die Software nutzen, ihre analogen Vorlagen, also zum Beispiel andere Menschen mit ihren Eigenschaften und Beziehungen, physische Gegenstände oder soziale Prozesse. Software sorgt einerseits für die verlässliche Dekonstruktion, die eine Bearbeitung der digitalen Objekte für den Prozessor erst möglich macht, stellt andererseits gleichzeitig sicher, dass die digitalen Objekte für die Benutzer*innen immer noch als Einheit erscheinen. Wie jede Funktion von Software, die für Menschen wahrnehmbar ist, beruhen Dekonstruktion, Verarbeitung und Darstellung digitaler Objekte auf einem komplexen Geflecht automatisierter Funktionen, welches sich in seiner Gesamtheit dem menschlichen Verständnis entzieht (Kallinikos

2009).[50] In dieser beinahe vollständigen, nicht nachvollziehbaren Dekonstruktion sieht Kallinikos den Ursprung zweier grundlegender Unterschiede zwischen digitalen Objekten und den durch sie repräsentierten Elementen der Welt, die Auswirkungen auf die sozialen Prozesse haben, in denen Software genutzt wird:

(1) Alle Eigenschaften der analogen Objekte, die durch die Software verarbeitet oder dargestellt werden sollen, müssen bei ihrer Dekonstruktion im digitalen Objekt nachgebildet werden. Wird zum Beispiel bei der digitalen Modellierung von Patient*innen nicht zu irgendeinem Zeitpunkt gezielt entschieden, dass Ernährungsgewohnheiten eine relevante Eigenschaft sind, so gibt es später keine Möglichkeit, an Hand der digitalen Objekte etwas über ernährungsbedingte Krankheiten herauszufinden. Die digitalen Patient*innen unterscheiden sich in ihren Ernährungsgewohnheiten nicht, denn sie besitzen einfach keine. Weil alle Eigenschaften digitaler Objekte von der Software nachgebildet werden, können sie auch beliebig verändert werden. Digitale Patient*innen können von jemandem, der die Software zu bedienen weiß, mit ein paar Tastendrücken verjüngt und geheilt werden, ihr Geschlecht, ihre Größe oder ihre Behandlungshistorie kann sich verändern, ohne dass die repräsentierte Patient*in davon etwas mitbekommt. Digitale Objekte sind vollständig über Software kontrollierbar; Software kann auch ohne Vorlage neue digitale Objekte produzieren, die nicht von Repräsentationen analoger Objekte unterscheidbar sind.

Während der Umgang mit analogen Objekten vor allem davon abhängt, wie sie von den Akteur*innen im Prozess interpretiert werden, wird der Umgang mit digitalen Objekten stark davon beeinflusst, welche Interpretationen sich bei der Entwicklung der Funktionen durchsetzen, über die ihre analogen Vorlagen

[50] Kallinikos zeigt am Beispiel der Darstellung von Logistikprozessen in der standardisierten Organisationssoftware SAP, dass sich die Dekonstruktion auf nachvollziehbare und nicht nachvollziehbare Ebenen verteilt: Die Software zeigt den Nutzer*innen durch die Einteilung von Funktionen in Kategorien und Subkategorien an, wie die Arbeitsprozesse beim Warenausgang dekonstruiert werden. Das, was den Nutzer*innen als einzelne Funktion angezeigt wird, entspricht jedoch nicht der untersten Ebene der Dekonstruktion, sondern nur einer für die Nutzer*innen akzeptablen kleinsten Einheit: *„At this juncture, computation moves away from culturally and perceptually recognized units of the reality. The computational mechanics of information processing construct now the minute processes by which a category, say goods movement, is comprised, and run how changes in the category are recorded and monitored. While humans may input data into the system, the link between data and the category is fully automated and beyond human recognition“* (Kallinikos 2009, S. 190).

dekonstruiert werden. Diese Interpretationsprozesse werden durch die automatisierte Dekonstruktion versteckt. Akteur*innen sehen beim Umgang mit digitalen Objekten nur diese, also das Ergebnis der Transformationen von Eigenschaften einer analogen Vorlage (z. B. bei der Eingabe von Patient*innendaten bei der Aufnahme im Krankenhaus). Was sie nicht sehen, sind die vielen Aushandlungen in den Modellierungsprozessen bei der Entwicklung der Software, die diese Transformation erst möglich macht.

(2) Auf der untersten Ebene der Dekonstruktion existieren keine qualitativen Unterschiede zwischen den digitalen Repräsentationen verschiedener Objekte; alle werden durch Wörter in der Maschinensprache dargestellt. Die Regeln für die Manipulation analoger Vorlagen beruhen auf unvereinbaren natürlichen Gesetzmäßigkeiten: die Manipulation von Tönen funktioniert anders als die Manipulation von Farben, eins kann nicht durch das andere ersetzt werden und die Regeln der Akustik haben mit den Regeln der Optik nichts zu tun. Die Manipulation aller digitalen Objekte hingegen folgt letztendlich nur den Regeln der digitalen Datenverarbeitung (a.a.O., S. 192).

Einerseits können dadurch Veränderungen an digitalen Objekten durchgeführt werden, die mit ihren analogen Vorlagen unmöglich wären, z. B. bei der Bearbeitung digitaler Bilder.[51] Andererseits entstehen dadurch neue Möglichkeiten, weitreichende gesellschaftliche Kontrolle auszuüben: Während über den Umgang mit analogen Objekten in unterschiedlichen gesellschaftlichen Bereichen verhandelt wird (Arthur 2011), erstreckt sich der Einfluss von z. B. Standards der Datenverarbeitung potentiell auf digitale Objekte, die in allen Bereichen eingesetzt werden. Über die Regulation der Datenverarbeitung wird damit potentiell Kontrolle über soziale Prozesse in vielen gesellschaftlichen Bereichen ausgeübt. Informatiker*innen und andere Expert*innen für Datenverarbeitung gewinnen dadurch weitreichende Einflussmöglichkeiten in der Gesellschaft.

Software kann soziale Prozesse zwar zum Teil durch automatische ersetzen, doch auch diese bleiben eingebettet in ein Geflecht an sozialen Prozessen.

[51] Jeder Aspekt digitaler Bilder kann mit der richtigen Software so verändert werden, dass es zumindest für die meisten Menschen nicht möglich ist, zu erkennen, welche Aspekte des digitalen Bildes auf eine analoge Vorlage zurückzuführen sind. Die Tatsache, dass digitale Bildbearbeitung inzwischen Voraussetzung für technologische Entwicklungen in unterschiedlichen gesellschaftlichen Bereichen wie Medizin, Militär und Wissenschaft ist, bezeichnet Kallinikos als *„indicative of the paradigmatic shift beyond visual unities in a field and social practice that predominantly deal with images presented and consumed by human perception“*(Kallinikos 2009, S. 185).

Der Umgang mit Software bleibt daher immer zu einem gewissen Teil unsicher (a.a.O., S. 191 f.). Der Umgang mit digitalen Objekten wird jedoch auf andere Weise und in anderen sozialen Prozessen reguliert als der Umgang mit ihren analogen Vorlagen. Durch die Undurchschaubarkeit der digitalen Dekonstruktion entzieht sich diese Regulation der Kontrolle der Akteur*innen, die Software benutzen (Kallinikos und Hasselbladh 2009). Kallinikos behauptet, dass diese Undurchschaubarkeit auch dazu führt, dass Akteur*innen ihre Erwartungen über das, was mit analogen Objekten möglich ist, an die Möglichkeiten anpassen, die digitale Objekte bieten. Die ubiquitäre Verbreitung von Software würde damit mit der Zeit grundsätzlich verändern, wie Menschen auf die Welt Einfluss nehmen (Kallinikos 2009). Diese Vorhersage passt zu aktuellen gesellschaftlichen Debatten über die Folgen der Digitalisierung[52] ebenso wie zu Erkenntnissen aus verschiedenen Bereichen der soziologischen Digitalisierungsforschung (Grenz und Möll 2014; Kropf und Laser 2019).[53] Belege dafür, dass die beobachtbaren Veränderungen der *„perceptive and action modalities by which human agents confront the world*" (Kallinikos 2005, 2009) auf die von ihm beschriebenen Besonderheiten der digitalen Dekonstruktion zurückzuführen sind, bleibt er jedoch schuldig.

In allen hier vorgestellten Arbeiten wird Software als Artefakt betrachtet, dessen Komplexität sich (zumindest zum Teil) aus der Tatsache ergibt, dass soziale Phänomene in mathematischen Regeln folgende Modelle übersetzt werden und dass Akteur*innen in sozialen Prozessen mit den Ergebnissen dieser Übersetzung umgehen, ohne genau nachvollziehen zu können, wie diese Ergebnisse zu Stande gekommen sind. Alle Arbeiten weisen darauf hin, dass die Übersetzung den Akteur*innen Teile der Kontrolle über den Nutzungsprozess entzieht und diese in die Prozesse verlagert, in denen Software entsteht. Wie in Abschnitt 2.1.2 dargestellt, sind diese Prozesse, in denen Software entwickelt und weiterentwickelt wird, in der Zwecke und Rahmenbedingungen von Softwareprojekten ausgehandelt werden, in denen technische Standards entstehen etc., auf vielfältige Weise voneinander abhängig. Diese Abhängigkeit zwischen den verschiedenen Phasen

[52] Nur ein Beispiel: Über den Einfluss von digital bearbeiteten Fotos auf problematische Körperbilder bei Jugendlichen wird seit längerem sowohl in der Öffentlichkeit (vgl. z. B. Großegger 2017) als auch in verschiedenen Kultur- und Humanwissenschaften diskutiert (siehe z. B. Dakanalis et al. 2014).

[53] Die Arbeiten in beiden zitierten Sammelbänden untersuchen, wie sich die Handlungspraktiken in verschiedenen gesellschaftlichen Bereichen durch den Umgang mit Software verändern, untersuchen aber nicht, ob es einen systematischen Zusammenhang zwischen den Veränderungen der Handlungspraktiken und den Eigenschaften der Software gibt, die sie als deren Verursacherin benennen.

des Softwarelebenszyklus und den Prozessen, die den Umgang mit Software umrahmen, bildet die zweite Dimension der Komplexität von Software. Auch sie ist Gegenstand soziologischer Forschungsarbeiten, die zentrale Erkenntnisse für eine Soziologie der Software bringen. Sie werden im folgenden Abschnitt vorgestellt.

2.2.2 Arbeiten zum Softwarelebenszyklus

Die Darstellung des Softwarelebenszyklus in Abschnitt 2.1.2 hat gezeigt, dass der Umgang mit Software von einer Kette von Modellierungsprozessen abhängt. Deren Verlauf und Ergebnis wird von den beteiligten Akteur*innen, ihrer Einbettung in Organisationen, professionelle Gemeinschaften und andere soziale Systeme sowie den Rahmenbedingungen beeinflusst wird, die von den vorhergehenden Modellierungsprozessen gesetzt werden. Diese Einflüsse strukturieren den Handlungsspielraum, den Akteur*innen im Umgang mit Software haben, ohne ihn völlig aufzuheben.

2.2.2.1 Spezifikation: Konfliktreiche Modellierung mit professionellem Machtgefälle

Ausgiebig untersucht sind vor allem die Prozesse, in denen Stakeholder*innen und Softwareentwickler*innen versuchen, gemeinsam die Unsicherheit sozialer Phänomene so weit zu reduzieren, dass ein daraus resultierendes Modell mit minimaler Beteiligung der Stakeholder*innen weiterverarbeitet werden kann. Softwareentwickler*innen nutzen in diesen Prozessen professionsspezifische Repräsentationspraktiken (Bloomfield und Vurdubakis 1997), Prozessvorlagen wie Vorgehensmodelle und Software-Werkzeuge, also Software für Softwareentwickler*innen, die zur Unterstützung solcher Praktiken entwickelt wurden (Schulz-Schaeffer und Bottel 2017). Funken (2001) und Rammert et al. (1998) betrachten die Modellierung als Form der Wissensproduktion. Dabei konzentriert sich Funken auf die Unterschiede zwischen den beiden Akteur*innengruppen und beschreibt die Aufgabenanalyse als einen asymmetrischen Aushandlungsprozess. In diesem Prozesse generieren dominante Profis (Softwareentwickler*innen, alternativ auch IT-Berater*innen) mit Lai*innen (Nutzer*innen) Wissen über den durch Software zu unterstützenden Ablauf. Entsprechend des Machtgefälles wird das Ergebnis dieser Wissensproduktion entscheidend durch die Hintergrundannahmen, also das implizite bzw. habitualisierte Erfahrungswissen der Softwareentwickler*innen geprägt. Diesen Hintergrundannahmen zu Folge geht es bei der Aufgabenanalyse darum, formalisierbares Wissen über eine als messbar

und problemlos strukturierbar angesehene Wirklichkeit zu generieren. Diese Hintergrundannahmen führen dazu, dass die Softwareentwickler*innen versuchen, einige Aspekte der Unsicherheit, die Stakeholder*innen in den Prozess einbringen, auszublenden, um grundlegende Eigenschaften des Gegenstands aus dem Modell auszuschließen:

> *„Da bei der Software-Entwicklung bestehende soziale und organisatorische Verhältnisse in digitale Informationssysteme übersetzt und strukturiert werden, sind Entwickler auf Insiderwissen und fachspezifisches know-how angewiesen, das häufig nicht generalisierbar ist. 'Standortgebundenheit' und 'Parteilichkeit' sind bestimmend für die Charakterisierung des Gegenstandes." (Funken 2001)*

Während Funken das Machtgefälle zwischen Softwareentwickler*innen und Stakeholder*innen thematisiert, konzentrieren sich Rammert et al. (1998) darauf, *wie* aus den Erfahrungen der Stakeholder*innen formalisierbares Wissen über den Gegenstand entsteht. Die Orientierungen und Fähigkeiten der Expert*innen für den Gegenstandsbereich (Stakeholder*innen) unterscheiden sich stark von denen der Wissensingenieur*innen (Softwareentwickler*innen): Erstere besitzen zwar tiefgreifendes Wissen über die für die Softwareentwicklung relevanten Eigenschaften des sozio-technischen Systems, aber dieses liegt *„wenn überhaupt dann nur teilweise in einer kodifizierten Form (…) vor. Häufig handelt es sich um soziale Praktiken, die erst noch verbalisiert oder in eine andere symbolische Form gebracht werden müssen."* (Rammert et al. 1998). Bei der Verbalisierung ihrer Praktiken stellen Expert*innen für den Gegenstandsbereich den Einzelfall und die konkrete Situation seiner Behandlung in den Mittelpunkt. Innere Widersprüche der Darstellung sind für sie – anders als für die Softwareentwickler*innen – wenig relevant, da sie sie als unvermeidlichen Teil der Praxis begreifen. Im Aufeinandertreffen dieser unterschiedlichen „Logiken" werden Informationen über das in der Spezifikation zu modellierende System nicht von den einen Experten*innen an die anderen übermittelt (wie es die rationalistische Tradition vorsieht), sondern in der gemeinsamen Interaktion erst als relevante Informationen konstruiert. Die Konstruktion ist das Ergebnis einer Aushandlung, verstanden als ein Prozess, in dem *„zwei Gesprächspartner zu einer, relativ zu ihren gegenwärtigen Zwecken, hinreichenden Übereinstimmung der Deutungen von Ereignissen, die in dem jeweiligen Kontext relevant sind, kommen"* (a.a.O., S. 140). Wie schwierig es in der Praxis sein kann, eine solche Übereinstimmung zu erzielen, zeigen Susan Leigh Star und Karen Ruhleder (1996) in ihrer Studie über die Entwicklung, Anwendung und Weiterentwicklung eines internetbasierten Softwaresystems zur wissenschaftlichen Kollaboration. Unterschiedliche professionelle Hintergrundannahmen können den beiden Forscherinnen zu Folge dazu führen,

dass Akteur*innen bei der Softwareentwicklung aneinander vorbeireden, ohne es zu bemerken.

Während Funken, Rammert et al. und Star und Ruhleder sich mit der Reduktion der Aspekte von Unsicherheit befassen, die auf unterschiedlichen professionellen Orientierungen von Softwareentwickler*innen und Stakeholder*innen beruhen, zeigen Edeltraud Egger und Ina Wagner (1993), dass es auch innerhalb dieser Gruppen verschiedene Vorstellungen von dem sozialen Phänomen gibt, die im Softwareentwicklungsprozess auf ein einheitliches Modell reduziert werden müssen. In der Studie, für die die Forscherinnen die Entwicklung einer Individualsoftware zur OP-Planung begleitet haben, wird herausgearbeitet, wie die unterschiedlichen Positionen, die Chirurg*innen, Pflegekräfte und Anästhesist*innen in der professionellen Hierarchie der Medizin einnehmen, die Spezifikationsphase beeinflussen. Angehörige aller drei Berufsgruppen arbeiten gemeinsam an der Operation von Patient*innen im Krankenhaus. Sie sind für unterschiedliche Aufgaben im Operationsprozess zuständig und haben folglich unterschiedliche Vorstellungen von der Gesamtdauer einer Operation (a.a.O., S. 262). Gleichzeitig werden den Berufsgruppen in der professionellen Hierarchie unterschiedliche Grade von Autonomie zugestanden. Dass die an der Studie beteiligten Akteur*innen ihre jeweiligen Ansprüche kennen und anerkennen, zeigt sich zum Beispiel an den unterschiedlichen Erwartungen, die Mitglieder der verschiedenen Berufsgruppen an die Terminplanungsfunktion der zu entwickelnden Software stellen: Während öffentlich einsehbare persönliche Kalender für Pflegekräfte und Anästhesist*innen akzeptable technische Hilfsmittel zur Terminabstimmung darstellen, werden sie von Chirurg*innen als unzumutbare Beschneidung ihrer Freiheit abgelehnt. Nur Chirurg*innen empfinden mögliche Diskussionen über ihren Terminplan als Eingriff in ihre Autonomie. Für Pflegekräfte und Anästhesist*innen ist es hingegen Alltag, dass andere Einfluss darauf nehmen, wann sie welche Aufgaben erfüllen.

In den bisher vorgestellten Arbeiten wird vor allem untersucht, wie sich die professionelle Orientierung und die persönlichen Erfahrungen der Akteur*innen darauf auswirken, welche Unsicherheiten sie im Modellierungsprozess wie reduziert sehen wollen und welche Einflussmöglichkeiten sie innerhalb dieses Prozesses haben. Beides hängt jedoch nicht nur von den individuellen Eigenschaften der Akteur*innen ab, die an der Modellierung mitwirken, sondern auch von den beteiligten Organisationen. Die Orientierungen, Interessen und Erfahrungen von Softwareentwickler*innen und Stakeholder*innen der Softwareentwicklung wird nicht nur durch ihre Einbettung in professionelle Gemeinschaften geprägt, sondern auch durch die Zugehörigkeit zu Organisationen.

Softwareentwickler*innen sind zum Beispiel häufig Beschäftigte eines Softwareunternehmens, Stakeholder*innen oft Beschäftigte eines anderen Unternehmens. In Abschnitt 2.1.2 wurde dargestellt, dass diese Zugehörigkeit zu Organisationen beeinflusst, wie Akteur*innen sich in den Aushandlungen rund um die Softwareentwicklung verhalten: Welche Zwecke durch Softwareentwicklungsprojekte erfüllt werden sollen, welche Ressourcen in diesen Projekten zur Verfügung stehen, wie Akteur*innen ihre Positionen in den unterschiedlichen Aushandlungen der Softwareentwicklungsprozesse legitimieren können und welche Machtmittel sie haben, um diese Positionen gegeneinander durchzusetzen – dies alles wird auch von den Organisationen geprägt, die an der Softwareentwicklung beteiligt sind. In der soziologischen Forschung kommt diese Bedeutung von Organisationen nur am Rande vor. Im Folgenden stelle ich dar, in welchen Arbeiten die Rolle von Organisationen in den einzelnen Phasen des Softwarelebenszyklus untersucht worden ist.

2.2.2.2 Das Softwareprojekt: Komplexe Prozesse mit Folgen für das Produkt

Mit der Tatsache, dass Projekte, in denen Individualsoftware entwickelt wird, von den sozialen Prozessen in den Organisationen, die diese Projekte tragen, abhängig sind, befassen sich Friedrich Weltz und Rolf Ortmann (1992). Sie zeigen, dass die Trägerorganisationen von der Initialisierung bis zur Evaluation beeinflussen, welche Aufgaben im Projekt von wem wie ausgeführt werden, wer diese Aufgaben nach welchen Maßstäben bewertet und wie die Ergebnisse vorhergehender Prozesse in folgenden aufgenommen werden. Nicht nur die Auswahl der Teammitglieder, auch die Rahmenbedingungen, unter denen diese arbeiten, beschreiben die Autoren als Ergebnis oft konfliktreicher sozialer Prozesse innerhalb der Organisationen, in denen die Komplexität der Aufgaben oft systematisch unterschätzt wird:

> *„Unverkennbar war die Kalkulation vielfach ein Politikum: Instrument in der Auseinandersetzung um den Umfang oder gar die Berechtigung des Entwicklungsprojekts […] Bisweilen wurde das Projektvolumen ganz offensichtlich und bewusst zu niedrig angesetzt, um die interne Durchsetzung des Projekts zu erleichtern. […] Nicht die Kostenkalkulation lieferte in solchen Fällen Kriterien für die Entscheidungen, sondern umgekehrt die ‚Berechnungen' orientierten sich an den Kriterien der Entscheidung." (Weltz und Ortmann 1992, 43 f.)*

Obwohl die zu Projektbeginn gesetzten Rahmenbedingungen sich während der Projektlaufzeit regelmäßig ändern, wenn Ereignisse im Projekt Anlass dazu

geben, die Bedingungen neu auszuhandeln (a.a.O., S. 46), beeinflussen sie in vielen der untersuchten Fälle dennoch, wie die Software aussieht, die im Rahmen des Projekts entsteht.

> *„Ob eine opulente Lösung mit einem Übermaß an Funktionalitäten angeboten wurde oder ein karges Miniprogramm, in welcher Weise der Anwendungsbezug, ergonomische Aspekte etc. berücksichtigt wurden, all dies hing bisweilen nicht so sehr von den Zielvorgaben eines Projekts oder von der Planung der einzelnen Arbeitsschritte ab, sondern schlicht auch davon, welcher zeitliche und budgetäre Rahmen mit welcher Verbindlichkeit vorgegeben wurde." (Weltz und Ortmann 1992, S. 47)*

Mit dem Fokus auf das Verhältnis von Organisationen und Projekten gelingt es Weltz und Ortmann, prinzipiell sichtbar zu machen, dass sich die Komplexität von Software aus soziologischer Perspektive auf die Vielzahl an Abhängigkeiten zwischen den sozialen Prozessen zurückführen lässt, die Einfluss auf die Softwareentwicklung haben. Die Autoren unterscheiden dabei konzeptionell zwischen struktureller Komplexität und prozessualer Komplexität. Strukturelle Komplexität ergibt sich dieser Unterscheidung zu Folge aus der Aufgabe und den Kontextbedingungen, prozessuale Komplexität entsteht daraus, dass im Softwareentwicklungsprozess getroffene Entscheidungen einerseits Bedingungen für spätere Entscheidungen setzen, andererseits aber auch dazu führen können, dass die Bedingungen rückwirkend angepasst werden (a.a.O., S. 16). Aus der in Abschnitt 2.1.2 dargestellten Perspektive dienen Entscheidungen im Softwareentwicklungsprozess häufig dazu, Unsicherheiten bei der Modellierung unter Kontrolle zu bekommen, indem aus verschiedenen möglichen Alternativen nur eine ausgewählt und in das Modell übernommen wird. Mit dem Begriff der prozessualen Komplexität bezeichnen Weltz und Ortmann eine zentrale Beschränkung dieser Strategie, die auch in Abschnitt 2.1.2. erläutert wird: Durch Entscheidungen kann die Unsicherheit niemals aufgelöst, sondern nur die Kontrolle von Unsicherheiten zwischen den einzelnen Prozessen verschoben werden.

2.2.2.3 Einflüsse von Organisation auf den Code

Während sich Weltz und Ortmann vor allem mit den Abhängigkeiten zwischen sozialen Prozessen in der Kund*innenorganisation und denen im Softwareprojekt beschäftigen, untersuchen Graham Button und Wes Sharrock, wie die Prozesse bei Softwareherstellerinnen Arbeit und Arbeitsergebnisse von Softwareentwickler*innen beeinflussen (Button und Sharrock 1998, 1996). Auch diese Prozesse bestimmen grundlegend mit, welche Art von Software entsteht: *„The application of technical knowledge could not be divorced from the organizational contingencies within which it was deployed; determinations of the technical*

design of the machine were entwined with organizational decisions" (Button und Sharrock 1998). Auch in den von den Autoren untersuchten Fällen ist die Bedeutung scheinbar offensichtlicher Rahmenbedingungen wie Zeitpläne und Budgets eines Entwicklungsprojekts von den Ergebnissen konkreter Aushandlungen in der Organisationen zur Laufzeit des Projektes abhängig. Entwicklungsprojekte, die zu lange dauern oder zu teuer werden, werden zwar gelegentlich ergebnislos beendet (Button und Sharrock 1998). Es kommt aber auch vor, dass Entwickler*innen sich entscheiden, einige der für die Software spezifizierten Anforderungen zu ignorieren, wenn diese nicht innerhalb der allgemeinen Vorgaben der Organisation erfüllbar sind, zum Beispiel weil für die Umsetzung der Einsatz leistungsstärkerer Hardware oder einer von den Richtlinien der Organisation nicht vorgesehene Programmiersprache nötig wäre (Button und Sharrock 1998). Häufig arbeiten die Entwickler*innen an mehreren Projekten gleichzeitig, so dass Verzögerungen dazu führen, dass die Projektverantwortlichen mit denen konkurrierender Projekte um Personal verhandeln müssen (Button und Sharrock 1998). Die Möglichkeit, eine Software wie spezifiziert fertig zu stellen, hängt damit auch davon ab, welche Mittel, welche Gelegenheiten und auch welche Motivation die Projektverantwortlichen haben, um die Bedarfe, die innerhalb des Projekts ausgehandelt werden, außerhalb des Projekts gegenüber der Trägerorganisation durchzusetzen.

Button und Sharrock thematisieren auch, dass aus der grundsätzlichen Unsicherheit, die im Vorgang des Programmierens liegt, neue Unsicherheiten für die Phase der Implementierung entstehen. Diese Phase kann nur erfolgreich abgeschlossen werden, wenn es gelingt, die Arbeitsergebnisse aller einzelnen Programmierer*innen zu integrieren. Unsicherheiten über die Details der individuellen Arbeitsergebnisse stehen dem im Weg:

> *„There are no unique solutions for the coding problems engineers have at hand. Any given coding problem can have many solutions: the problem-solving code written by one engineer will not necessarily be the same as that written by another engineer facing the same problem, even though both sets of code would execute the desired operation. Thus different engineers might 'work out the code' in different ways. In any software development project, the code written by one engineer may not 'fit' that written by another."* (Button und Sharrock 1998, S. 78)

Die Autoren zeigen mit ihren Studien zur Implementierungsphase, dass Organisationen den Umgang mit Unsicherheiten bei der Modellierung im Code beeinflussen, ohne dadurch die Unsicherheiten, die aus der Aufgabe selbst stammen, aufzuheben. Die Übersetzung von sozialen Phänomenen in Code und die organisatorischen Rahmenbedingungen der verketteten Modellierungsprozesse tragen beide auf je eigene Weise zur Komplexität von Software bei.

2.2.2.4 Organisatorisch eingebettete Aushandlungen in der Praxis des Testens

Während Spezifikation häufig, Projektmanagement, Entwurf und Implementierung zumindest gelegentlich Gegenstand soziologischer Untersuchungen sind, gibt es nur wenige Forschende, die sich mit den sozialen Prozessen während der Testphase auseinandersetzen. Einen Einblick in die Praxis des Softwaretests bieten immerhin John Rooksby, Mark Rouncefield und Ian Sommerville (2009), die sich in vier kurzen Feldstudien mit der Frage beschäftigen, wie die Praxis des Testens unter unterschiedlichen organisatorischen und technischen Bedingungen abläuft. Die Arbeit beinhaltet eher anekdotische Eindrücke vom Softwaretest in einem kleinen Softwarehaus, zwei Open-Source-Projekten, wovon eines unter Mitarbeit eines Unternehmens entwickelt wird, und einer Eigenentwicklung in einer großen Universität. Alle Beispiele gleichen sich darin, dass die Akteur*innen in der Praxis vom idealisierten Modell des Softwaretests abweichen. Diesem idealisierten Modell zu Folge sollten zu Beginn der Entwicklung Fehlerklassen definiert und festgelegt werden, wie viele Fehler aus jeder dieser Klassen am Ende des Projekts behoben sein müssen, damit die Software als erfolgreich fertiggestellt gelten kann. In keinem der beschriebenen Fälle berücksichtigen die Beteiligten eine solche Vorgabe oder folgen einem vorher fixierten Plan, mit dessen Hilfe alle Anforderungen überprüft werden. (a.a.O., S. 573). Stattdessen werden Ausmaß, Umfang und Art der Tests während der Testphase ebenso ausgehandelt wie die Frage, ob und unter welchen Bedingungen gefundene Fehler tatsächlich behoben werden sollen: *„conversation is a key part of the work of testing, between developers, between board members, between users etc."* (a.a.O. S. 576). Der Verlauf und das Ergebnis der Aushandlungen ist abhängig von den Ressourcen des Softwareprojekts und den Prioritäten und Einschätzungen der konkreten Teilnehmer*innen: *„there is congruence between organizational structure, organizational priorities and the way tests are performed"* (Rooksby et al. 2009, S. 575). Nicht nur bedeutet das, dass Funktionen der Software, die von den Anforderungen abweichen, unbemerkt bleiben, weil sie im Test nicht geprüft werden, sondern auch, dass erkannte Abweichungen bestehen bleiben, weil sie während der Testphase als nicht (unmittelbar) signifikant eingeschätzt werden. Diese Einschätzung ist ein zentraler Aspekt aller untersuchten Beispiele. Nicht nur die Spezifikation, sondern auch der Test *„involves reasoning and speculation about practices and situations of use"* (a.a.O., S. 576). Ohne Anspruch auf Verallgemeinerbarkeit zeigt die Arbeit von Rooksby et al. damit, dass auch beim Test Modellierungsprozesse stattfinden, die sowohl mit den übrigen Prozessen im Lebenszyklus der Software als auch mit deren (organisatorischer) Einbettung verknüpft sind.

2.2.2.5 Customizing von standardisierter Organisationssoftware

Im Rahmen ihrer organisationssoziologischen Dissertation zu SAP untersucht Hannah Mormann (2016) wie in Customizingprojekten generische Standardsoftware und spezifische Kund*innenorganisation aneinander angepasst werden. Sie analysiert dazu die Abhängigkeiten zwischen der Software, den Entscheidungen innerhalb der Organisationen, die das Customizingprojekt tragen, und den Interaktionen von Berater*innen und ausgewählten Mitarbeiter*innen der Kund*innenorganisation während des Projekts. Die Software charakterisiert sie dabei als Werkzeug zur *„(Re-)Modellierung von Arbeitsabläufen"* (a.a.O., S. 131), das auf einem Organisationsmodell beruht, das frei von Unsicherheiten ist (a.a.O., S. 225). Aus dieser Annahme leiten die Akteur*innen die Überzeugung ab, dass die in der Software vorgefundenen Vorlagen für Arbeitsprozesse *„vernünftig"* sind (a.a.O., S. 173). Berater*innen gelten daher sowohl als Expert*innen für die Software als auch als Expert*innen für den Gegenstandsbereich Es wird davon ausgegangen, dass die Kenntnis der Vorlagen und Erfahrungen aus anderen Customizingprojekten ebenso Grundlage einer solchen Expertise sein kann wie praktische Erfahrung mit den Arbeitsprozessen selbst (a.a.O., S. 171). Die Mitarbeiter*innen der Kund*innenorganisation, die am Customizing teilnehmen, sind oft Mitglieder höherer Hierarchieebenen, die wenig Bezug zu den konkreten Alltagsabläufen haben (Mormann 2010). Als solche wissen sie entweder nicht im Detail, wie die Arbeitsprozesse in ihrer Organisation aussehen, oder sind mehr daran interessiert, diese mit Einführung der Software zu verändern, als sie beim Customizing abzubilden (Mormann 2016). Die Einführung von standardisierter Organisationssoftware wird nämlich häufig als Gelegenheit genutzt, um lang etablierte Arbeitsprozesse, die als ineffizient gelten, entsprechend der Vorlagen aus der Software umzugestalten und damit vermeintlich zu optimieren. Begrenzte Projektlaufzeiten, die weitere technische Veränderungen der Software unmöglich machen (Mormann 2016) tragen zusätzlich dazu bei, dass die Berater*innen im Customizingprojekt vor allem versuchen, von den Kundenrepräsentant*innen zu erfahren, welche Kombination von Standardvorlagen aus der Software am ehesten ihren Vorstellungen entspricht (a.a.O., S. 182). An diese Kombination sollen dann die Arbeitsprozesse angepasst werden.

Neil Pollock und James Cornford stellen in ihrer Studie zum Customizing eines ERP-Systems heraus, dass beim Customizing Teile des Standardmodells gelegentlich auch übernommen werden, ohne dass irgendeine der Akteur*innen inhaltlich von ihnen überzeugt ist. In solchen Fällen bietet die Übernahme der Standardvorlagen den Projektmitgliedern eine Möglichkeit, das Projekt voranzubringen, obwohl dazu eigentlich notwendige Vorgaben aus verknüpften sozialen Prozessen (noch) nicht verfügbar sind:

> *„At a number of points in the process it becomes clear that there is more than one way, in current practice, in which a particular step in the process was being handled. If the issue cannot be resolved one way or another, the consultant leading the meeting identifies the issue as "a matter for policy", a matter on which a definitive ruling must be given by the university centrally. […] [T]hese demands for policy were so copious that many of the requests simply remained on the database without senior management ever having the time to deal with them. […] As a result, most of the issues have to be resolved within the project team on more technical grounds, meaning the team had to deploy its own criteria while configuring the system. And, in many cases, this meant just accepting the default settings." (Pollock und Cornford 2004, S. 41–42)*

Im zitierten Beispiel versucht der Beratende, Unsicherheiten, die im Customizingprozess sichtbar werden, an die Entscheidungsprozesse im oberen Management auszulagern. Die Unsicherheiten werden aber innerhalb dieser Prozesse nicht bearbeitet, da die Mitglieder der Universitätsleitung sich gar nicht mit den Anfragen aus dem Projektteam befassen. Sie bleiben damit unsicher und blockieren irgendwann die Modellierung im Customizingprozess. Da es den Akteur*innen aber weiterhin nicht gelingt, die Unsicherheiten durch Aushandlungen selbst zu reduzieren, nutzen sie das Standardmodell aus der Software. Die Software wird damit performativ, weil sie eine Möglichkeit anbietet, die Komplexität der sozialen Prozesse zu reduzieren, an denen die Akteur*innen selbst beteiligt sind. Statt sich zu einigen oder bei der Universitätsleitung vorstellig zu werden und eine Entscheidung zu verlangen, können die Mitarbeiter*innen des Customizingprojekts einfach übernehmen, was die Software bereits anbietet. Die Komplexität, die dem Standardmodell zugrunde liegt, ist für die Akteur*innen im Customizing nicht von Belang.

In der soziologischen Forschung zur Softwareentwicklung wird nur selten thematisiert, dass Softwareentwicklung durch die Organisationen beeinflusst wird, in denen sie stattfindet. Die hier vorgestellten Arbeiten wurden mit Blick auf dieses Thema ausgesucht; ihr Fokus auf Organisationen stellt eine Ausnahme im Forschungsstand dar. Anders sieht dies bei Studien aus, in denen die Forschenden sich mit den Phasen des Lebenszyklus beschäftigen, die nach der Fertigstellung der ersten Version der Software liegen. Hier werden in der Regel Organisationen fokussiert, die Frage nach der Performativität der Software bleibt hingegen meist unbeantwortet oder wird gar nicht erst gestellt. Einige dieser Studien sind jedoch entscheidend, um die in dieser Arbeit entwickelte soziologische Perspektive auf Software zu verstehen. Dieser zu Folge entsteht die Performativität von Software ja nicht allein im Softwareentwicklungsprozess, sondern ebenso in den sozialen Prozessen, in denen Akteur*innen – meist in irgendeiner Form beeinflusst von Organisationen – mit einer fertiggestellten Version der Software umgehen.

Software wird dabei Teil der mikropolitischen Auseinandersetzungen im Nutzungskontext (Abschnitt 2.2.2.6) und hat Auswirkungen auf die Praktiken und Regeln, welche diesen Kontext strukturieren (Abschnitt 2.2.2.7).

2.2.2.6 Einführung und Nutzung von Software als Geflecht mikropolitischer Auseinandersetzungen

Mit der Einführung von Software im Rahmen von IT-Beratungsprojekten befasst sich die Arbeit von Michaela Wieandt-Ledebur (2014). In IT-Beratungsprojekten arbeiten Mitarbeiter*innen einer Beratungsorganisation mit Mitarbeiter*innen einer Kund*innenorganisation zusammen, meist um neue Software in der Kund*innenorganisation einzuführen. IT-Beratungsprojekte werden immer von der Kund*innenorganisation beauftragt. Wieandt-Ledebur betrachtet IT-Beratungsprojekte als besondere soziale Systeme, als Organisationen mit von vorneherein begrenzter Laufzeit,[54] die von zwei Organisationen ohne eine solche Laufzeitbegrenzung abhängen. Solche Projekte versteht sie als *„Relaissysteme“* (a.a.O., S. 49) zwischen Beratungs- und Kund*innenorganisationen: Über die sozialen Prozesse innerhalb des Projekts werden soziale Prozesse innerhalb der beiden Organisationen, die das Projekt tragen, miteinander verknüpft. Auf Basis der strategischen Organisationsanalyse (Crozier und Friedberg 1979) entwickelt sie einen Forschungsrahmen, den sie einsetzt, um zu untersuchen, wie die Organisationsmitglieder versuchen, Kontrolle über die Regulierung eines internationalen IT-Beratungsprojekts zu gewinnen. Es wird nachgezeichnet, dass die Akteur*innen mit Hilfe dieser Kontrollversuche ihre Positionen in den Träger*innenorganisationen zu stärken, in denen sie beschäftigt sind. Die mikropolitische Betrachtung zeigt, dass die Dynamik der Aushandlungen innerhalb solcher Projekte nur vor dem Hintergrund der vielfältigen Aushandlungen verstanden werden kann, die sich in den Träger*innenorganisationen um Aufgaben, Rahmenbedingungen und Folgen des Projekts entwickeln. Bei Projekten zur Softwareeinführung entstehen besonders viele Verknüpfungen zu sozialen Prozessen in den Träger*innenorganisationen, die intensive Aushandlungen auf allen Seiten nach sich ziehen, weil die Projekte *„die IT-Strukturen und damit Ungewissheitszonen und Machtbeziehungen in Organisationen tatsächlich und dauerhaft verändern“* (Wieandt-Ledebur 2014). Mitarbeiter*innen der Kund*innenorganisation versuchen daher, Kontrolle über das Projekt auszuüben, in dem über die zukünftigen Bedingungen ihres Handelns entschieden wird. Software wird in der Arbeit von Wieandt-Ledebur als Gegenstand der Auseinandersetzung und Mittel zur Kontrolle der sozialen Prozesse in Projekt und

[54] vgl. dazu auch die Ausführungen zu Projekten in Abschnitt 2.1.2.2.

Organisation betrachtet. Nicht untersucht wird sie als performatives Element der Prozesse, das eigene Modelle der sozialen Prozesse mit sich bringt, deren Umgestaltung bei der Einführung der Software mit verhandelt wird.

Eine wichtige Vorlage für Wieandt-Ledeburs Theorierahmen ist die umfassende Arbeit zu den Bedingungen und Folgen von Informatisierung und Reorganisation im Betrieb von Günther Ortmann, Arnold Windeler, Albrecht Becker und Hans-Joachim Schulz (1990). In der Arbeit kommt ebenfalls die strategische Organisationsanalyse zum Einsatz; diese Herangehensweise wird jedoch nicht nur zur Analyse der empirischen Ergebnisse genutzt, sondern auch sozialtheoretisch reflektiert und weiterentwickelt. Die Autoren betrachten Einführung und Nutzung von Software gemeinsam. Sie zeigen anhand mehrerer detaillierter Fallstudien, wie stark Softwarenutzung zumindest in Organisationen durch bestehende Machtverhältnisse und Aushandlungen zwischen einflussreichen Akteur*innen beeinflusst wird. Das beginnt bereits lange vor der Einführung von Software damit, dass bestimmte Phänomene als zu lösende Probleme erst bestimmt und spezifisch ausformuliert werden müssen.

> *„‚Probleme' […] sind präformiert durch organisationale Standards und Normen, durch Identifizierungen mit partiellen Organisationszielen, durch eine organisationale Denk- und Sprechweise […] und sie sind gefiltert durch ein Raster mikropolitischer Interessen und Mittel, das die Chancen für Sachverhalte, zu anerkannten Problemen zu avancieren, ungleich verteilt." (Ortmann et al. 1990)*

Die Problemdefinition ist nur eine von unzähligen Entscheidungen, mit denen innerhalb der Organisationen Unsicherheit reduziert wird, bevor die Nutzung von Software überhaupt beginnt (a.a.O., S. 377 ff.). Diese Entscheidungen folgen nicht einer im Nachhinein kolportierten ökonomischen oder technischen Rationalität (a.a.O., S. 408). Sie entstehen aus den von asymmetrischen Machtverhältnissen geprägten Aushandlungen zwischen Akteur*innen, deren Ziele, Perspektiven und Machtressourcen auch durch ihre Positionierung in den Organisationen bestimmt sind, denen sie angehören. Ortmann et al. verstehen Organisationen dabei als *„Arenen mikropolitischer Aushandlungsprozesse und Kämpfe, in denen jeder ‚sein' Spiel spielt und das Ganze nur funktioniert, wenn die Spiele in Organisationen günstig strukturiert und aneinander gegliedert sind."* (a.a.O., S. 8). Die formale Hierarchie ist in solchen Aushandlungen nur eine von mehreren Machtressourcen und auch den auf den ersten Blick unterlegenen Akteur*innen bieten sich im Laufe der Projekte der Softwareeinführung ausreichende Gelegenheiten, eigene Interessen zumindest zum Teil umzusetzen (a.a.O., S. 399 ff.). Die Akteur*innen machen Software zur Ressource, um ihre Vorstellungen von der

Organisation durchzusetzen (z. B. a.a.O., S. 296–324) oder nutzen ihre (zugeschriebene) Expertise in Bezug auf Software, um unliebsame Veränderungen zu verhindern (a.a.O., S. 266–295).

Software spielt in den internen Aushandlungsprozessen, die Ortmann et al. betrachten, nicht nur als Mittel zur Durchsetzung bereits bestehender (Macht)-Interessen eine Rolle. Die Möglichkeit, Software selbst zu entwickeln statt ein fertiges Standardprodukt zu kaufen, wird durch die Rahmenbedingungen begrenzt, die die Organisation setzt (z. B. verfügbares Personal und Kostenrahmen). Die Entscheidung für Standardsoftware bedeutet, dass die Akteur*innen an ein gegebenes Angebot gebunden sind. Extern produzierte Software wiederum trägt Modelle von Arbeitsorganisation in sich, die im Rahmen von Aushandlungen in und zwischen anderen Organisationen entstanden sind (a.a.O., S. 409 ff.). Über die Software entsteht damit eine Verbindung von zwischen- und innerorganisatorischen Aushandlungsprozessen, denn die Software trägt dazu bei, dass die Ergebnisse von ersteren in letzteren aufgegriffen und damit wirksam werden können. Dieser Aspekt wird von Ortmann et al. als bedeutsames Forschungsgebiet angesprochen, aber nicht untersucht (a.a.O., S. 410). Diesen Vorschlag greife ich mit der vorliegenden Arbeit auf und integriere die Erkenntnisse von Ortmann et al. in den Forschungsrahmen, der in Kapitel 3 vorgestellt wird.

2.2.2.7 Verbindungen zwischen Entwicklung und Nutzung: Leitvorstellungen und Modi des Umgangs mit Software

Die Frage, welcher Zusammenhang zwischen den Modellen sozialer Phänomene, die bei der Softwareentwicklung entstehen, und der Art und Weise, wie Nutzer*innen mit Software umgehen, besteht, steht nur selten im Mittelpunkt soziologischer Forschung zu Software. Rammert et al. (1998) bezeichnen die verbindenden Elemente zwischen Entwicklung und Nutzung als Leitvorstellungen.[55] Diese entstehen aus den Erwartungen, die Entwickler*innen über die zukünftige Nutzung der Technik haben. Leitvorstellungen beinhalten Nutzer*innengruppen, Praktiken und Bedingungen der Techniknutzung und auch die Zwecke, die Nutzer*innen mit der Technik verfolgen. Nicht nur bei der Entwicklung, sondern auch in späteren Phasen des Lebenszyklus orientieren sich Akteur*innen an diesen Erwartungen. Leitvorstellungen beeinflussen dadurch, wie Software gestaltet und genutzt wird. Leitvorstellungen beeinflussen einerseits, welche Handlungsmöglichkeiten Software den Nutzer*innen bietet, also zum Beispiel welche Funktionen sie hat, wie die Benutzer*innenoberfläche aussieht und welche Daten eingegeben werden sollen. Sie prägen andererseits auch,

[55] Damit orientieren sie sich an Dierkes et al.s Begriff der Leitbilder (vgl. Kapitel 1).

wie diese Angebote angenommen werden, also zum Beispiel wie und wozu die Software genau eingesetzt wird und ob Nutzer*innen viele Auswahlmöglichkeiten als unnötig kompliziert verurteilen oder als Ausweis für erfreuliche Flexibilität schätzen. Leitvorstellungen, zum Beispiel über die zukünftige Anwendung und den dafür mindestens notwendigen Funktionsumfang, sind jedoch nicht einfach in der Software abgebildet. Sie hängen auch vom sozialen Kontext ab, in dem der Umgang mit der Software stattfindet. Daher transportiert Software nur einen Teil der Leitvorstellungen aus der Entwicklung in die Nutzung:

> *„Die Aneignung der Leitvorstellungen durch die Nutzer ist ein Übersetzungsprozeß für die im Artefakt verkörperten Sinngehalte. […] Der funktionierende technische Kern wird […] von den Entwicklern an die Nutzer weitergereicht. Ein Teil der semantischen Rahmung verschwindet dabei […]. Ein anderer Teil wird von den Nutzern nur bruchstückhaft angeeignet […]. Keineswegs sind die […] Leitvorstellungen also völlig uninteressant. Sie werden aber mehrmals gebrochen und zum Teil durch andere Vorstellungen ersetzt, wenn das System auf den Tisch der Nutzer kommt.“* (Rammert et al. 1998)

Rammert et al. interessieren sich vor allem für die Leitvorstellungen selbst und rekonstruieren weder, wie die „semantische Rahmung“ im Entwicklungsprozess entsteht, noch, welche sozialen Prozesse Anteil daran haben, dass sie bei der Nutzung „zum Teil durch andere Vorstellungen ersetzt“ werden. Das Konzept der Leitvorstellungen wird hier dennoch aufgeführt, weil es viele der Fragen miteinander verbindet, die in Abschnitt 2.1.2 vorgestellt wurden. Hinter dem Konzept steckt der zentrale Gedanke, der auch die vorliegende Arbeit trägt: Vorstellungen vom Umgang mit Software, die von den Rahmenbedingungen der Entwicklung beeinflusst sind, prägen die Nutzung, aber die Art und Weise, wie sie das tun, hängt von den Rahmenbedingungen ab, unter denen Nutzer*innen sich die Software aneignen.

Nicht nur angedeutet, sondern auch theoretisch fundiert und ausgearbeitet wird der gleiche Gedanke in dem Konzept der Dualität von Technik (Orlikowski 1992). Wanda Orlikowski versucht mit diesem Konzept, den Zusammenhang zwischen Softwareentwicklung und -nutzung aus einer soziologischen Perspektive zu beschreiben. Angelehnt an das strukturationstheoretische Grundmodell der Dualität von Struktur und Handeln, in dem soziale Strukturen als Medium und Resultat menschlicher Auseinandersetzung mit der Welt konzeptionalisiert werden (Giddens 2008 [1984]), lässt sich die Rolle von Software im Sozialen nur über den Umgang der Akteur*innen mit ihr verstehen:

> *„On its own, technology is of no import: it plays no meaningful role in human affairs. It is only through the appropriation of technology by humans (whether for productive or symbolic ends) that it plays a significant role and hence exerts influence. It is only through human action that technology qua technology can be understood.“* (Orlikowski 1992, S. 409)

Entwicklung und Nutzung von Software sind Orlikowski zu Folge gleichermaßen Formen des Umgangs mit Technik. Dieser Umgang ist immer eingebettet in einen sozialen Kontext, der Regeln zur Interpretation und Bewertung des Handelns und Ressourcen beinhaltet, die den Umgang mit Software erst möglich machen. Diese Regeln und Ressourcen bestimmen zwar nicht vollständig, wie Akteur*innen mit Software umgehen können, machen aber bestimmte Umgangsformen mehr und andere weniger wahrscheinlich. Software wird dabei als (auch) materielles Artefakt betrachtet, das aus bestimmten (gestaltenden) Umgangsformen entsteht und gleichzeitig Voraussetzung für alle Umgangsformen ist. Über die Software wirken Akteur*innen und soziale Kontexte, in denen frühere Umgangsformen entwickelt werden, auf Akteur*innen ein, die später eigene Umgangsformen entwickeln und damit ihre eigenen sozialen Kontexte verändern. Orlikowski unterscheidet dabei nicht prinzipiell zwischen Softwareentwicklung und –nutzung, sondern definiert stattdessen zwei Modi des Umgangs mit Software, die in allen Phasen des Lebenszyklus vorkommen können, und diskutiert die Auswirkungen dieses Umgangs auf den sozialen Kontext, in dem er stattfindet.

(1) Im *design mode* betten Akteur*innen die Regeln und Ressourcen, die den Umgang mit der Technik ermöglichen und beschränken, in die Technik ein.
(2) Im *use mode* wirken diese eingebetteten Regeln und Ressourcen dann gemeinsam mit denen, die aus dem sozialen Kontext stammen, auf die Akteur*innen ein, die mit der Software umgehen (a.a.O., S. 410).
(3) Der *soziale Kontext* wiederum verändert sich in beiden Modi durch die Umgangsformen, die Akteur*innen in Auseinandersetzung mit der Software entwickeln.

Die Idee, dass Regeln und Ressourcen in Technik eingebaut werden und so zu Strukturen des sozialen Kontexts ihrer Nutzung werden, widerspricht der zentralen Grundfigur der Dualität von Struktur und Handeln in der Strukturationstheorie. Dieser Grundfigur zu Folge können die strukturellen Eigenschaften eines Sozialsystems, zu denen Regeln und Ressourcen zählen, nicht dauerhaft fixiert werden. Sie existieren nur im Handeln, wenn Akteur*innen in den Praktiken des Sozialsystems auf sie Bezug nehmen (Giddens 2008 [1984]). Die Figur der

Dualität von Struktur und Handeln in der Strukturationstheorie bildet den Ausgangspunkt für das Modell der Dualität von Technik. Der Widerspruch zwischen dem Konzept und seiner sozialtheoretischen Grundlage wird in einer späteren Arbeit der Autorin aufgelöst (Orlikowski 2000). In dieser Arbeit werden die Regeln und Ressourcen in eigenen Strukturen der Techniknutzung (*technologies-in-practice*) statt in der Software verortet. Die Lösung des theoretischen Problems wird dabei jedoch mit einer Abwendung von zentralen Ideen des ursprünglichen Modells der Dualität von Technik erkauft: Im ursprünglichen Modell sind die Software, das Handeln der Akteur*innen und die verschiedenen sozialen Kontexte, in denen Software entwickelt, eingeführt oder genutzt wird, für die Analyse des Umgangs mit Software gleich bedeutsam. In dem Modell, in dem die Widersprüche zu sozialtheoretischen Grundannahmen aufgelöst sind, werden die Praktiken der Techniknutzung, also das Handeln der Akteur*innen, als das zentrale Element hervorgehoben, durch das sich der Umgang mit Software verstehen lässt. Durch die Konzentration auf Praktiken der Techniknutzung tritt die Bedeutung der Software und die der sozialen Kontexte, in denen Software entwickelt, eingeführt oder genutzt wird, in den Hintergrund. Für die soziologische Perspektive auf Software, die ich in dieser Arbeit entwickeln will, greife ich die Grundidee der Trias dreier gleich bedeutsamer Einflussfaktoren wieder auf. Wie sich der Umgang mit Software soziologisch als ein Zusammenspiel aus Software, sozialen Kontexten und Handeln von Akteur*innen verstehen lässt, wird in Abschnitt 2.3 theoretisch ausgeführt. Auch für den Forschungsrahmen, der in Kapitel 3 ausgearbeitet wird, bildet das Konzept der Dualität von Technik von Wanda Orlikowski eine wichtige Grundlage.

In den hier vorgestellten Arbeiten werden verschiedene der in 2.1.2 aufgeworfenen Fragen zum Softwarelebenszyklus untersucht. Die Autor*innen berücksichtigen dabei jedoch jeweils nur wenige der Einflüsse, die beim Umgang mit Software zwischen professioneller und organisationaler Einbettung der Akteur*innen, ihren Weltbildern, Machtressourcen und Interessen und den Handlungsmöglichkeiten der Akteur*innen bestehen. Vereinfacht gesagt erfasst jede der Arbeiten nur einen Ausschnitt der Komplexität der Software, die im Lebenszyklus entsteht. Die Rekursivität von Software wird in keinem der Ansätze explizit berücksichtigt. Sie wird in der Soziologie allgemein selten thematisiert. Eine Ausnahme stellen die wenigen Arbeiten dar, in denen Soziolog*innen sich gezielt mit der Weiterentwicklung und Wiederverwendung von Software auseinandersetzen. Solche Arbeiten werden im Folgenden vorgestellt.

2.2.3 Arbeiten zu Weiterentwicklung und Wiederverwendung

Bei der soziologischen Auseinandersetzung mit Software gehen Forschende in der Regel (explizit oder implizit) davon aus, dass Technik und soziale Prozesse einander in einem Verhältnis gegenüberstehen, das durch das Begriffspaar von Inskription und Deskription (Akrich 2000) beschrieben werden kann. Dieses Verhältnis lässt sich kurz wie folgt beschreiben: Aspekte sozialer Phänomene werden bei der Entwicklung in (wie auch immer spezifizierte) Eigenschaften der Technik überführt (Inskription). Bei der Nutzung werden aus diesen Eigenschaften Hinweise über das aktuelle soziale Phänomen extrahiert (Deskription), an denen sich die Akteur*innen in ihrem Handeln (in irgendeiner Form) orientieren. Nach der Inskription ist die Technik fertig, Veränderungen während der Nutzung sind nur interessant, wenn das Artefakt stabil bleibt, aber die Umgangsformen sich wandeln (Orlikowski 2000). Der bisher dargestellte Forschungsstand zur Komplexität von Software folgt diesem Modell: Die Komplexität von Software wird als Folge der Übersetzungen zwischen sozialen Phänomenen und codierten Arbeitsanweisungen für die Hardware behandelt (1. Dimension). Im Rahmen der Modellierungsprozesse, in denen diese Übersetzungen stattfinden, versuchen die beteiligten Akteur*innen, Unsicherheiten des Sozialen zu kontrollieren, indem sie die Eigenschaften der Modelle aushandeln. Im Modell ist nur sichtbar, *wofür* die Akteur*innen sich in den Aushandlungen entschieden haben, also zum Beispiel, dass das Modell einer Patient*in in einer Krankenhaussoftware die Eigenschaften Geschlecht, Alter und Symptome haben soll. Sichtbar wird weder, *wogegen* entschieden wurde, noch wer dies mit welchen Argumenten durchgesetzt hat. Am Beispiel: Das Modell der Patient*in zeigt nicht, dass es den Vorschlag gab, die Ernährungsgewohnheiten in der Software zu berücksichtigen, dass dieser aber nach intensiver Debatte durch ein Veto der Projektleiter*in abgelehnt wurde. Es zeigt auch nicht, dass diese Ablehnung damit begründet wurde, dass die zusätzliche Eigenschaft im Patient*innenmodell die Benutzer*innenoberfläche unnötig unübersichtlich machen würde. Indem die Akteur*innen solche Modell einfach nutzen, akzeptieren sie auch die Entscheidungen, die hinter den Modellen liegen, und geben damit Kontrolle über die Unsicherheiten ab, die in dem Modell reduziert sind. Sie verschieben damit implizit die Kontrolle, die sie auch selbst ausüben könnten, in frühere Prozesse. Sobald die Modelle jedoch in Frage gestellt werden, erscheinen die Unsicherheiten erneut. Am Beispiel: Entscheiden die Entwickler*innen des Diagnosemoduls der Krankenhaussoftware, dass Daten zu den Ernährungsgewohnheiten der Patient*innen unabdingbar sind, wenn das Modul seinen Zweck erfüllen soll, lehnen sie die Modellentscheidungen ihrer Vorgänger*innen ab und bringen die Unsicherheit wieder hervor, die in der

Modellierung von Patient*innen steckt. Wenn es ihnen gelingt, im Projekt die gewünschte Änderung des Patient*innenmodells durchzusetzen, so verschieben sie die Kontrolle aus dem Prozess der Patient*innenmodellierung in den Prozess der Entwicklung des Diagnosemoduls. Bei der Softwareentwicklung werden so aufeinander aufbauende Modelle gestaltet, übernommen, geprüft und verändert. Insgesamt wird damit Kontrolle über die Unsicherheiten des Sozialen zwischen den einzelnen sozialen Prozessen hin- und hergeschoben (2. Dimension). Da Software nicht beliebig genutzt werden kann, haben die vielen Entscheidungen während der Softwareentwicklung Auswirkungen auf die Nutzung. Solange Nutzer*innen die Modelle, auf denen die Software beruht, nicht in Frage stellen, wird über die Modellentscheidungen somit auch Kontrolle über Unsicherheiten ausgeübt, die bei der Nutzung auftreten. Entwicklungsprozesse schaffen so die Grundlage für eine Performativität der Software, die sich in Nutzungsprozessen entfaltet.

Wie in Abschnitt 2.1.3 dargestellt sind jedoch auch die Entwicklungsprozesse von Software beeinflusst. Software wird nicht nur langfristig weiterentwickelt, sondern kann auch als Komponente in andere Software eingebaut werden. Die komplexen Zusammenhänge zwischen sozialem Phänomen und Code, die in dieser Arbeit als die erste und zweite Dimension der Komplexität von Software bezeichnet werden, gewinnt dabei eine neue Qualität. Über den Code bestehender Versionen oder wiederverwendeter Softwareelemente werden Rück- und Querverbindungen zwischen Entwicklungs- und Nutzungsprozessen in ganz unterschiedlichen Kontexten vermittelt. Um solche Rück- und Querverbindungen und die dadurch gesteigerte Komplexität von Software geht es in den Arbeiten, die ich im Folgenden zum Abschluss meines Literaturüberblicks vorstelle.

2.2.3.1 Strategische Komplexitätssteigerung bei Online-Plattformen

In der Soziologie erregen Online-Plattformen seit einigen Jahren besondere Aufmerksamkeit. Online-Plattformen sind ein idealtypisches Anwendungsgebiet für agile Softwareentwicklung, bei der die Anbieter*innen permanente Weiterentwicklung nutzen, um Informationen über das soziale Phänomen zu generieren, das durch Software gesteuert werden soll. Anbieter*innen von Online-Plattformen führen zum Beispiel sogenannte A/B-Tests durch. Bei diesen bieten sie ihren Besucher*innen zufällig eine von zwei Websiteversionen mit unterschiedlichen Layoutversionen an, prüfen, welche der Besucher*innengruppen mehr Umsatz generiert, und verändern dann die Plattform entsprechend der Testergebnisse. In der Wirtschaftssoziologie werden die Anbieter*innen von Online-Plattformen als Marktorganisator*innen betrachtet. Sie setzen statt formaler Organisationsregeln

Software ein, um über die klassischen Organisationsgrenzen hinweg Verhaltenskontrolle auszuüben und Nutzer*innen so in Quasi-Mitglieder der Organisation zu verwandeln (Kirchner und Schüßler 2019). In einer Studie über die Entwicklung einer Fitness-Online-Plattform untersucht Tilo Grenz (2014) Wechselwirkungen, die bei der Softwareevolution zwischen Entwicklungsprozess und Nutzungsprozess entstehen. Dabei zeigt er, dass Anbieter*innen von Online-Plattformen strategisch Kontrolle über einige Aspekte des sozialen Phänomens aus den internen Modellierungsprozessen in die Nutzungsprozesse auslagern, um mit Hilfe der Software andere Aspekte, die für die Ziele des Unternehmens bedeutsamer sind, kontrollieren zu können.

Für die Arbeit untersucht Grenz die Entwicklung einer Fitness-Online-Plattform, die im Auftrag eines Unternehmens entwickelt wird, das eine Kette von Fitnessstudios betreibt. Die Plattform soll Kund*innen an die Studios binden, indem sie sie beim individualisierten Training unterstützt. Zu diesem Zweck eröffnet die Plattform den Kund*innen die Möglichkeit, aus verschiedenen aktuellen Fitnesstrends einen auszuwählen und ihren persönlichen Trainingsplan entsprechend des ausgewählten Trends zu organisieren (Grenz 2014). Statt wie im üblichen Geschäftsmodell von Fitnessstudios Informationen über Trends, Vorlieben und Trainingsfortschritte durch eigenes Personal im Rahmen von formal geregelten Organisationsprozessen zu erheben und an die Kund*innen weiterzugeben (a.a.O., S. 24), lässt die Organisation eine Online-Plattform entwickeln. Auf dieser können Trainingspläne, die von internen und externen Expert*innen und von Nutzer*innen selbst erstellt worden sind, eingesehen, diskutiert, geordnet und abgerufen werden (a.a.O., S. 25). Der Umgang mit der Plattform wird anhand von direktem Feedback von Nutzer*innen und mit Hilfe von Datenspuren analysiert. Für diese Analyse müssen die Mitglieder des Teams, das die Online-Plattform weiterentwickelt, die Datenspuren interpretieren. Ziel dieser Interpretationen ist es, Anhaltspunkte für neu zu entwickelnde Funktionen zu finden, die geeignet scheinen, die zukünftige Nutzung im Sinne der betrieblichen Interessen zu steuern (a.a.O., S. 39). Die Interpretation der Daten aus dem Nutzungsprozess und die Ableitung von Funktionen sind dabei ebenso Gegenstand mikropolitischer Auseinandersetzungen innerhalb der Organisation wie die Frage, welche Nutzer*innengruppen das Unternehmen mit der Online-Plattform eigentlich genau ansprechen will (a.a.O., S. 31). Durch die dauerhafte Einbindung der Nutzungsprozesse steigt mit der Zeit die Komplexität der Softwareevolution, auch, weil dadurch immer weitere externe Technologien (z. B. Tracking-Apps für Smartphones) an die Online-Plattform angebunden werden (a.a.O., S. 32), deren Funktionen immer neue Umgangsformen erzeugen. Die Möglichkeit, permanent Daten über den Nutzungsprozess in die Weiterentwicklung einzuspeisen, wird

im Unternehmen *„als permanenter* Anpassungsdruck *wahrgenommen“* (a.a.O., S. 42): Die Daten scheinen den Weg zu immer weiteren Verbesserungen der Plattform aufzuzeigen. Solche vermeintlichen Verbesserungen nicht umzusetzen und die Online-Plattform dauerhaft in einem festen Zustand zu belassen käme einer Verweigerung von Innovation gleich. Der Innovationsimperativ (oder zumindest der hohe Stellenwert von Innovationen) in unserer Gesellschaft (Hutter et al. 2011) ist für das Fitnessunternehmen schwer zu ignorieren.

Insgesamt zeigen sich Online-Plattformen in der Untersuchung von Grenz damit nicht als funktionales Äquivalent zu formalen Organisationsregeln, sondern als zentrales Element eines durch digitale Technik geprägten Fitnessangebots, das durch die Wechselwirkungen zwischen Entwicklungs- und Nutzungsprozessen permanent an Komplexität gewinnt. Durch die Weiterentwicklung wird diese Komplexitätssteigerung vorangetrieben. Sie bietet aber aus Sicht des Unternehmens, das die Online-Plattform anbietet, die besten Chancen, Kontrolle über soziale Phänomene wie die Trainingsvorlieben der Kund*innen auszuüben, die größtenteils unbekannt sind und sich (auch durch die Weiterentwicklung selbst) ständig verändern.

2.2.3.2 Kontrollierte Komplexitätssteigerung bei standardisierter Organisationssoftware

Im Fall der Online-Plattformen ist relativ offensichtlich, dass die Komplexität der Software durch die wiederholte Einbindung der Nutzungsprozesse in die Entwicklungsprozesse permanent steigt. Der gleiche Effekt kann auch bei traditionelleren Formen von Software beobachtet werden. Dass auch andere Software permanent weiterentwickelt wird und dadurch ständig an Komplexität gewinnt, wird der soziologischen Forschung selten beachtet. Eine Ausnahme stellen Studien dar, die an das Forschungsprogramm zur Biographie von Artefakten anschließen. Im Zentrum dieses Programms steht die Frage, wie Forschende die Einflüsse untersuchen können, die im Laufe der langfristigen Entwicklung und Weiterentwicklung auf eine Technik (wie Software) und den Umgang mit ihr einwirken. Diese Einflüsse stammen aus den unterschiedlichen sozialen Kontexten, in denen das Artefakt, dessen Biographie untersucht wird, im Laufe der Zeit entsteht und verändert wird. Grundlage des Ansatzes sind eine Reihe von Studien zur Entwicklung und Weiterentwicklung standardisierter Organisationssoftware, die vor allem von Neil Pollock und Robin Williams durchgeführt worden sind (Pollock und Williams 2009; Williams und Pollock 2012; Pollock et al. 2007; Pollock und Cornford 2004). Die Studien zeigen, wie sich ein Softwareprodukt über Jahrzehnte hinweg entwickelt und weltweit und über verschiedenste Branchen hinweg verbreitet. Einerseits orientiert sich das Unternehmen, das die

Software herstellt, bei der Entwicklung (Meissner 1997) und Weiterentwicklung (Pollock et al. 2007) an den Anforderungen von den Repräsentant*innen konkreter Kund*innenorganisationen. Andererseits werden diese Anforderungen von Anfang an mit Blick auf die strategischen Interessen der Organisation ausgewählt, die für die Entwicklung der Software verantwortlich zeichnet, und so interpretiert, wie es diesen Interessen entspricht: Die Software soll an möglichst viele unterschiedliche Organisationen vermarktet werden; dafür müssen die Funktionen sowohl hochgradig generisch sein als auch sich an die im Detail hochgradig unterschiedlichen sozialen Phänomene anpassen lassen. Eine Funktion zum Erstellen von Rechnungen soll zum Beispiel in möglichst vielen Unternehmen genutzt werden können und die Rechnungen sollen gleichzeitig in jedem Unternehmen in einem Format erzeugt werden, das den spezifischen Vorgaben des Unternehmens möglichst genau entspricht. Dieser scheinbare Widerspruch wird durch die technische Struktur teilweise gelöst. In dieser Struktur werden bestimmte Eigenschaften der Software als generischer Kern festgehalten und durch Prozessvorlagen, sogenannte Templates, ergänzt, aus welchen Kund*innen lokal die Varianten auswählen können, die ihren spezifischen Anforderungen am ehesten entsprechen (Pollock et al. 2007).

Der generische Kern wird von Softwareunternehmen strategisch geschützt (Pollock und Williams 2009), zum Beispiel indem ihre Mitarbeiter*innen bei der Weiterentwicklung bereits möglichst früh im Modellierungsprozess darauf achten, dass die Modelle für neue Funktionen kompatibel mit den Modellen der bestehenden Software sind. Besondere Anstrengungen in dieser Hinsicht werden bei der Neuentwicklung von Branchenvarianten unternommen: Hier sind die Hersteller*innen darauf angewiesen, entweder Repräsentant*innen von Pilotorganisationen in die internen Entwicklungsprojekte einzubeziehen (Pollock et al. 2007) oder Modelle, die im Rahmen des Customizing in einzelnen Organisationen entstanden sind, in den generischen Kern zu übernehmen, um Prozessvorlagen in der Software als „best practices“ bewerben zu können (Locke und Lowe 2007; Wagner et al. 2006). Die Komplexität der Software steigert sich mit der Zeit, weil die Modelle, auf denen die hinzugefügten Elemente aufgebaut sind, von den konkreten sozialen und technischen Bedingungen in den Pilotorganisationen abhängen. Diese Bedingungen beeinflussen sowohl die Auswahl der zugelassenen Repräsentant*innen als auch deren Beiträge zur Modellierung des sozialen Phänomens.[56]

[56] Wagner et al. zeigen, dass solche Prozesse keineswegs nur vernachlässigbare Details bei der Interpretation des sozialen Phänomens beeinflussen, sondern grundsätzlich Auswirkung darauf haben, was überhaupt modelliert wird. In der Studie von Wagner et al. versuchen die Verwaltungsspitzen einer großen Universität, die Fakultäten zur Umsetzung eines von ihnen

Mitarbeiter*innen des Softwareunternehmens versuchen die Kontrolle über diese Komplexität in der eigenen Organisation zu halten. Dazu entscheiden sie im Modellierungsprozess mit Blick auf die von ihrem Unternehmen vorgegebenen Projektziele, wessen Anforderungen an neue Funktionen wie berücksichtigt werden oder welche in anderen Organisationen entstandenen Erweiterungen in den generischen Kern aufgenommen werden (Pollock et al. 2007). Gelingt es den Mitarbeiter*innen, hinter der Anforderung einer Repräsentant*in, die eine eigene Funktion für die an ihrer Universität übliche Form der Benotung einfordert, ein allgemeineres Problem zu entdecken, das von den Repräsentant*innen anderer Organisationen als solches bestätigt wird, wird die Anforderung aufgenommen. Scheint das Problem auf eine Organisation beschränkt, wird die Anforderung abgelehnt (a.a.O. 264–266).

Auf das, was bei der regulären Weiterentwicklung geschieht, die außerhalb der initialen Modellierung neuer Branchenvarianten stattfindet, passt die Idee einer primär am Nutzungsprozess orientierten Modellierung dann überhaupt nicht mehr: In dieser Phase lehnt das Softwareunternehmens die meisten Anfragen von einzelnen Kund*innenorganisationen ab und begründen die Ablehnung damit, dass die Anforderungen zu spezifisch seien. Während des Projekts, bei dem das Branchenmodul entwickelt wurde, bemühte sich eine Mitarbeiter*in des Softwareunternehmens darum, das generische Problem hinter der spezifischen Anforderung zu identifizieren. Bei der Weiterentwicklung müssen die Kund*innenorganisationen sich nun allein um diese Aufgabe kümmern, wenn sie wollen, dass ihre Anforderungen in der Software berücksichtigt werden. Das Softwareunternehmen organisiert Communities, in denen Repräsentant*innen der Kund*innenorganisationen sich austauschen und andere finden können, die ihren Wunsch nach einer bestimmten Anforderung unterstützen (a.a.O., S. 267). In den Communities identifizieren die Repräsentant*innen gemeinsam das allgemeine Problem hinter ihren spezifischen Anforderungen und übernehmen damit die Aufgabe, die vorher eine Mitarbeiter*in des Softwareunternehmens erfüllt hat: Sie formulieren ihre Wünsche so aus, dass sichtbar wird, dass deren Erfüllung den Zielen des Softwareunternehmens dient.

---

abgelehnten Verfahrens zur Abrechnung von Forschungsprojekten zu zwingen. Dazu bringen sie das gewünschte Verfahren als „best practice“ in die Standardsoftware ein, deren Universitätsmodul sie in Kooperation mit dem Hersteller entwickeln. Obwohl die Verwaltungsleitung den internen Konflikt verliert und die Standardsoftware in der Universität nur mit umfangreichen Erweiterungen genutzt wird, die es den Mitarbeiter*innen erlauben, weiterhin nach dem etablierten Verfahren abzurechnen, verbreitet sich das neue Verfahren mit der Software und dem Namen der Pilotkundin in andere Universitäten (Wagner et al. 2006).

Bei der Betrachtung des Konzepts der Biographie von Artefakten zeigt sich, dass über die Weiterentwicklung mit der Zeit immer neue Prozesse zu dem Geflecht aus Prozessen hinzukommen, in welchem ausgehandelt wird, wie Software und der Umgang mit ihr genau aussehen sollen. In jeder dieser Aushandlungen versuchen die Akteur*innen, Unsicherheiten des Sozialen durch Entscheidungen zu reduzieren. In der eben vorgestellten Studie geschieht dies zum Beispiel, indem sie die spezifischen Prozesse zur Benotung von Studierenden, die an jeder Universität anders sind, auf zwei verschiedene generische Prozessvorlagen reduzieren, aus denen spätere Nutzer*innen dann eine auswählen müssen, wenn sie die Software nutzen wollen, um Noten zu speichern. Bei jedem dieser Reduktionsversuche entstehen jedoch auch immer neue Unsicherheiten, zum Beispiel durch die Auswahl der an der Aushandlung Beteiligten, die zukünftige Nutzer*innen repräsentieren sollen, ohne deren Anforderungen genau kennen zu können. Anders als bei Online-Plattformen versuchen Softwareunternehmen in diesen Fällen jedoch die Komplexität der Software durch gezielte Kontrolle auch des Codes zu steuern (Pollock et al. 2007). Robert Schmidt zeigt in seiner Arbeit zu den Ursachen der Software-Alterung (Schmidt 2012), dass die Kontrolle auch dadurch begrenzt ist, dass die Biographie der Software im Code präsent bleibt, wodurch die Software bei der Weiterentwicklung performativ wird:

> *„Because [the code, DA] acts not only as a passive, but also as a (re)active operative entity, the continuing modification of the software continually creates situations in which pieces of the code perform erstwhile tasks and organizational conditions in a modified context." (a.a.O., S. 206).*

Die Praxis der Programmierung bringt es mit sich, dass der Zusammenhang zwischen den bei der Modellierung intendierten und den vom Code tatsächlich verursachten Funktionen mit der Zeit immer undurchschaubarer wird.

2.2.3.3 Auslagerung von Komplexität aus dem sozialen Phänomen in Standardisierungsprozesse

Die Kontrolle, die Hersteller*innen von Standardsoftware über die Modellierungsprozesse ausüben, kann als Versuch interpretiert werden, die Komplexität von Nutzungsprozessen im Voraus so zu begrenzen, dass die Funktionen ihrer Produkte auf Dauer als hilfreiche Unterstützung der Nutzung wahrgenommen werden. Standards der Datenverarbeitung setzen mit der Komplexitätsreduktion früher an. Sie sind das Ergebnis von Versuchen, für ein ganzes Bündel an sozialen Phänomenen festzuschreiben, welche Aspekte der Unsicherheit in den Modellierungsprozessen der Softwareentwicklung berücksichtigt werden sollen. Standards sind, einmal etabliert, schwer zu verändern und berühren per definitionem eine

große Zahl an sozialen Phänomenen. Im Detail gibt es oft sehr unterschiedliche Ansichten darüber, wie unterschiedliche Aspekte dieser sozialen Phänomene zu interpretieren sind, welche der Aspekte für den Standard zentral und welche nebensächlich sind (Bowker und Star 2000). Zentrale Aspekte müssen bei der Standardisierung einheitlich definiert, nebensächliche können ignoriert werden. Anders gesagt: Wie bei der Softwareentwicklung sind die Akteur*innen auch bei der Standardisierung mit einem hohen Grad an Unsicherheit konfrontiert, der durch Standards reduziert werden soll. In den Aushandlungen, die in Standardisierungsprozessen stattfinden, wird darüber entschieden, wie die standardisierten Phänomene in den Prozessen interpretiert werden sollen, in denen die Standards zum Einsatz kommen. Die Aushandlungen stellen also den Versuch dar, Unsicherheiten der Anwendungsprozesse im Voraus zu kontrollieren. Wegen ihrer weitreichenden Auswirkungen – Standards reduzieren sehr viele Unsicherheiten auf einmal und werden gleichzeitig in sehr vielen Prozessen genutzt – sind Standardisierungsprozesse immer hochgradig politisch: *„Whatever appears as universal or indeed standard, is the result of negotiations, organizational processes, and conflict"* (Bowker und Star 2000). Ole Hanseth und Eric Monteiro (1997) zeigen dies am Beispiel der Entstehung eines Standards zum elektronischen Austausch von Labordaten in Norwegen. In ihrer Fallstudie machen sie sichtbar, wie viele unterschiedliche Akteur*innengruppen versuchen, Einfluss auf einen Standardisierungsprozess zu nehmen, bei dem die Unsicherheiten eines sozialen Phänomens so reduziert werden sollen, dass sich primär auf Software gestützte Prozesse durchsetzen können. Während der Anstoß für den Standardisierungsprozess noch von den Hersteller*innen von Praxis- und Laborsoftware ausgeht, die in engem Kontakt mit den Nutzer*innen der Nachrichtenformate stehen (a.a.O., S. 192), versuchen große Unternehmen von Beginn an den Standard so zu beeinflussen, dass er möglichst weitgehend mit den Modellen übereinstimmt, die ihren eigenen Softwareprodukten zu Grunde liegen. Diese strategischen Interessen werden im Standardisierungsprozess nicht expliziert, sondern als Ausdruck „objektiver" Interpretationen des sozialen Phänomens dargestellt:

> *„As there was a general consensus […] about the need for standards, the fight about what these standards should look like and how they should be developed started. This race was a seemingly neutral and technical discussion about which technology fitted the needs best. In reality, however, it was a race between different actors trying to manoeuvre themselves into key positions [...]" (Hanseth und Monteiro 1997, S. 192).*

Dabei wird schnell überlegt, an welche bereits existierenden internationalen Standards für medizinischen Datenaustausch der neue angeschlossen werden soll. Diese werden in Arbeitsgruppen internationaler Standardisierungsorganisationen

wie der ISO entwickelt, in denen Repräsentant*innen von Unternehmen und Regierungen neue Standards ausarbeiten. Die einflussreichen Akteur*innen in diesen Arbeitsgruppen sind vor allem an der Erweiterung bestehender Standards interessiert, da ein neuer technischer Standard nicht nur aus Modellen für die Verarbeitung von Information besteht, sondern von Software umgesetzt und eingebunden werden muss, um wirksam zu werden (a.a.O., S. 196). Die Existenz dieser technischen Infrastruktur erzeugt nicht nur strategische Interessen bei den Hersteller*innen, den neuen Standard möglichst im Einklang mit ihren bestehenden Produkten zu formen, sondern auch hohe Eintrittsbarrieren für Akteur*innen, die sich nur mit dem ursprünglichen Handlungsproblem, nicht aber mit Standardisierung auskennen. *„To be involved in the standardization work, one needs to know all the rules of the game […] in order to make lab standards, one also has to be familiar with standards in virtually all other sectors as well."* (a.a.O., S. 196).

Über den Standard zum Austausch von Labordaten entscheiden am Ende Standardisierungsbürokrat*innen, die sich für die praktischen Probleme, die mit Hilfe des elektronischen Datenaustauschs gelöst werden sollen, weniger interessieren als für die Konsistenz des neuen Datenformats mit bestehenden Standards. Der Standard, der in der Fallstudie letztendlich ausgehandelt wird, ist kompliziert, löst viele der Probleme nicht, die Nutzer*innen in der Praxis haben und ist schwer in die vorhandenen Arbeitsabläufe einzubinden (a.a.O., S. 197). Dennoch können sich weder Nutzer*innen noch Softwarehersteller*innen dem Standard verweigern, ohne selbst eine Alternative zu konstruieren, die ihrerseits von einer großen Gruppe einflussreicher Akteur*innen unterstützt würde (a.a.O., S. 207).

Standards liegen nicht nur dem Code zu Grunde, sondern auch Programmiersprachen und –paradigmen, Vorgehensmodellen und den Programmen, die im Rahmen des Entwicklungsprozesses genutzt werden (Schulz-Schaeffer und Bottel 2018). Die hohe Standardisierung ist Voraussetzung und Folge der hohen internationalen Kooperation, die für die heutige Softwareentwicklung kennzeichnend ist (a.a.O.). Da Standards der Datenverarbeitung die Komplexität der Softwareentwicklung verringern sollen, werden die komplexen Prozesse, die zu ihrer Entstehung führen, bei ihrem Einsatz gezielt ausgeblendet.

2.2.3.4 Softwaresysteme als Infrastrukturen

Die Wiederverwendung von Softwarekomponenten ist die in der Softwareentwicklung bedeutendste Form, die Komplexität im Entwicklungsprozess zu reduzieren. Doch in der Soziologie ist Wiederverwendung bisher praktisch nicht untersucht worden. Bei der Wiederverwendung werden Teile der Komplexität des sozialen Phänomens aus dem Softwareentwicklungsprozess ausgelagert, indem

sie, ähnlich wie bei der Standardisierung, in andere Prozesse verschoben werden. Diese Strategie hat Folgen auch für Endnutzer*innen, zum Beispiel weil unsichtbare Abhängigkeiten zwischen den Prozessen der Softwareentwicklung (und damit auch der späteren Nutzung dieser Software) entstehen, in denen wiederverwendete Software eingesetzt wird, und denen, in denen sie entwickelt wird. Solche Folgen werden teilweise in der Forschung zu digitalen Informationsinfrastrukturen betrachtet. Grundlage dieses Forschungsfelds ist die Arbeit von Bowker und Star zur Entstehung von Kategorien und deren Bedeutung als Informationsinfrastrukturen, die Gesellschaft strukturieren (Bowker und Star 2000). Infrastrukturen werden in dieser Forschung als relationales Konzept genutzt, um Artefakte und den Umgang mit ihnen gemeinsam zu untersuchen (Star und Ruhleder 1996). Als Informationsinfrastrukturen (IIs) werden dabei sowohl die technischen Systeme verstanden, die aus (im hier vorgestellten Sinn) komplexer Software bestehen, als auch die Umgangsformen, die sich mit diesen Systemen in verschiedenen Nutzungsprozessen entwickeln. Mit dem Begriff der Informationsinfrastruktur versuchen Forscher*innen, alle Aspekte zu berücksichtigen, die für die Komplexität des Gegenstands relevant sein könnten, ohne die Verhältnisse, in denen diese Aspekte zu einander stehen, über die Definition festzulegen:

> *„IIs are characterised by openness to number and types of users (no fixed notion of "user"), interconnections of numerous modules/systems (i.e. multiplicity of purposes, agendas, strategies), dynamically evolving portfolios of (an ecosystem of) systems and shaped by an installed base of existing systems and practices (thus restricting the scope of design, as traditionally conceived). IIs are also typically stretched across space and time: they are shaped and used across many different locales and endure over long periods (decades rather than years)" (Monteiro et al. 2013, S. 576)*

Eine solche übergreifende Definition ist für die Operationalisierung und den Vergleich verschiedener Forschungsergebnisse wenig hilfreich, was innerhalb des Forschungsfeldes auch kritisiert wird (Lee und Schmidt 2018). Dennoch sind in diesem Forschungsfeld in den letzten Jahren eine Vielzahl an empirischen Studien entstanden, die jeweils mit sehr unterschiedlichen Definitionen des Begriffs der Infrastruktur arbeiten (Pipek et al. 2017; Lee und Schmidt 2018; Monteiro et al. 2014). In all diesen Arbeiten wird aber betont, dass Software, Akteur*innen, welche egal in welcher Rolle mit ihr umgehen, und die sozialen Prozesse, in denen dieser Umgang stattfindet, sich gegenseitig beeinflussen und nur verstanden werden können, wenn die Wechselwirkungen zwischen ihnen berücksichtigt werden (Karasti et al. 2018, 270 f.).

Für die in der vorliegenden Arbeit entwickelte soziologische Perspektive auf Software ist die Forschung zu Informationsinfrastrukturen vor allem deswegen

relevant, weil in ihr thematisiert wird, dass Modellierungsprozesse bei der Entwicklung einer Software auch von den Ergebnissen der Modellierungsprozesse anderer Software beeinflusst werden:

> *„Infrastructures are not designed from scratch – they are normally designed by modifying and extending what already exists. So the infrastructure as it currently is – the installed base – wield a strong influence on what it may become in the future." (Monteiro et al. 2013, S. 597)*

Das komplexe Geflecht an sozialen Prozessen, in dem sich die Kontrolle des Umgangs mit Software verteilt, umfasst mit dieser Perspektive alle Prozesse, die jemals einen Beitrag zur Modellierung irgendeiner ihrer Komponenten geleistet haben.

2.2.4 Fazit: Fragmentierte Soziologie der Software

Durch die Aufarbeitung des Forschungsstands wurde gezeigt, dass viele der in Abschnitt 2.1 aufgeworfenen Fragen zur Soziologie der Software bereits in der soziologischen Forschungsarbeiten thematisiert worden sind. Dabei ist aber auch sichtbar geworden, dass kaum Forschung existiert, in der mehrere Dimensionen der Komplexität von Software gleichzeitig berücksichtigt werden. Die Komplexität von Software entsteht jedoch vor allem durch die Wechselwirkungen zwischen den drei vorgestellten Dimensionen: Um die Aufgabe, soziale Phänomene in durch Computer ausführbaren Code zu übersetzen, zu erfüllen, müssen so viele Unsicherheiten reduziert werden, dass Akteur*innen sich gezwungen sehen, diese Reduktionen auf verschiedene Modellierungsprozesse zu verteilen. In den einzelnen Prozessen und durch die Abhängigkeiten zwischen ihnen entstehen jedoch neue Unsicherheiten. Diese versuchen die Akteur*innen zu kontrollieren, indem sie bestehende Software wiederverwenden. Dadurch entstehen neue Abhängigkeiten, neue Akteur*innen, soziale Prozesse und Unsicherheiten kommen hinzu und so weiter. Die soziologischen Arbeiten, in denen solche Bezüge zwischen den Dimensionen angesprochen werden, eignen sich kaum als Vorlage für weitergehende Forschung zum Thema. Die Arbeiten von Kallinikos (2009) und Orlikowski (1992) bleiben in entscheidenden Punkten abstrakt, was es schwierig macht, mit neuen Arbeiten an ihre Konzepte anzuschließen. Bei Arbeiten wie der von Ortmann et al. (1990) geht es nur nebenbei um Software. Die Komplexität von Software wird zwar erwähnt, aber nicht in die primär organisationssoziologischen Konzepte integriert, die im Fokus der Studien stehen. Die techniksoziologischen Untersuchungen von Rammert et al. (1998) befassen sich

im Gegensatz dazu beinahe ausschließlich mit Software. Sie sind aber so eng auf einen speziellen Gegenstand zugeschnitten, dass sie nicht auf andere Formen von Software verallgemeinert werden können. Insgesamt zeigt sich beim Blick auf den Forschungsstand, dass viele Erkenntnisse, die für eine Soziologie der Software notwendig sind, bereits vorliegen, aber nicht miteinander verbunden sind. Im folgenden Abschnitt schlage ich eine Möglichkeit vor, wie sich diese Elemente verbinden lassen. Dazu identifiziere ich zwei Perspektiven auf Software, welche sich in der soziologischen Forschung zu den verschiedenen Ebenen der Komplexität finden, und setze sie miteinander in Beziehung. Aus der einen Perspektive wird Software als Mittel betrachtet, über das Akteur*innen versuchen, Kontrolle auszuüben. Aus der anderen Perspektive wird Software als Resultat solcher Kontrollversuche betrachtet. Ich zeige, wie diese beiden Perspektiven verbunden werden können, und lege so die Grundlage für die Entwicklung eines Forschungsrahmens, der zur Untersuchung der Performativität von Software im Sozialen eingesetzt werden kann. Dieser Forschungsrahmen wird in Kapitel 3 entwickelt.

2.3 Unsicherheitsreduktion als Mittel und Resultat von Kontrollversuchen

Die Analyse der Komplexität von Software und der Art und Weise, wie Soziolog*innen sich mit dieser Komplexität auseinandersetzen, zeigt, dass Software in soziologischen Arbeiten in der Regel aus einer von zwei Perspektiven untersucht wird: Sie wird entweder als Mittel oder als Resultat von Versuchen betrachtet, Kontrolle über soziale Prozesse auszuüben. Diese Perspektiven stimmen in etwa mit den zwei Aspekten der Technik überein, welche in dem in der Einleitung von 2.2 vorgestellten Konzept der Dualität von Ressourcen und Routinen identifiziert werden. Beiden Perspektiven liegt zumindest implizit die Annahme zu Grunde, dass Entwicklungs- und Nutzungsprozesse grundsätzlich unterschiedlich sind: Es wird untersucht, wie aus Entwicklungsprozessen über Software Einfluss auf Nutzungsprozesse genommen wird, aber nicht, ob und wie das gleiche zwischen Entwicklungs- oder zwischen Nutzungsprozessen geschieht. Die Arbeiten zur Biographie von Artefakten stellen in dieser Hinsicht eine Ausnahme dar und werden in der soziologischen Forschung zu Software bisher selten aufgegriffen. Weder die Rekursivität des Softwarelebenszyklus noch die Performativität, die Software in allen Prozessen aufweist, in denen mit ihr umgegangen wird, werden in der Soziologie bisher hinreichend berücksichtigt.

Aus der ersten Perspektive wird Software als Mittel betrachtet, mit dem Kontrolle über soziale Prozesse ausgeübt werden soll. Dazu besteht sie aus Modellen, in denen die sozialen Phänomene, die für die zu kontrollierenden Prozesse als bedeutsam gelten, so abgebildet werden, dass sie von einem Computer verarbeitet werden können. Während bei sozialen Phänomenen oft nicht ganz klar ist, wo sie anfangen und wo sie enden, haben diese Modelle klare Grenzen; wo die Eigenschaften sozialer Phänomene interpretationsbedürftig sind, sind Modelleigenschaften eindeutig; im Sozialen können spontan ganz neue Handlungsweisen entstehen, aber die Möglichkeiten, Modelle zu transformieren, sind durch die Definitionen in Software festgeschrieben. Am Beispiel: Eine Patient*in mag unsicher sein, ob das Gefühl in ihrem Kopf nur ein Zeichen von Erschöpfung ist oder schon einen Schmerz darstellt, ob der Schmerz nur mehrmals aufgetreten ist oder ob er regelmäßig wiederkehrt und ob sie darauf reagieren möchte, indem sie Medikamente einnimmt, mehr Sport treibt oder in Zukunft auf Laktose (Gluten, Zucker oder rote Lebensmittel) verzichtet. In den Modellen, aus denen Software besteht, gibt es all diese Unsicherheiten nicht: Das Symptom Kopfschmerz ist vorhanden oder es ist nicht vorhanden. Es kann unterschiedliche Ausprägungen haben, wenn diese definiert worden sind. Ist die Häufigkeit des Auftretens als Eigenschaft von Symptomen im Modell vorgesehen, kann diese Information aufgenommen werden. Ist sie nicht vorgesehen, gibt es für die Software keinen Unterschied zwischen einmaligen und regelmäßigen Kopfschmerzen. Auch die möglichen Reaktionen auf dieses Symptom sind im Modell wohldefiniert. Ein Verzicht auf rote Lebensmittel gehört vermutlich nicht dazu. Modelle, die die Unsicherheiten sozialer Phänomene auf eine Reihe an eindeutigen Eigenschaften reduzieren, werden im Rahmen der Softwareentwicklung geschaffen, weil sie die Verarbeitung von Informationen über das Soziale durch Prozessoren ermöglichen sollen. Diese Modelle sind das Ergebnis von Aushandlungen, in denen die relevanten Elemente und Eigenschaften des sozialen Phänomens bestimmt werden und festgelegt wird, wie diese in Software umgesetzt werden sollen. In diesen Aushandlungen wird entschieden, wie Unsicherheiten des Sozialen reduziert werden sollen. Die Zwecke, die durch den Umgang mit der fertigen Software erreicht werden sollen, liefern die Kriterien, nach denen bei der Modellierung entschieden wird. Die Akteur*innen, die an diesen Entscheidungen beteiligt sind, bringen aber auch ihre eigenen Kriterien mit. Am Ende der Aushandlungen steht eine Software, die aus Modellen besteht, die das abbilden, was Nutzer*innen beim Umgang mit Software vorfinden. Übernehmen die Nutzer*innen die Vorgaben aus den Modellen, lösen sie Unsicherheiten, die im Nutzungsprozess bestehen, so auf, wie es bei der Modellierung entschieden wurde: Die Patient*in mit Kopfschmerzen bekommt Schmerzmittel und nicht die Empfehlung, keine roten Lebensmittel

mehr zu essen. Über die Modelle wird kontrolliert, was Nutzer*innen in den sozialen Prozessen tun, in denen sie mit Software umgehen. Genauer gesagt: die Modelle stellen einen Kontroll*versuch* dar, denn die eben beschriebene Kontrolle hat Grenzen. Wenn Software als Mittel zur Kontrolle sozialer Prozesse durch Unsicherheitsreduktion betrachtet wird, so wird diese Kontrolle erstens dadurch begrenzt, dass Software auf Eingaben aus diesen Prozessen angewiesen ist. Diese Eingaben, aus denen die Daten entstehen, die für die Funktionen von Software notwendig sind, werden durch die Modelle zwar strukturiert, sind in ihnen aber niemals vollständig vorgegeben. Was Nutzer*innen eingeben, bleibt also unsicher. Sie wird zweitens dadurch begrenzt, dass die Modellierung sozialer Phänomene für den Computer so aufwändig ist, dass sie nur gelingt, weil jedes komplexe Modell aus einer Vielzahl an einfacheren Modellen besteht und keine Gruppe von Modellierer*innen wissen muss, wie die Modelle im Detail aussehen, die zwischen den von ihnen abgebildeten sozialen Phänomenen und der Maschinensprache liegen, in der diese Abbildungen vom Mikroprozessor bearbeitet werden, und welche Abhängigkeiten genau zwischen ihnen bestehen. Durch diese Komplexität, die bei der Modellierung für den Computer unvermeidlich ist, entzieht sich Software dem vollständigen Verständnis (und damit auch der Kontrolle) aller Beteiligten (Kallinikos 2009). Auch die Basis, auf der die Modellierung ruht, bleibt also unsicher.

Aus der zweiten Perspektive wird Software als Resultat von Kontrollversuchen betrachtet, die von sozialen Prozessen ausgehen und sich auf andere soziale Prozesse richten. Aus dieser Perspektive stehen soziale Prozesse als das Mittel, mit dem Unsicherheit reduziert wird, im Fokus. Prozesse der Softwareentwicklung und –einführung werden als aufeinander aufbauende Modellierungsprozesse betrachtet, deren Ziel die Produktion von Modellen sozialer Phänomene ist, die möglichst frei von Unsicherheit sind. Bei der Softwareentwicklung werden dabei primär Modelle produziert, die in Softwarecode überführt werden sollen. Bei der Softwareeinführung werden primär Modelle geschaffen, die die Form von formalen und informellen Regeln für den Umgang mit Software annehmen. Das Customizing bildet eine Mischform, denn dabei können beide Arten von Modellen entstehen (Mormann 2016; Pollock und Cornford 2004). Nachdem die Akteur*innen bei der Entwicklung und Einführung (und eventuell beim Customizing) eine Vielzahl an Entscheidungen über Modelle getroffen und damit Unsicherheiten kontrolliert haben, die beim Umgang mit Software auftreten können, sind bei der Nutzung scheinbar nur noch wenige Unsicherheiten übrig. Durch die Entscheidung, im Prozess auf Modelle zurückzugreifen, sind bestimmte Interpretations- und Handlungsspielräume der Akteur*innen von vorneherein begrenzt. Prinzipiell können Einzelheiten der in den Modellen vorgenommenen

Reduktionen zwar jederzeit hinterfragt oder abgelehnt werden. Faktisch betrifft solche Kritik jedoch immer die Modelle als Ganzes und richtet sich damit auch an die Beteiligten der Prozesse, in denen sie entstanden sind und in denen über ihre Nutzung entschieden wurde. Statt sich nur mit den Einzelheiten der Nutzung auseinanderzusetzen, müssen Akteur*innen, die sich an diesen Einzelheiten stören, deutlich weitergehen und Kritik an Prozessen üben, an denen sie oft nicht beteiligt waren, deren Beteiligte ihnen möglicherweise nicht nahestehen, ihnen hierarchisch übergeordnet oder einfach unbekannt sind. Solche Kritik hat also potentiell deutlich weiter reichende Folgen, die für die Kritisierenden schwer bis gar nicht zu überblicken sind. Durch die Ablehnung von Modellen entstehen also neue Unsicherheiten, die es nicht gäbe, wenn von vorneherein auf den Einsatz von Modellen verzichtet worden wäre. Die Grenzen der möglichen Kontrolle liegen aus dieser Perspektive erstens in den Interpretations- und Handlungsspielräumen, die Akteur*innen trotz der Einflüsse haben, die verschiedene Modelle auf die Nutzung ausüben (und die eben auch dazu führen können, dass grundsätzliche Kritik geübt wird). Sie liegen zweitens darin, dass die sozialen Prozesse, in denen es eigentlich darum geht, Kontrolle auszuüben, sich gegenseitig beeinflussen und auch davon abhängig sind, was in den sozialen Prozessen geschieht, über die eigentlich Kontrolle ausgeübt werden soll. Zwischen unterschiedlichen Modellierungsprozesse bestehen Abhängigkeiten, weil die Modelle, die in ihnen produziert werden, aufeinander aufbauen und weil die Kontrolle, die durch Modellierung ausgeübt werden kann, immer nur vorläufig ist: Entscheidungen, mit denen in einem der Prozesse Unsicherheiten kontrolliert werden, können jederzeit durch Entscheidungen in einem der anderen Prozesse in Frage gestellt werden, mit Auswirkungen auf das Modell, welches bereits als fertig galt. Modellierungsprozesse sind darüber hinaus auch von anderen sozialen Prozessen abhängig, weil sie in soziale Systeme eingebettet sind, in denen über die Bedingungen entschieden wird, unter denen Modellierer*innen ihre Aufgabe erfüllen können. Diese Abhängigkeit führt dazu, dass in Modellierungsprozessen zwar Software entsteht und dass diese Software auch als Resultat von Kontrollversuchen betrachtet werden kann, die von sozialen Prozessen über andere soziale Prozesse ausgeübt werden. Aber: Wo der Ursprung dieser Kontrollversuche liegt, wann und in welchem sozialen Kontext also die Grundlagen für die Entscheidungen gelegt worden sind, mit denen in den Modellen der Software Unsicherheit reduziert wird, kann nicht eindeutig zugeordnet werden. Diese Abhängigkeiten zu anderen sozialen Prozessen werden in den Modellierungsprozessen zum Teil bewusst in Kauf genommen (Grenz 2014) oder strategisch genutzt, um Unsicherheiten in andere Prozesse auszulagern. Zum Teil bleiben sie aber auch unerkannt, weil diejenigen, die Modelle

für die Software oder für den Umgang mit Software produzieren nicht hinterfragen, welche Unsicherheitsreduktionen zu den Modellen geführt haben, die sie in ihren eigenen Prozessen einsetzen (Weltz und Ortmann 1992; Ortmann et al. 2000; Wieandt-Ledebur 2014).

In den meisten der hier diskutierten Arbeiten wird prinzipiell von den Grundannahmen ausgegangen, die im Konzept der Dualität von Ressourcen und Routinen ausgearbeitet worden sind. Übertragen auf Software besagen sie, dass Software einerseits als Ressource eingesetzt wird, um etwas zu erreichen, was ohne ihren Einsatz nicht möglich wäre. Dies gelingt, weil sie verlässliche Ereigniszusammenhänge herstellt, die Nutzer*innen nicht verstehen müssen, an die sie aber ihr eigenes Handeln anschließen können. Andererseits kann Software diese Ereigniszusammenhänge nur verlässlich herstellen, wenn sie in Routinen eingebunden ist, die es möglich machen, dass sie auf die vorgesehene Art und Weise funktioniert. Software ist also im Sozialen nicht von sich aus wirksam, sondern nur als Teil von Nutzungsprozessen, welche ihre genaue Funktionsweise beeinflussen. Übertragen in die hier verwendete Begrifflichkeit bedeutet das, dass Software für Nutzer*innen nur zur Ressource werden kann, wenn sie die Unsicherheitsreduktionen, die in ihren Modellen stecken, zumindest zum Teil übernehmen. Es bedeutet auch, dass Nutzer*innen über ihren Umgang mit Software beeinflussen, welche Unsicherheiten entsprechend der Modelle kontrolliert werden und welche Modellentscheidungen und damit Kontrollversuche in Frage gestellt oder abgelehnt werden.

Eine umfassende Soziologie der Software sollte es Forscher*innen erlauben, Software gleichzeitig als Mittel und als Resultat von Kontrollversuchen zu betrachten. Dazu ist muss es möglich sein, Erkenntnisse aus Forschungsarbeiten zu verbinden, die Software nur aus einer der beiden hier vorgestellten Perspektiven betrachten. Keiner der in Abschnitt 2.2 vorgestellten Ansätze nimmt die Komplexität der Software in der Theorie ganz überzeugend auf oder eignet sich alleine dazu, diese in der Empirie systematisch zu untersuchen. Die wenigen Autor*innen, die es versuchen (speziell Orlikowski 1992 und Rammert et al. 1998) berücksichtigen vor allem die dritte Dimension der Komplexität, die sich der für die soziologische Untersuchung von Technik charakteristischen Unterscheidung zwischen Entwicklungs- und Nutzungsprozessen entzieht, nicht oder nur am Rande. Darüber hinaus schenken sie der Bedeutung konkreter Eigenschaften von Software, die ihrer Performativität zu Grunde liegen, zu wenig Aufmerksamkeit. Im folgenden Kapitel entwickle ich daher einen konzeptionellen Rahmen, der diese Lücke schließt. Im Mittelpunkt steht die Wechselbeziehung zwischen Unsicherheit reduzierenden Modellen, die bei der Entwicklung Software als Mittel zur Kontrolle sozialer Prozesse hergestellt werden, und den

sozialen Prozessen, in denen Akteur*innen diese Modelle einsetzen, um Kontrolle auszuüben oder sich ihr zu entziehen (Abbildung 2.6). Die Grundannahmen der Dualität, über die die beiden hier vorgestellten Perspektiven zusammengeführt werden können, nehme ich auf, indem ich Software als Expert*innensystem betrachte. Über dieses Konzept lässt sich die Software gleichzeitig als Mittel zur Kontrolle sozialer Prozesse oder als Resultat von Kontrollversuchen aus sozialen Prozessen untersuchen. Wie dies gelingen kann und wie Expert*innensysteme überhaupt genau definiert sind, wird im nächsten Kapitel ausgeführt.

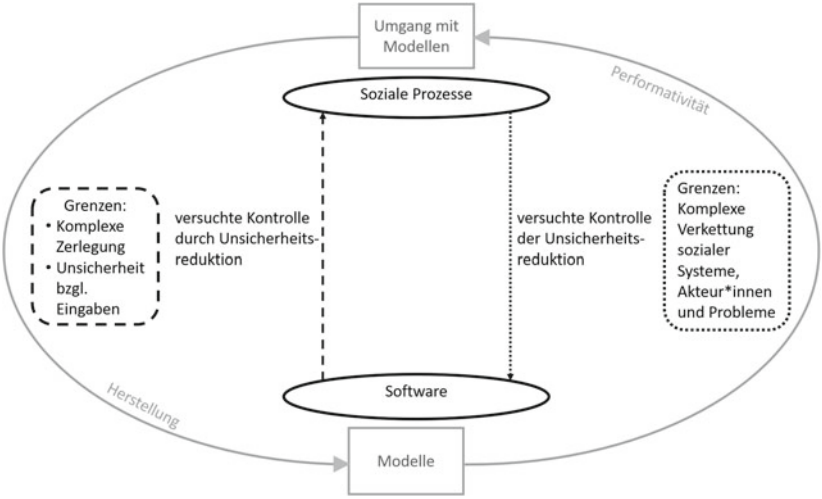

Abbildung 2.6 Die Rolle von Software im Sozialen als Verhältnis von Kontrollversuchen und Unsicherheitsreduktion

Spiele mit Expert*innensystemen

3

Der erste Teil dieses Kapitels dient der Entwicklung und Operationalisierung eines Forschungsrahmens für die Untersuchung des Umgangs mit Software als Mittel und Resultat von Kontrollversuchen. Ziel ist es, bei solchen Untersuchungen einerseits sichtbar zu machen, wie über Modelle Kontrolle über Unsicherheiten, die beim Umgang mit Software entstehen, zwischen sozialen Prozessen verschoben wird, andererseits aber auch die Grenzen dieser Kontrollverschiebung im Blick zu behalten. Beides, Kontrollversuche und ihre Grenzen, können als Ausdruck der Performativität von Software betrachtet werden: Software besteht aus Modellen, in denen Aspekte des Umgangs mit Software beschrieben werden, um diesen Umgang in Hinblick auf bestimmte Zwecke zu beeinflussen. Die Beschreibungen in den Modellen wirken sich auf das aus, was Menschen mit Software tun, ohne es vollständig zu determinieren. Mit der hier vorgeschlagenen Perspektive rückt die Performativität von Software, also das Verhältnis zwischen diesen Beschreibungen und den Auswirkungen, die sie auf den Umgang mit Software haben, ins Zentrum der Betrachtung. Diese Fokussierung auf die Performativität von Software soll Forscher*innen helfen herauszufinden, welche der Prozesse, die im Lebenszyklus der Software rekursiv miteinander verschränkt sind, relevante Auswirkungen auf die von ihnen untersuchten Formen des Umgangs mit Software haben. Im zweiten Teil des Kapitels werden Forschungsdesign und Methodik der empirischen Untersuchung vorgestellt, mit der ich in Kapitel 4 vorführe, wie soziologische Forschung aussehen kann, in der die Performativität von Software fokussiert wird.

Um der Komplexität von Software in all ihren Facetten gerecht zu werden, verzichte ich dabei auf die grundsätzliche konzeptionelle Unterscheidung zwischen Entwicklungs- und Nutzungsprozessen, die traditionell bei der soziologischen Untersuchung von Software im Mittelpunkt steht. Wie im vorangegangenen

D. Ametowobla, *Zur Soziologie der Software*,
https://doi.org/10.1007/978-3-658-37256-9_3

Kapitel gezeigt wurde, nehmen Entwickler*innen und Nutzer*innen zwar häufig unterschiedliche Haltungen gegenüber Software ein. Ihre Kontrolle darüber, wie Software die Unsicherheit in sozialen Prozessen reduziert, in denen sie zur Anwendung kommt, unterscheidet sich aber nicht grundsätzlich, sondern nur graduell voneinander. Software wird in Entwicklungs- und Nutzungsprozessen gleichermaßen performativ: In beiden wird sie von Akteur*innen nur partiell verstanden, in beiden ist sie nur zum Teil ein Mittel, mit dem die Akteur*innen Kontrolle ausüben. Immer lagern Akteur*innen, die Software einsetzen, Entscheidungen über Unsicherheiten an diejenigen aus, die die Software geschaffen haben, und geben damit einen Teil der Kontrolle über ihre eigenen Prozesse ab. Vor diesem Hintergrund reicht es nicht, den Umgang mit Software als ein Verhältnis zwischen Prozessen zu betrachten, deren Position im Lebenszyklus von vorneherein festlegt, ob in ihnen Kontrolle ausgeübt wird (weil sie Entwicklungsprozesse sind) oder ob in ihnen mit Kontrollversuchen umgegangen werden muss (weil sie Nutzungsprozesse sind). Stattdessen charakterisiere ich den Umgang mit Software als eine in all diesen Prozessen angesiedelte Kombination aus Versuchen, Kontrolle auszuüben, Kontrolle abzugeben und sich Kontrollversuchen zu entziehen, indem Modelle erzeugt, angenommen oder abgelehnt werden, in denen die Unsicherheiten des Sozialen reduziert sind.

3.1 Theoretische Grundannahmen

Um die Charakterisierung des Umgangs mit Software theoretisch zu fundieren und eine Operationalisierung zu ermöglichen, die es erlaubt, Kontrollversuche, Unsicherheitsreduktionen und die Performativität von Software in sozialen Prozessen sichtbar zu machen, greife ich auf zwei etablierte Konzepte der soziologischen Theorie zurück. Modelle verstehe ich als Expert*innensysteme, die in sozialen Systemen im Rahmen von Modellierungsprozessen produziert werden. Den Umgang mit Software betrachte ich als mikropolitische Spiele im Sinne der strategischen Organisationsanalyse. Beide Konzepte werden im Folgenden kurz eingeführt und ihre Verbindung zum Forschungsrahmen dargestellt.

3.1.1 Modelle als Expert*innensysteme

Expert*innensysteme sind „Systeme technischer Leistungsfähigkeit oder professioneller Sachkenntnis, die weite Bereiche der materiellen und gesellschaftlichen Umfelder, in denen wir heute leben, prägen“ (Giddens 2010 [1995], S. 40 f.).

Das technische Wissen, aus dem sie bestehen, wird von Expert*innen in sozialen Systemen ausgehandelt. Als soziales System bezeichne ich dabei mit Giddens ein relativ dauerhaftes Geflecht sozialer Beziehungen und Interaktionen, das in sozialen Prozessen (re)produziert wird, in denen sich die beteiligten Akteur*innen an den gleichen sozialen Regeln und Ressourcen orientieren. Dieses „sich orientieren" meint nicht, dass Regeln und Ressourcen das Handeln der Akteur*innen determinieren, sondern nur, dass sie darauf in irgendeiner Form Bezug nehmen. Akteur*innen interpretieren Regeln in Abhängigkeit von den konkreten Bedingungen der Situation, in der sie handeln. Sie können sich entscheiden, die Regeln zu beugen, zu brechen oder bewusst zu ignorieren und Ressourcen einzusetzen oder darauf zu verzichten. In allen Fällen sind Regeln und Ressourcen der Systeme aber in irgendeiner Form bedeutsam für die sozialen Prozesse, in denen Akteur*innen im System miteinander interagieren. „Soziale Regeln" sind dabei weit mehr als nur die formalen oder auch nur ausformulierten Regeln, „Ressourcen" sind mehr als nur die materiellen Mittel, die Akteur*innen zur Verfügung stehen: *„The ‚rules' involved here are social conventions, and knowledge of the contexts of their application. By ‚resources' I mean ‚capabilities of making things happen', of bringing about particular states of affairs"* (Giddens 1995, S. 341). Soziale Systeme sind diesem Verständnis nach nicht abgeschlossen, sondern können überlappen oder sich überlagern (Giddens 2008 [1984]). Das kann sowohl über gemeinsame soziale Beziehungen geschehen (z. B. wenn Beschäftigte eines Unternehmens auch Mitglieder der gleichen Gewerkschaft sind), als auch dadurch, dass innerhalb der Systeme die gleichen Regeln und Ressourcen als Orientierung für das Handeln der Akteur*innen dienen (z. B. wenn die akademische Informatik ein Teil des Wissenschaftssystems ist). Der hier nur grob spezifizierte Systembegriff wird gewählt, weil er die gegenseitige Abhängigkeit von sozialer Ordnung und sozialem Handeln fasst und es möglich macht, verschiedene Formen sozialer Ordnung einerseits zu unterscheiden (über die Systeme), andererseits aber auch Überlagerungen und Wechselwirkungen zu betrachten. Darüber hinaus erlaubt es der hier gewählte Systembegriff, die strukturierende Wirkung sozialer Ordnung ernst zu nehmen und gleichzeitig die Handlungsfreiheit der Akteure zu berücksichtigen. Dies ist Voraussetzung für einen Forschungsrahmen, der über Software vermittelte Kontrolle ebenso sichtbar machen soll wie deren Grenzen. Betrachtet man Modelle als Expert*innensysteme, können sie als Formen technischer Leistungsfähigkeit und gesellschaftlicher Sachkenntnis untersucht werden, die zu bestimmten Zeitpunkten und an bestimmten Orten produziert und dann zeitlich und räumlich getrennt verwendet werden. Gleichzeitig können auch ihre Produktionsbedingungen betrachtet werden, also die *sozialen Prozesse*, in denen

Akteur*innen nicht nur Modelle herstellen, sondern auch die Regeln und Ressourcen sozialer Systeme (re)produzieren und sie dadurch aufrechterhalten und / oder verändern.

Expert*innensysteme können gesellschaftlich deswegen so bedeutsam werden, weil Akteur*innen mit ihnen Ereigniszusammenhänge herstellen können, deren Prinzipien sie nicht verstehen. Es reicht, die Expert*innensysteme in eigene Handlungsabläufe einzubauen. Akteur*innen können so Wissen nutzen, das innerhalb der sozialen Systeme generiert wird, in denen die Expert*innensysteme entstehen, ohne dieses Wissen selbst zu besitzen. Wie in Abschnitt 2.2. dargestellt, können Techniken immer als Expert*innensysteme betrachtet werden. Die Bereitschaft der Nutzer*innen, auf Wissen über die Funktionsweise der Techniken zu verzichten, die sie in ihre Handlungsabläufe einbauen, ist eine Grundvoraussetzung dafür, dass sich komplexe Techniken überhaupt entwickeln können:

> *„Denn nun muss jeweils nur eine begrenzte Anzahl von Experten die Regeln kennen und befolgen können, die einen bestimmten Ereigniszusammenhang sicherstellen. Dadurch lassen sich regelbasierte Ereigniszusammenhänge in einer Vielzahl und Komplexität realisieren, die vollkommen ausgeschlossen wäre, läge die Hauptlast der Regelbefolgung weiterhin auf Seiten des durchschnittlich kompetenten Gesellschaftsmitgliedes." (Schulz-Schaeffer 1999, S. 417)*

Komplexe Techniken basieren damit auf dem *Vertrauen*, das Nutzer*innen in das „richtige[…] *Funktionieren"* (Giddens 2010 [1995]) der Expert*innensysteme und der sozialen Systeme, die diese hervorbringen, setzen. Vertrauen bedeutet in diesem Zusammenhang nicht, dass Nutzer*innen davon überzeugt sind, dass die Technik ein erwünschtes Ergebnis garantiert. Es bedeutet nur, dass die Technik das Risiko, das erwünschte Ergebnis nicht zu erreichen, auf ein Maß reduziert, das Nutzer*innen im Kontext des Technikeinsatzes zu akzeptieren bereit sind (a.a.O., S. 50 f.). Das kann auch bedeuten, dass Nutzer*innen ernsthafte Zweifel am richtigen Funktionieren der Expert*innensysteme haben, aber das Risiko, das mit dem Nicht-Nutzen der Technik einhergeht, als höher einschätzen als jenes, das sie eingehen, wenn sie ihr Vertrauen unfreiwillig (und möglicherweise zeitlich begrenzt) dem Expert*innensystem schenken.

In den sozialen Systemen, in denen die Expert*innensysteme produziert werden, findet die für soziale Prozesse charakteristische *„lebendige […] Sinnbildung"* (Blumenberg nach Schulz-Schaeffer 1990, S. 419) für die Ereigniszusammenhänge statt, die Expert*innensysteme herstellen. Diese Sinnbildung wird damit den Prozessen entzogen, in denen Expert*innensysteme genutzt werden. Expert*innensysteme sind also Mittel, mit denen Akteur*innen versuchen, die

Unsicherheit der Nutzungsprozesse zu reduzieren. Aus dieser Perspektive lässt sich nicht nur Software (als komplexe Technik) als Expert*innensystem betrachten, sondern auch die Modelle, auf denen der Umgang mit Software basiert. Diese Betrachtungsweise hat für die Untersuchung der Rolle von Software im Sozialen mehrere Vorteile, denn sie ermöglicht es, abhängig vom Forschungsinteresse

(1) alle sozialen Prozesse, deren Produkte dazu genutzt werden, den Umgang mit Software zu kontrollieren, als systemisch eingebettete Modellierungsprozesse zu untersuchen und
(2) dabei auch die Abhängigkeiten aufzunehmen, die durch den Einsatz von Modellen in diesen Prozessen entstehen.

3.1.1.1 Einfluss sozialer Bedingungen der Modellierung auf den Umgang mit Software

Der Verlauf eines Modellierungsprozesses und damit auch sein Ergebnis, das Modell, ist in gewissem Ausmaß immer kontingent, da er von der Praxis der Akteur*innen abhängt, also davon, was sie in konkreten Situationen miteinander tun. Betrachtet man Modelle als Expert*innensysteme, so wird sichtbar, dass sie als Mechanismen der *„Entbettung“* (Giddens 2010 [1995], S. 33–43) funktionieren. Mit Entbettung ist gemeint, dass soziale Beziehungen aus den raum-zeitlich eng begrenzten Zusammenhängen gelöst werden, in denen direkte Interaktion stattfindet, und sich über diese Interaktionszusammenhänge hinaus ausdehnen. Entbettungsmechanismen stellen soziale Beziehungen her, die die Grenzen verschiedener sozialer Kontexte überschreiten. Die Bedeutung sozialer Kontexte für soziale Beziehungen wird durch Entbettung nicht geringer, sondern sie verändert sich: Entbettungsmechanismen verbinden soziale Kontexte miteinander, die nicht durch direkte Interaktionen verbunden sind. Eben dies geschieht über die aufeinander aufbauenden Modelle, aus denen Software besteht: In jedem Modellierungsprozess werden bereits vorhandene Modelle eingesetzt und damit Verbindungen zu den sozialen Systemen hergestellt, in denen diese entstanden sind. Diese Verbindungen entstehen einerseits, weil die Geschehnisse in den früheren Modellierungsprozessen die Interpretationen und Handlungsmöglichkeiten der Akteur*innen in den späteren Modellierungsprozessen beeinflussen.

Andererseits hängt die Praxis der Modellierung nicht nur von den Regeln und Ressourcen ab, die im Modellierungsprozess selbst (re)produziert werden, sondern wird auch durch die Regeln und Ressourcen geprägt, an denen sich die Akteur*innen in anderen Prozessen innerhalb des gleichen Sozialsystems orientieren. Modelle in der Software verbinden soziale Systeme miteinander, nicht

nur einzelne Prozesse. Nicht nur die Softwareentwicklung im engeren Sinn, sondern alle sozialen Bedingungen, unter denen Software entwickelt worden ist, sind daher potentiell relevant für Forscher*innen, die untersuchen wollen, welche Auswirkungen die Software auf die Praxis der Akteur*innen hat.

3.1.1.2 Die Bedeutung von Vertrauen für den Umgang mit Software

Wird Software als Ergebnis einer komplexen Verkettung von Modellierungsprozessen innerhalb von sozialen Systemen betrachtet, lassen sich Modelle und ihre Bedeutung für die Reduktion von Unsicherheit auf zwei Arten gleichzeitig betrachten. Erstens lässt sich untersuchen, wie Unsicherheit reduziert wird, um Modelle zu erzeugen. Nennen wir die Modelle, die als Resultat der Unsicherheitsreduktion untersucht werden, für den Augenblick Modelle vom Typ B. Zweitens lässt sich untersuchen, wie bestehende Modelle genutzt werden, um festzulegen, welche Aspekte der Unsicherheit in den Modellen vom Typ B überhaupt behandelt werden sollen. Nennen wir diese bestehenden Modelle, die als Mittel zur Reduktion von Unsicherheit eingesetzt werden, Modelle vom Typ A. Aus der hier vorgeschlagenen Perspektive rückt das Vertrauen der Modellierer*innen in Modelle in den Mittelpunkt. Die Akteur*innen müssen in den Modellierungsprozessen, in denen sie Modelle vom Typ B schaffen, darauf vertrauen, dass die Modelle vom Typ A, die sie benutzen, um ihre eigene Modellierungsaufgabe zu erfüllen, im Sinne dieser Aufgabe „richtig“ sind. Für die bei der Untersuchung von Software relevanten Modellierungsprozesse drückt sich dieses Vertrauen dadurch aus, wie die Akteur*innen mit folgenden Fragen umgehen:

(1) Inwieweit sind die Angaben der Hersteller*innen über die Leistungen des Modells vom Typ A zutreffend?

Zum Beispiel: Erfüllt die verwendete Komponente tatsächlich die Anforderungen des angegebenen Sicherheitsstandards zur Verschlüsselung von Kommunikationsdaten, oder wurden die Anforderungen des Standards nur nachlässig umgesetzt?

(2) Inwieweit stimmt das soziale Phänomen, das Modell A abbildet, mit dem überein, das die Nutzer*innen im Modell B modellieren wollen?

Zum Beispiel: Ist die Vergabe von Wohnheimplätzen an Studierende eine Variante der Vergabe von Hotelzimmern an Gäste? Brauchen also Mitarbeiter*innen der Wohnheimverwaltung des Studierendenwerks ähnliche Funktionen, arbeiten sie

nach vergleichbaren Logiken wie die Mitarbeiter*innen von Hotels, wenn sie diese Aufgabe erledigen?

(3) Inwieweit ermöglicht es die Art und Weise, wie die Unsicherheit der sozialen Phänomene in den Modellen A reduziert worden ist, die Unsicherheiten des sozialen Phänomens, das im Modell B modelliert wird, so zu kontrollieren, dass das Modell B den Zwecken genügt, die im Rahmen des Modellierungsprozesses verfolgt werden?

Zum Beispiel: Enthält der Entwurf alle Funktionen, die bei der Implementierung programmiert werden müssen, damit spätere Nutzer*innen der fertigen Software ihre Arbeitsprozesse erfolgreich durchführen können?

Wenn das Konzept der Expert*innensysteme genutzt wird, um die Modelle zu beschreiben, aus denen Software aufgebaut ist, so zeigt sich, dass der Einsatz von Modellen in anderen Modellierungsprozessen als Ausdruck des Vertrauens betrachtet werden kann. Die Modelle vom Typ A, die als Grundlage für Modelle vom Typ B genutzt werden, werden von Softwareentwickler*innen in der hier dargestellten Hinsicht als „richtig“ wahrgenommen. Insbesondere lässt sich über die eben aufgeführten drei Fragen sichtbar machen, welche Unsicherheiten durch den Einsatz von Modellen des Typs A der Kontrolle innerhalb des Modellierungsprozesses für Modelle des Typs B entzogen werden. Durch den Einsatz von Modellen vom Typ A werden alle Entscheidungen über die Unsicherheiten, die in ihnen reduziert sind, an die sozialen Systeme ausgelagert, in denen diese Modelle hergestellt worden sind. Durch diese Auslagerung kommen die Unsicherheiten im Modellierungsprozess für die Modelle des Typs B nicht mehr zur Sprache, oder nur noch, indem das Vertrauen in die Modelle vom Typ A in Frage gestellt wird. Solange Akteur*innen, die Modelle vom Typ B schaffen, in die Richtigkeit der Modelle vom Typ A vertrauen, geben sie Kontrolle an die Akteur*innen ab, die die Modelle vom Typ A geschaffen haben.

Die Ausführungen in Kapitel 2 zeigen, dass allein aufgrund theoretischer Überlegungen kein einzelner Aspekt von Software identifiziert werden kann, an dem sich ihre Rolle im Sozialen allgemein untersuchen ließe. Kein einzelner Aspekt von Software ist an sich bedeutsam für das Soziale, und keiner ist für alles, was mit Software im Sozialen getan werden kann, gleichermaßen relevant. Jeder Standard, jede Projektdefinition, Spezifikation oder wiederverwendete Softwarekomponente, jedes Testkonzept oder organisationsspezifische Benutzer*innenhandbuch haben Anteil an den Unsicherheitsreduktionen, über die der Umgang mit Software vorstrukturiert wird. Jedes dieser Elemente kann als Modell eines sozialen Phänomens betrachtet werden. Gleichzeitig ist jedes dieser

Elemente das Produkt eines Modellierungsprozesses, der innerhalb von sozialen Systemen abläuft. Als Ursprung von Expert*innensystemen können daher nicht nur Professionen wie zum Beispiel die Informatik betrachtet werden, sondern auch Standardisierungsgremien, Softwarehersteller*innen, Customizingprojekte oder Fachabteilungen, in denen Regeln für den Umgang mit neu erworbener Software ausgehandelt werden. Was als relevantes Modell und was als produzierendes soziales System untersucht werden soll, ist nicht durch den Forschungsrahmen vorgegeben, sondern lässt sich nur in Abhängigkeit von der konkreten Software und dem Forschungsinteresse bestimmen. Die Entscheidung für ein Modell oder mehrere zu untersuchende Modelle und die dazugehörigen sozialen Systeme ist damit eine Frage des Untersuchungsdesigns. Damit wird keine theoretische Aussage darüber getroffen, ob ein bestimmtes Element der Software prinzipiell entscheidender für die Kontrolle von Nutzungsprozessen wäre als ein anderes. Da soziale Systeme sich überlappen und überlagern, können mit der Betrachtung von Expert*innensystemen und den sozialen Systemen, aus denen sie stammen, auch Wechselwirkungen zwischen sozialen Prozessen in verschiedenen sozialen Systemen (z. B. Kunden- und Beratungsunternehmen beim Customizing) erfasst werden. Solche Wechselwirkungen werden immer dann zum Gegenstand der Untersuchung, wenn der Modellierungsprozess durch sie beeinflusst wird.

Das Konzept der Expert*innensysteme erlaubt es, die Einbettung von Prozessen der Modellbildung in relevante Kontexte aufzunehmen und die Wechselwirkungen zwischen Modellen und Modellierungsprozessen sowie zwischen Modellierungsprozessen und anderen sozialen Prozessen sichtbar zu machen. Dieser Perspektive liegen jedoch zwei Annahmen zugrunde, von denen nicht bei allen Untersuchungen des Umgangs mit Software ausgegangen werden kann. Die erste Annahme ist, dass die (selbst produzierten und im Prozess herangezogenen) Modelle von den Akteur*innen **als Technik** genutzt werden: als Mittel, um vorher definierte Zwecke zu erreichen, die mit den über die Technik (vermeintlich) hergestellten Ereigniszusammenhängen in Verbindung stehen. Die zweite Annahme ist, dass den im Prozess herangezogenen Modellen **Vertrauen** entgegengebracht wird (im oben beschriebenen Sinn). Das Konzept der Expert*innensysteme ist also dann nützlich, wenn wir verstehen wollen, wo und unter welchen Bedingungen die Modelle entstanden sind, die in sozialen Prozessen genutzt werden. Jede Nutzung bringt mit sich, dass die Art und Weise, wie Unsicherheiten in den Modellen reduziert werden, von den Nutzer*innen an- oder zumindest hingenommen wird. Der Fokus auf Technik und das in sie gesetzte Vertrauen erlaubt es zu untersuchen, wohin Kontrolle ausgelagert wird und welche Interpretations- und Handlungsspielräume durch den Umgang mit Software verschlossen werden.

Selbst in den Prozessen der Softwareentwicklung geht es allerdings nicht nur darum, die zukünftige Nutzung zu gestalten. Auch Softwareentwickler*innen sollen und wollen mit ihrer Arbeit Gewinn erzielen, persönliche Ziele erreichen, ungeliebte Kolleg*innen vorführen und viele andere Dinge erreichen. Kurz: Bei jedem Umgang mit Software werden gleichzeitig auch noch andere Zwecke als die offiziellen verhandelt. Software wird in allen Kontexten, in denen mit ihr umgegangen wird, eben **nicht nur** als Technik genutzt, sondern zum Beispiel auch als Produkt, als Ausweis des eigenen Erfolgs, als Verhandlungsmasse in organisationsinternen Konflikten etc. Auch in diesen Fällen handelt es sich um einen Umgang mit Software. Wenn Software nicht als Technik genutzt wird, stellt sich die Frage, inwieweit die Akteur*innen, die mit Software umgehen, auf die Richtigkeit der herangezogenen Modelle vertrauen. Auch lässt sich fragen, welche Folgen es für einen Prozess hat, in dem mit Software umgegangen wird, wenn dieses Vertrauen bei einigen, aber nicht bei allen Beteiligten vorhanden ist. Nur weil Software und die Modelle, auf denen sie basiert, verfügbar sind, werden sie nicht zwangsläufig von allen Akteur*innen genutzt. Noch weniger setzen sie diese unbedingt für die offiziellen Zwecke ein, also für die Zwecke, die bei der Entwicklung oder bei der Einführung der Software in das aktuelle soziale System ausgehandelt worden sind. Weder der Einsatz als Technik noch das Vertrauen in die Modelle sind selbstverständlich. Akteur*innen können Software und die Modelle, auf denen sie basiert, ebenso ignorieren, strategisch darauf Bezug nehmen oder das Vertrauen in sie grundsätzlich in Frage stellen, um Ziele zu erreichen, die mit den offiziellen nichts zu tun haben. Die Unterscheidung zwischen den Zielen eines Systems und den Zielen der Akteur*innen, die in irgendeiner Form am System mitwirken, ist grundlegend für die meisten organisationssoziologischen Perspektiven (z. B. Simon 1997 [1945], S. 14 f.). In techniksoziologischen Konzepten wie dem, in dem Technik als Expert*innensystem betrachtet wird, steht diese Unterscheidung jedoch nicht im Vordergrund. Daher muss das Konzept der Expert*innensysteme, durch welches Zweckgerichtetheit und Vertrauen im Umgang mit Modellen betont wird, für die Untersuchung konkreter sozialer Prozesse durch Konzepte ergänzt werden, die es erlauben, differierende Zwecke sowie Produktion und Destruktion von Vertrauen in Modelle sichtbar zu machen.

Für meinen Forschungsrahmen wähle ich zu diesem Zweck die strategische Organisationsanalyse, da sie einen differenzierteren Blick auf das Verhältnis von Unsicherheit und Kontrolle ermöglicht: Akteur*innen können versuchen, Kontrolle über soziale Prozesse zu gewinnen, indem sie über Modelle Unsicherheiten in diesen Prozessen reduzieren. Sie können aber ebenso die von den Modellen angebotenen Formen der Unsicherheitsreduktion in Frage stellen, um sich der

Kontrolle zu entziehen, die andere mit Hilfe von Modellen auszuüben versuchen. Mit der strategischen Organisationsanalyse lässt sich sichtbar machen, dass Kontrolle beim Umgang mit Software in Geflechten sozialer Beziehungen hin- und hergeschoben wird und jederzeit in Frage gestellt werden kann. Ich betrachte dabei die Modelle als Mittel, mit dem Akteur*innen versuchen, Unsicherheiten in einem sozialen Prozess zu kontrollieren, indem sie sie an entfernte soziale Systeme auslagern. Bevor ich diesen Gedanken weiter ausführe, werde ich zuerst die Grundannahmen darlegen, auf denen die strategische Organisationsanalyse beruht.

3.1.2 Strategische Organisationsanalyse

In der strategischen Organisationsanalyse wird nach den Bedingungen, Ausprägungen und Folgen kollektiven Handelns gefragt. Grundannahme ist, dass Menschen *„relativ autonome Akteure“* (Crozier und Friedberg 1979, S. 7) mit individuellen, aber kontextabhängigen Zielen und Interessen sind, die in der Verfolgung dieser Ziele und Interessen stets aus ihrer eigenen Perspektive rational handeln. Rationalität meint dabei, dass das Handeln von Akteur*innen als Ergebnis einer Entscheidung interpretiert werden kann, die aus der Perspektive der Akteur*innen rational erscheint. Das bedeutet, dass Akteur*innen aus den Verhaltensmöglichkeiten, die sie zu haben glauben, die auswählen, die es unter den von ihnen wahrgenommenen Bedingungen am ehesten zu erlauben scheinen, ein von ihnen positiv bewertetes Ziel zu erreichen. Verhaltensoptionen, Bedingungen und Ziele sind jeweils abhängig von den konkreten Gelegenheiten, die sich Akteur*innen bieten (Friedberg 1995). Akteur*innen sind in der strategischen Organisationsanalyse also keine freien und rationalen Nutzenmaximierer*innen, die objektive Informationen über ihre Handlungsoptionen gegeneinander abwägen und die auswählen, deren Konsequenzen sie optimal finden (vgl. March und Simon 1994 [1958], S. 158). Sie sind auch mehr als Ausführende, die geskriptete Praktiken exekutieren, die ihnen als Folge institutionalisierter Strukturen die Hand leiten und jede Entscheidung überflüssig machen (vgl. Meyer 2008, S. 794 f.). Stattdessen entscheiden sie im Rahmen ihrer Möglichkeiten autonom über ihre Handlungen und berücksichtigen dabei sowohl ihre eigenen Interessen als auch die strukturellen Bedingungen, insofern sie Handlungen, Interessen und Bedingungen als relevant für die Entscheidung betrachten (Friedberg 2019).

Die „innere“ Rationalität der Akteur*innen kann dazu führen, dass ihre Handlungen für Beobachtende irrational erscheinen. Beobachter*innen kennen weder

die Ziele der Akteur*innen noch wissen sie mit Bestimmtheit, welche Handlungsmöglichkeiten diese in einer konkreten Situation wahrnehmen. Aus der Grundannahme über die Autonomie der Akteur*innen, auf der die strategische Organisationsanalyse basiert, entsteht gerade in solchen Fällen der Auftrag für die Forschenden, die innere Rationalität zu rekonstruieren. Es gilt herauszufinden, unter welchen strukturellen Bedingungen die Akteur*innen das beobachtete Handeln zum Erreichen welcher Ziele aus welchen als solche wahrgenommenen Handlungsmöglichkeiten gewählt haben (Friedberg 1995).

In der strategischen Organisationsanalyse werden soziale Prozesse als Ergebnis des Handelns relativ autonomer Akteur*innen in konkreten Handlungssystemen untersucht. Handlungssysteme werden als Geflechte von Spielen zwischen Akteur*innen verstanden, die durch ihr eigenes Handeln wiederum die Handlungsmöglichkeiten der anderen Mitglieder des Systems beeinflussen. Diese Spiele basieren auf Machtbeziehungen zwischen den Akteur*innen, die zwar hochgradig systemisch eingebettet, im Kern aber dyadisch[1] angelegt sind (Crozier und Friedberg 1979, S. 39 f.). Aus den Grundregeln der Interaktion in diesen Dyaden entstehen in jedem System konkrete Handlungsregeln. Diese werden einerseits vom Handeln der Akteur*innen im System hervorgebracht und beschränken dieses Handeln andererseits. Bevor ich mich den Bedingungen in Handlungssystemen zuwende, werden im Folgenden zunächst die Grundregeln der dyadischen Interaktion ausgeführt.

3.1.2.1 Grundregeln der Interaktion in Dyaden: Tausch, Macht und Ungewissheit

Die relativ autonomen Akteur*innen, von denen in der strategischen Organisationsanalyse ausgegangen wird, gehen Beziehungen miteinander ein, weil sie bestimmte Ziele nur durch kollektives Handeln, also in der Zusammenarbeit mit anderen, erreichen können. Für das Erreichen ihrer Ziele im kollektiven Handeln sind sie auf ein bestimmtes Verhalten ihres Gegenübers angewiesen. In jeder Beziehung geht es also um den *Tausch von Verhaltensweisen* und es besteht immer eine gegenseitige Abhängigkeit zwischen den Akteur*innen. Da sich auf beiden Seiten der Beziehung autonome Akteur*innen befinden, enthält jede Beziehung mindestens ein Element der *Ungewissheit*: Jede Akteur*in kann das Verhalten zeigen, das sich das Gegenüber wünscht, oder dieses Verhalten unterlassen (und eventuell stattdessen etwas Anderes tun). Diese Ungewissheit des Gegenübers in

[1] Die grundlegende Bedeutung der Dyaden für die strategische Organisationsanalyse ergibt sich aus der Tatsache, dass Macht als intransitiv verstanden wird. Die Beziehungen, aus denen Konstrukte kollektiven Handelns „vor allem“ bestehen (Crozier und Friedberg 1979, S. 39), sind also jeweils spezifische Beziehungen zwischen genau zwei Akteur*innen.

Bezug auf das Verhalten der Akteur*in erlaubt es der Akteur*in, einen Tausch einzugehen: Solange das Gegenüber weiß, dass die Akteur*in nicht tun muss, was es sich von ihr wünscht, hat die Akteur*in die Möglichkeit, dem Gegenüber dafür Zugeständnisse (also von der Akteur*in erwünschtes Verhalten) abzuverlangen (Friedberg 2019).

In der strategischen Organisationsanalyse wird daher *Macht* als Grundkonstante des Sozialen betrachtet. Macht wird als die Möglichkeit definiert, das Gegenüber in einer Beziehung zu erwünschtem Verhalten zu bewegen (Crozier und Friedberg 1979). Da zwei Akteur*innen nur dann eine Beziehung miteinander eingehen, wenn beide etwas voneinander brauchen, das ihnen das jeweilige Gegenüber auch verwehren kann, hat in einer Beziehung jede Seite Macht über die andere. Ein Beispiel: In einem Unternehmen besitzen Untergebene gegenüber Vorgesetzten damit ebenso Macht wie umgekehrt, im Umgang mit Expert*innensystemen gilt das gleiche für die Beziehung zwischen Expert*innen und Lai*innen. Hat jemand keine Möglichkeit, das Verhalten zu verwehren, welches das Gegenüber sich wünscht, wird diese Person in der strategischen Organisationsanalyse nicht mehr als eine Akteur*in betrachtet: *„[W]enn B seine Bereitschaft zu tun, was A von ihm verlangt, nicht mehr verweigern kann, dann ist auch keine Machtbeziehung mehr zwischen beiden möglich, denn dann existiert B gegenüber A nicht mehr als autonomer Akteur und wird lediglich ein Ding“* (Crozier und Friedberg 1979, S. 40).

Macht ist somit zwar ein Grundelement sozialer Beziehungen, sie ist in der Regel jedoch ungleich verteilt: Die Macht einer Akteur*in über das Gegenüber ist umso größer, je mehr dieses davon überzeugt ist, für das Erreichen seiner Ziele auf das Verhalten der Akteur*in angewiesen zu sein und je größer die Ungewissheit darüber ist, ob die Akteur*in das erwünschte Verhalten zeigen wird. Umgekehrt ist die Macht einer Akteur*in umso geringer, desto mehr sie davon überzeugt ist, dass sie sich im Einklang mit den Wünschen des Gegenübers verhalten muss, um ihre eigenen Ziele zu erreichen (a.a.O., S. 40 f.). Programmierer*innen haben damit zum Beispiel potentiell weniger Macht gegenüber einer Teamleitung, die aus der Arbeitsebene in die Leitungsposition aufgestiegen ist, die Arbeitsabläufe im Team kennt und den Code beurteilen kann, als gegenüber einer mit reiner Managementausbildung, die auf die Aussagen der Programmierer*innen angewiesen ist, um sich ein Urteil über deren Arbeit bilden zu können.

Die Machtverteilung innerhalb einer Beziehung ist abhängig von den für das Gegenüber relevanten Handlungsmöglichkeiten, die Akteur*innen diesem jeweils zum Tausch gegen die Verhaltensweisen anbieten können, die sie brauchen, um

ihre eigenen Ziele zu erreichen. Innerhalb einer Beziehung sind Handlungsoptionen somit *Ressourcen.* Die Handlungsmöglichkeiten aller Akteur*innen innerhalb einer Beziehung sind nicht nur durch deren Kompetenzen bestimmt, sondern beeinflusst von deren Zielen, den Vorstellungen darüber, wie sie diese Ziele erreichen können, und von der Gesamtheit ihrer Handlungsmöglichkeiten in allen anderen Beziehungen, in denen sie gleichzeitig stehen (und der Macht, die sie jeweils in diesen Beziehungen haben). Ressourcen innerhalb der Beziehung sind nur die Handlungsmöglichkeiten, die für das Gegenüber einen Wert haben. Dieser Wert hängt von den Handlungsmöglichkeiten des Gegenübers und damit auch von dessen Position in seinem eigenen Beziehungsgeflecht ab. Jeder Tausch und also das strategische Handeln der Akteur*innen insgesamt lassen sich daher nicht über die dyadische Beziehung verstehen, sondern nur, indem die konkreten Handlungssysteme betrachtet werden, in die sie eingebettet ist (a.a.O., S. 44–46). Die Führungskraft mit Managementausbildung aus dem Beispiel ist möglicherweise nicht an der Codequalität interessiert, sondern nur daran, zum Abnahmetermin ein lauffähiges Programm zu haben. Um unwillige Teammitglieder zu Überstunden zu drängen, kann sie damit drohen, ihre guten Beziehungen zur Abteilungsleitung spielen zu lassen und dafür zu sorgen, dass die Teilnahmegebühren für die nächste Konferenz in deren Spezialbereich nicht übernommen werden.

3.1.2.2 Handlungsprobleme, Handlungssysteme und die Kontrolle relevanter Ungewissheiten

Konkrete Handlungssysteme bilden die Grundeinheit der Untersuchung in der strategischen Organisationsanalyse. In einem solchen Handlungssystem kommen Akteur*innen zusammen, um ein gemeinsames Handlungsproblem zu lösen, das sie nicht (oder nur schwer) allein lösen können. Die Akteur*innen verfolgen in der Zusammenarbeit eigene, unterschiedliche und sich verändernde Ziele (a.a.O., S. 12). Die Lösung des Handlungsproblems kann, muss aber nicht eines dieser Ziele sein. Ein gemeinsames Handlungsproblem kann zum Beispiel die Erstellung eines Softwareentwurfs sein oder die Produktion eines Jahresabschlusses für ein Unternehmen mit Hilfe einer Software. Akteur*innen verfolgen in solchen Handlungssystemen zum Beispiel das Ziel, ein erprobtes Architekturmodell einzusetzen, eine Lösung zu finden, die alle Beteiligten zufrieden stellt, oder bei Vorgesetzten für die nächste Beförderung Punkte zu sammeln. Neben solchen eher konventionellen Zielen können Akteur*innen, die sich an der Lösung des Handlungsproblems beteiligen, aber auch daran interessiert sein, ihr Gehalt mit möglichst wenig Anstrengung zu verdienen oder dem rivalisierenden Team aus der Nachbarabteilung eins auszuwischen. Um das Handlungssystem auf Dauer aufrechtzuerhalten genügt es, dass alle Mitglieder bei der Verfolgung ihrer Ziele

in Kauf nehmen, dass das Handlungsproblem gelöst wird (Crozier und Friedberg 1979, S. 68). Jedes Mitglied des Handlungssystems trägt etwas Relevantes zur Lösung des Handlungsproblems bei, kann oder weiß also etwas, das notwendig ist, um das Problem zu lösen; jedes Mitglied des Handlungssystems ist damit (zumindest kurzfristig) nicht ersetzbar. Die Akteur*innen sind also zur Kooperation gezwungen, wenn sie ihre jeweiligen Ziele erreichen wollen (Friedberg 2019).

Wie im dyadischen Tausch gibt es auch in den Interaktionsprozessen[2] des Handlungssystems relevante Ungewissheiten, die der Macht im System zu Grunde liegen. Macht existiert nur, weil nicht sicher ist, was geschehen wird, und es Akteur*innen gibt, die die Wahl haben, ob und wie sie das zukünftige Geschehen beeinflussen. Ursprung dieser relevanten Ungewissheiten sind also einerseits die Unsicherheiten, die die Aufgabe, das Handlungsproblem zu lösen, mit sich bringt,[3] andererseits die relative Autonomie der Akteur*innen, deren Beiträge für die Lösung notwendig sind (Crozier und Friedberg 1979). Denn: Alle Mitglieder des Handlungssystems haben zwar ein Interesse an der Lösung des Handlungsproblems und damit auch daran, ihre Beiträge zu erbringen, aber weder steht fest, wie lange ihr Interesse anhalten wird, noch, welche Gegenleistung sie für ihre Kooperation von den anderen erwarten.

Da alle Mitglieder etwas Relevantes beizutragen haben, ist keines den anderen völlig unterworfen. Alle besitzen also Macht. Herausgehobene Machtpositionen im System besetzen jedoch nur diejenigen, die systemrelevante Ungewissheiten kontrollieren können:

> *„Denn gegenüber den relevanten Ungewißheiten eines Problems sind die Akteure nicht gleichgestellt. Diejenigen, die dank ihrer Situation, ihrer Ressourcen und ihrer Fähigkeiten (…) dazu fähig sind, diese Ungewißheiten zu kontrollieren, werden ihre Macht dazu benützen, um ihren Standpunkt anderen aufzuzwingen." (Crozier und Friedberg 1979, S. 13)*

[2] Alle Interaktionsprozesse können in der strategischen Organisationsanalyse als Tauschprozesse betrachtet werden. Wenn Handlungssysteme statt Dyaden betrachtet werden, wird sichtbar, dass die Handlungsoptionen, die jede Akteur*in sich wünscht und zum Tausch anbieten kann, sehr stark von denen aller anderen Akteur*innen im Handlungssystem abhängen. Interaktionsprozesse in Handlungssystemen lassen sich daher als vielseitige Tauschprozesse untersuchen.

[3] Ein Handlungs*problem* liegt per definitionem nur dann vor, wenn es Ungewissheiten in Bezug auf die Lösung gibt.

Wer die Interaktionsprozesse im Handlungssystem kontrollieren will, muss daher versuchen, Kontrolle über die relevanten Ungewissheiten des Systems zu erlangen. Dabei ist keineswegs festgelegt, worin diese relevanten Ungewissheiten bestehen: Sowohl die Definition des kollektiven Handlungsproblems als auch die Bestimmung von relevanten Ungewissheiten und Beiträgen sind Ergebnisse kollektiven Handelns (a.a.O., S. 10). Akteur*innen können nicht nur versuchen, bestehende Machtpositionen zu besetzen, sondern sich ebenso darum bemühen, die Bedingungen des Handlungssystems so zu verändern, dass ihre aktuelle Position zu einer Machtposition wird (a.a.O., S. 42 f.).

In Abschnitt 3.1.1 wurde dargestellt, dass Modelle als Ergebnis des Versuchs betrachtet werden können, Aspekte der Unsicherheit in sozialen Prozessen zu reduzieren und dadurch Kontrolle über diese Prozesse auszuüben. Über diese Betrachtungsweise lässt sich das Konzept der Expert*innensysteme mit der strategischen Organisationsanalyse verknüpfen: Theoretisch bieten Modelle eine Möglichkeit, Aspekte der Unsicherheit in sozialen Prozessen zu reduzieren und damit relevante Ungewissheiten aufzulösen, die in Bezug auf die Lösung des Handlungsproblems bestehen. Für die Akteur*innen im Handlungssystem werden die relevanten Ungewissheiten durch den Einsatz der Modelle kontrolliert. Sie können sie bei der Bearbeitung des Handlungsproblems (vorläufig) ignorieren – wenn sie das wollen. Die Kontrolle über diese Ungewissheiten ist mit dem Einsatz der Modelle auch keiner Teilnehmer*in der Prozesse mehr direkt zuzuordnen. Solange alle Akteur*innen auf die Richtigkeit der Modelle vertrauen, werden diese Ungewissheiten ganz einfach ausgeblendet. Das Vertrauen in die Modelle wird zur neuen relevanten Ungewissheit im Handlungssystem, hinter der die Unsicherheiten des Prozesses temporär verschwinden, weil sie ja in den Modellen reduziert sind. Die Entscheidungen, durch die diese Unsicherheiten reduziert worden sind, werden nicht mehr im betrachteten Handlungssystem getroffen. Durch die Modelle werden sie aus dem Prozess ausgelagert, für den sie relevant sind; sie sind bereits bei der Modellierung getroffen worden, und zwar in dem oder den sozialen Systemen, in denen die Modelle erstellt worden sind. Die Akteur*innen in diesen sozialen Systemen sind im Allgemeinen nicht (bzw. nicht unbedingt) an den Tauschvorgängen im Handlungssystem beteiligt, in dem die Modelle eingesetzt werden. Der Rückgriff auf Modelle kann so als Chance erscheinen, Machtpositionen innerhalb eines Handlungssystems zu beseitigen. Ein Beispiel: Wenn nur die Sekretär*in einer Abteilung in einem Großunternehmen weiß, welche Formulare wie ausgefüllt und in welcher Reihenfolge wohin geschickt werden müssen, damit das Unternehmen die Rechnung einer Lieferant*in rechtzeitig begleicht, so kontrolliert die Sekretär*in eine relevante Ungewissheit der Abteilung. Gehört die Abrechnung zu ihren vertraglich

festgelegten Aufgaben, so wird sie diese in aller Regel auch durchführen, egal, welches Mitglied der Abteilung ihr eine Rechnung vorlegt. Sie wird allerdings einen gewissen Spielraum haben und entscheiden können, ob sie sich dabei beeilt und sorgfältig arbeitet oder andere Aufgaben vorzieht und durch schnelles Abarbeiten der Aufgabe einen Fehler riskiert, der den Prozess verzögern kann. Wenn sie will, kann sie diese Kontrolle nutzen, um Entgegenkommen von den Mitgliedern ihrer Abteilung einzufordern: Es lohnt sich, ein gutes Verhältnis zur Sekretär*in zu pflegen und es damit wahrscheinlicher zu machen, dass die eigenen Aufgaben schnell und gründlich erledigt werden. Wird nun eine Software eingeführt, in die jedes Abteilungsmitglied selbst die Rechnungsdaten eingegeben kann, um eine Zahlung auszulösen, wird das Wissen um die genauen Abrechnungsprozesse für die Abteilung irrelevant. Die Mitglieder der Abteilung wissen weiterhin nicht, was genau zwischen Erhalt der Rechnung und Zahlung geschieht – dies ist nun im Modell des Abrechnungsprozesses in der Software automatisiert. Diese Ungewissheit ist aber für die Interaktionen im System nicht mehr von Bedeutung. Wer schnelle Zahlungen will, muss sich nun nicht mehr mit der Sekretär*in gut stellen, sondern nur noch darauf vertrauen, dass die Software wie erwartet funktioniert.

Durch den Einsatz von Modellen verschwinden die relevanten Ungewissheiten und Machtpositionen im Handlungssystem nicht, sie werden nur verschoben: Erstens verändert sich durch eingesetzte Modelle das Handlungsproblem und möglicherweise auch die Zusammensetzung der Akteur*innen, die etwas Relevantes zu seiner Lösung beizutragen haben. Im Beispiel tragen nach der Einführung der Abrechnungssoftware nicht mehr die Rechnungsnehmer*in und die Sekretär*in, sondern die Rechnungsnehmer*in und die Entwickler*innen der Abrechnungssoftware dazu bei, dass die Rechnung bezahlt wird. Offensichtlich ist dies, wenn es um die Bedeutung bestimmter Kompetenzen für das Handlungssystem geht: Die Kompetenz, das Modell anzuwenden, wird erst durch seinen Einsatz bei der Lösung des Handlungsproblems relevant. Die Kompetenzen, die nötig sind, um die Ungewissheiten aufzulösen, die durch das Modell reduziert werden (wie im Beispiel das Wissen der Sekretär*in um die Abrechnungsprozesse), verlieren innerhalb des Handlungssystems an Relevanz.[4] Zweitens

[4] Anschauliche Beispiele für diese Auf- und Abwertung von Kompetenzen finden sich in Barleys Studie zur Veränderung von Interaktionsstrukturen in zwei radiologischen Abteilungen nach der Einführung modernerer Verfahren der medizinischen Bildgebung: In *„a surprisingly short period of time“* (der gesamte Studienzeitraum betrug nur ein Jahr) wurden die jüngsten Radiolog*innen, denen am meisten Kompetenzen in der Interpretation der „neuen“ Bilder zugeschrieben wurde, zu den wichtigsten diagnostischen Ratgeber*innen, während die älteren bei der Lösung von diagnostischen Problemen innerhalb der Abteilung

werden die Ungewissheiten, die Modelle beseitigen, nun durch Ungewissheiten darüber ersetzt, inwieweit im Prozess auf diese Modelle vertraut werden kann. Dass die Software im Beispiel wie vorgesehen funktioniert, ist nämlich keineswegs ausgemacht, auch wenn ihre Nutzer*innen dies meist erst bemerken, wenn es zu Problemen bei der Abrechnung kommt. Zu den in 3.1.1 aufgelisteten Fragen nach der Richtigkeit der Modelle gesellt sich auch noch die Frage nach ihrem dauerhaften Funktionieren. Die Einführung der Software im Beispiel bedeutet nämlich auch, dass für Abteilungsmitglieder, die Rechnungen bezahlt wissen wollen, in Zukunft neue Themen relevant werden: Wird die Software auch nach einem Austausch der Unternehmensserver noch stabil laufen? Werden die Regeln darüber, welche Daten einzugeben sind, auch nach einer Änderung der Steuergesetze Bestand haben? Durch den Einsatz von Modellen entstehen neue dauerhafte Abhängigkeiten zwischen den Handlungssystemen, in denen die Modelle produziert, verändert, am Laufen gehalten und genutzt werden. Diese Abhängigkeiten sind keineswegs nur einseitig, wie z. B. in der Diskussion der Evolution von Software in Kapitel 2 gezeigt wurde. Die Beziehungen zu und die Kommunikation mit diesen anderen Handlungssystemen können wiederum von einzelnen Akteur*innen kontrolliert werden, wodurch neue Machtpositionen entstehen können.[5]

Durch den Einsatz von Modellen, die Unsicherheit reduzieren sollen, wird Kontrolle über den sozialen Prozess also sowohl innerhalb des Handlungssystems, als auch über dessen Grenzen hinaus, verschoben. Gerade solche Systemgrenzen überschreitende Machtfragen können mit der strategischen Organisationsanalyse gut untersucht werden. Sie bietet ein Analyseraster an, in dem der Zusammenhang zwischen Handeln, konkreten Handlungssystemen und deren Einbettung in lokale Ordnungen im Mittelpunkt steht. Dieses Analyseraster kann als Spielanalyse bezeichnet werden. Grundlage der Spielanalyse ist das Konzept der Spiele, die Akteur*innen in konkreten Handlungssystemen miteinander spielen. Dieses Konzept wird im Folgenden erläutert.

---

trotz ihrer prinzipiell größeren fachlichen Erfahrung wenig bis keine Rolle mehr spielten (Barley 1990, S. 77–79).

[5] Die Beziehung zu Umweltsegmenten ebenso wie die Kontrolle von Informationen und Kommunikationskanälen werden von Crozier und Friedberg als typische Ungewissheitsquellen in Organisationen betrachtet (Crozier und Friedberg 1979, S. 50), eine Betrachtung, die auf alle Handlungssysteme, also auch auf nicht formal organisierte, verallgemeinert werden kann.

3.1.2.3 Handlungsfelder, Spiele und Strategien

Die Handlungsmöglichkeiten aller Akteur*innen im System entstehen aus ihren Beziehungsgeflechten, auch aus denen, die über die Grenzen des Handlungssystems hinausreichen. Konkrete Handlungssysteme sind dadurch miteinander verknüpft. Sind diese Verknüpfungen so eng, dass das Handeln der Akteur*innen in einem System regelmäßige und bedeutsame Auswirkungen auf die Handlungsmöglichkeiten in den anderen Systemen hat, z. B. in einer Organisation, bilden die Systeme ein gemeinsames Handlungsfeld mit einer lokalen Ordnung (Friedberg 1995).

Die lokale Ordnung konkreter Handlungsfelder ist (vor allem im Fall formaler Organisationen oder anderer sozialer Systeme mit größerer zeitlicher Ausdehnung) relativ stabil. Das bedeutet, das Handlungsfeld besitzt zwar dauerhaft eine lokale Ordnung, wie diese genau aussieht, bleibt aber *„stets kontingent und problematisch“* (a.a.O., S. 107). Das liegt daran, dass die lokale Ordnung sich aus den Handlungsmöglichkeiten der Akteur*innen zusammensetzt, die sich permanent durch die Tauschprozesse in den Beziehungsgeflechten der Systeme verändern. Was die Akteur*innen wann mit wem tauschen oder, einfacher gesagt, wie sie miteinander interagieren, wird von ihren Handlungsmöglichkeiten beeinflusst, aber nicht vollständig bestimmt (Crozier und Friedberg 1979). In der strategischen Organisationsanalyse werden die Interaktionsprozesse in konkreten Handlungssystemen als *Spiele* betrachtet, gespielt von Akteur*innen, die in der Zusammenarbeit eigene Ziele verfolgen, strukturiert durch *Spielregeln*, die festlegen, welche Handlungen für die Akteur*innen innerhalb der lokalen Ordnung welche Folgen nach sich ziehen. Solange die Akteur*innen sich am Spiel beteiligen, unterwerfen sie sich freiwillig den durch die Regeln des Spiels gebildeten Zwängen. Dies bedeutet nicht, dass über die Regeln Einigkeit bestünde, sondern nur, dass Akteur*innen ihnen nicht entkommen können, sofern sie daran interessiert sind, ihre Interessen in diesem Handlungssystem weiter zu verfolgen (Crozier und Friedberg 1979). Die Regulierung wirkt damit nur indirekt auf das Verhalten der Akteur*innen ein: Sie können ihr Verhalten jederzeit wählen, müssen dafür aber die Konsequenzen tragen, die die Spielregeln vorgeben. Das Spiel und seine Regeln ermöglichen es allen Mitgliedern des Handlungssystems, sich den Zwängen der Kooperation mit anderen zu unterwerfen und dabei gleichzeitig einen Teil ihrer Handlungsfreiheit zu bewahren.

Die Spielregeln beschreiben Gewinne und Verluste, die bestimmte Verhaltensweisen für die Akteur*innen nach sich ziehen. Damit beschreiben sie auch, welche Ziele jede Akteur*in unter den aktuell geltenden Bedingungen mit ihrem Verhalten erreichen kann und was dafür zu tun ist. Aus den Regeln lässt sich

also eine Bandbreite an rationalen *Strategien* für jedes Mitglied im System ableiten. Diese Strategien hängen von der Position des Mitglieds im Handlungsfeld ab. Strategien sind das analytische Konstrukt, mit dem Forscher*innen aus dem beobachteten Verhalten der Akteur*innen die Spiele und Spielregeln rekonstruieren können (a.a.O., S. 73). Voraussetzung für eine solche Rekonstruktion ist die Annahme, dass Akteur*innen aus ihrer eigenen Perspektive rational handeln. Alle Strategien besitzen einen offensiven und einen defensiven Aspekt. Defensiv versuchen Akteur*innen, den ihnen verfügbaren Handlungsspielraum zu verteidigen. Offensiv versuchen sie, Gelegenheiten zu nutzen, um eigene Ziele zu erreichen. Zu den offensiven Strategien zählen dabei auch solche Verhaltensweisen, die dazu dienen, den Handlungsspielraum der Tauschpartner*innen zu reduzieren und so die Vorhersagbarkeit ihres Tuns zu steigern (Friedberg 2019).

Im Folgenden werden Spiele, in denen sich die Akteur*innen auf Strategien beschränken, in welchen sie die bestehenden Spielregeln als gegeben akzeptieren, als *Routinespiele* bezeichnet. Aber Akteur*innen *„können umgekehrt auch versuchen, das Spiel so umzuformen, dass andere ihnen zur Verfügung stehende Ressourcen darin relevant und mobilisierbar werden“* (a.a.O., S. 70 f.). Spiele, die durch solche Umformungsbemühungen erzeugt werden, werden hier *Innovationsspiele* genannt. Das Handlungsproblem, das in Innovationsspielen gelöst werden soll, ist die Regelung von Routinespielen.[6]

Mit dem Konzept der Spiele kann jede Form des Umgangs mit Software und den mit ihr verbundenen Modellen als systemisch eingebetteter sozialer Prozess untersucht werden. In diesem Prozess ist Kontrolle Gegenstand permanenter Aushandlungen. Wird der Umgang mit Software als Teil von Spielen betrachtet, wird erstens die Auslagerung von Kontrolle durch Modelle erkennbar, die bereits bei der Vorstellung des Konzepts der Expert*innensysteme in 3.1.1 diskutiert wurde: Wenn Akteur*innen vorhandene Modelle in sozialen Prozessen einsetzen, um Aufgaben zu erledigen (d. h. Zwecke zu erreichen), die in ihrem

[6] Ich nutze hier die Begrifflichkeit Innovations- und Routinespiel von Ortmann et al. (1990), es soll aber betont werden, dass die Begriffe im Rahmen der vorliegenden Arbeit allein über die hier aufgeführten, sehr allgemeinen Definitionen bestimmt sind. Ortmann et al. erarbeiten eine deutlich weitreichendere Konzeption dieser beiden Spieltypen und ihrer Zusammenhänge, die hier nicht aufgegriffen wird. In der folgenden empirischen Untersuchung dient die hier getroffene Definition ausschließlich der besseren Unterscheidung der verschiedenen im Fall beobachteten Spielformen. Eine grundlegende Analyse im Hinblick auf die für die beobachteten Innovations- und Routinespiele charakteristischen Gewinnchancen (a.a.O., S. 48 ff.) und Gewinnstrategien (a.a.O., S. 464 ff.) sowie der Zusammenhänge zwischen diesen Spieltypen (a.a.O., S. 468), die notwendig erscheint, um die Begriffe in ihrer ursprünglichen Definition zu verwenden, lag nicht im Fokus der vorliegenden Arbeit.

sozialen System ausgehandelt worden sind, lagern sie Kontrolle über Ungewissheiten innerhalb ihres sozialen Systems an die sozialen Systeme aus, in denen die Modelle entstanden sind. Zu diesen Systemen bestehen dadurch Abhängigkeiten, solange die Modelle eingesetzt werden. Betrachten wir soziale Prozesse in sozialen Systemen als Spiele im Sinne der strategischen Organisationsanalyse, können wir den Mechanismus, durch den diese Abhängigkeiten entstehen, als *Verkettung von Spielen* bezeichnen: Die Spiele, in denen Modelle eingesetzt werden, die, in denen die Zwecke ihres Einsatzes ausgehandelt werden und die, in denen sie gestalten werden, bilden eigene Einheiten, da sie je eigenen Regeln folgen, hängen jedoch wie die Glieder in einer Kette zusammen. Sie sind so miteinander verbunden, so dass jede Veränderung in einem der Spiele oder in der Verbindung zwischen zwei Spielen sich auf die gesamte Kette und jedes einzelne andere Spiel auswirkt und an jeder Stelle eine Veränderung anstoßen kann. *Kann* ist hier der operative Begriff: Keine dieser Veränderungen ist zwangsläufig, denn keines der Spiele kann von einem anderen Spiel heraus gezielt gesteuert werden. Die Folgen, die ein Anstoß an einer Stelle der Kette auf die anderen Stellen hat, sind nicht vollständig vorhersehbar. Sie sind, wie immer im Sozialen, unsicher. Mit Hilfe der strategischen Organisationsanalyse lässt sich nicht nur feststellen, dass diese Unsicherheit besteht – dies leistet schon das Konzept der Expert*innensysteme. Es lässt sich auch zeigen, dass Akteur*innen diese Unsicherheit in ihren Spielen nutzen und dadurch soziale Systeme strukturieren. An Hand des Spielkonzepts kann sichtbar gemacht werden, dass die Auslagerung von Kontrolle aus dem Handlungssystem an eine bestimmte Struktur der Kontrolle innerhalb des Handlungssystems gebunden ist. Besteht diese nicht, kann der Einsatz der Modelle völlig andere Ausprägungen annehmen, als dies in den Modellen vorgesehen ist. Um erneut das Beispiel der Abrechnungssoftware zu bemühen: Wenn die Abteilungsleiter*in verhindern will, dass jede Mitarbeiter*in ihre eigenen Abrechnungen macht, kann sie es ablehnen, die Mitarbeiter*innen für eine Schulung freizustellen, und sie anweisen, die Rechnungen weiterhin bei der Sekretär*in abzugeben. Diese kann die neue Software einsetzen und so dafür sorgen, dass die Abteilung nach außen hin regelkonform, d. h. mit Hilfe der Software, abrechnet, dass gleichzeitig aber die alte Kontrollstruktur erhalten bleibt, in der alle Rechnungen über den Tisch der Sekretär*in gehen müssen, um bezahlt zu werden. Durch die Abhängigkeiten, die über die Verwendung von Modellen zwischen Handlungssystemen hergestellt werden, können unvorhergesehene Nutzungsformen in einem Handlungssystem dazu führen, dass der Umgang mit den Modellen auch in anderen Handlungssystemen beeinträchtigt wird.

Ein anschauliches Beispiel für so eine Folge der Verkettung von Spielen findet sich in Ortmann et al.s Studie zur Einführung eines Produktionsplanungs- und -steuerungssystems (PPS-Systems) in einem Industrieunternehmen (Ortmann et al. 1990, S. 139–144): Im Modell des Produktionsprozesses, das in der Software implementiert war, war vorgeschrieben, dass jeder Arbeitsschritt erst begonnen werden konnte, wenn der vorangegangene Arbeitsschritt erfolgreich abgeschlossen worden war. Zum Beispiel konnte mit der Produktion eines Bauteils erst begonnen werden, wenn im System vermerkt worden war, dass alle Vorprodukte fertig produziert waren. Die in dem Unternehmen übliche Praxis, mit der nächsten Stufe der Produktion anzufangen, sobald die ersten nötigen Vorprodukte fertig sind, war im Modell nicht vorgesehen. Bestand ein Bauteil also aus fünf Vorprodukten, von denen eines erst eingebaut werden musste, nachdem die anderen vier in einem zwei Wochen dauernden Prozess zusammengefügt worden waren, hätte dieser zweiwöchige Prozess nach der Logik des PPS-Systems nicht beginnen können, solange das fünfte Vorprodukt nicht bereit läge. Diese Logik war für die Mitarbeiter*innen in der Produktion nicht nachvollziehbar, denn sie wollten vor allem Zeiten des Stillstands der Maschinen verhindern. Obwohl das PPS-System eingesetzt wurde, entzogen sich die Mitarbeiter*innen im Detail den Vorgaben des Modells, indem sie im großen Stil „Zwangsfreigaben“ benutzten, also eine eigentlich für Ausnahmefälle gedachte Möglichkeit, den nächsten Prozessschritt freizugeben, obwohl noch nicht alle Bedingungen erfüllt sind. Durch den Einsatz der Zwangsfreigaben behoben die Mitglieder des Handlungssystems rund um die Produktion das Problem, das die Software erzeugte und das der Lösung ihres Handlungsproblems (kontinuierliche Produktion ohne Unterbrechungen) im Weg stand. Indem sie den Ausnahmefall zum Normalfall machen, verhinderten sie die Auslagerung von Kontrolle aus dem Produktionsprozess: Trotz der Nutzung der Software wurde weiterhin in der Produktion entschieden, wann der nächste Prozessschritt anstand, und nicht in dem Handlungssystem, in dem die Vorgaben zur Modellierung des Prozesses im PPS-System festgelegt wurden. Diese (fehlende) Bereitschaft der Produktionsarbeiter*innen, Kontrolle über ihren Prozess abzugeben, hatte einschneidende Auswirkungen auf die Fachabteilungen wie zum Beispiel den Einkauf. Die Mitarbeiter*innen dieser Abteilung hatten die Vorgabe, so einzukaufen, dass der Materialbedarf der Produktion zu jeder Zeit gedeckt wäre, aber möglichst wenige Lagerflächen benötigt würden. Mit Hilfe des PPS-Systems sollten sie sich einen Überblick über den regelmäßigen und akuten Materialbedarf der Produktion und die vorhandenen Lagerbestände verschaffen. Durch die Zwangsfreigaben bildete das PPS-System aber weder den Materialbedarf noch die Lagerbestände richtig ab; stattdessen schienen die Mitarbeiter*innen in der Produktion permanent mit

Bauteilen zu arbeiten, die noch gar nicht im Unternehmen vorhanden bzw. noch nicht zu den nötigen Vorprodukten zusammengebaut worden waren. Statt ihre Aufgabe mit Hilfe der Software zu erfüllen, mussten die Mitarbeiter*innen der Einkaufsabteilung regelmäßig ins Lager und in die Produktionshallen gehen, um vor Ort zu prüfen, ob Nachschub bestellt werden musste. Da die Übersicht fehlte und ständig ein Mangel an Material angezeigt wurde, stiegen die Lagerbestände sogar an, statt wie geplant reduziert zu werden.

Mit Hilfe der strategischen Organisationsanalyse kann untersucht werden, wie Modelle in einzelnen Spielen innerhalb von Handlungssystemen eingesetzt werden, um bestehende Strukturen der Kontrolle aufrecht zu erhalten oder zu verschieben, und gleichzeitig berücksichtigt werden, welche Folgen das Geschehen in diesen Spielen für andere Spiele und Handlungssysteme hat. Dieser doppelte Blick – auf die einzelnen Spiele und auf ihre Verkettung – kann vor allem dann hilfreich sein, wenn Prozesse untersucht werden, in denen Software dauerhaft so eingesetzt wird, dass sie weder die vorgesehenen noch offensichtliche andere Zwecke erfüllt (wie im Beispiel des PPS-Systems, das in der Art und Weise, wie es in dem Industrieunternehmen eingesetzt wird, nur zusätzlichen Aufwand verursacht). Er kann auch dabei helfen zu verstehen, warum Software dauerhaft nicht eingesetzt wird, obwohl sie verfügbar wäre und keine – wiederum offensichtlichen – Gründe dagegensprechen (wie im Beispiel der Abrechnungssoftware für alle Mitarbeiter*innen, die dann nur von der Sekretär*in genutzt wird). Mit der strategischen Organisationsanalyse lassen sich aber nicht nur Spiele betrachten, in denen die Akteur*innen bereits Routinen für den Umgang mit Modellen etabliert haben. Sie bietet sich auch dafür an, die Spiele zu untersuchen, in denen es darum geht, die initialen Regeln für diesen Umgang mit Modellen festzulegen. Auch in diesen, den Innovationsspielen, spielen die Modelle selbst eine Rolle:[7] Diese Spiele drehen sich schließlich darum, inwieweit und auf welche Art und Weise die Unsicherheitsreduktionen, die die Modelle anbieten, angenommen werden sollen. Die Teilnehmer*innen spielen also um die Frage, ob das, was die Modelle vorgeben, als Teil der Spielregeln akzeptiert werden soll, wer dadurch Kontrolle über welche Ungewissheiten gewinnen kann und wer Kontrolle wohin abgeben muss. Akteur*innen können Vertrauen in die sozialen Systeme, aus denen das Modell stammt, in solchen Spielen als Ressource nutzen, um eine ihren Interessen eher entsprechende Kontrollstruktur in ihrem eigenen Handlungssystem durchzusetzen. Sie können dieses Vertrauen aber auch

[7] Diese Aussage gilt nur, wenn die Modelle in den Innovationsspielen relevant (gemacht) werden. Natürlich existieren auch Innovationsspiele, in denen Software oder mit ihr verbundene Modelle keine besondere Rolle spielen. Diese sind für die Ausführungen hier uninteressant, weil sie nicht im Bereich des Forschungsinteresses liegen.

untergraben, um eben solche Veränderungen zu verhindern oder es vortäuschen, um weitergehende Interventionen aus dem Handlungsfeld abzuwehren (und dann in Routinespielen die von den Modellen angebotenen Unsicherheitsreduktionen zu ignorieren). Sie können es gegen Zugeständnisse eintauschen, die ihren Interessen innerhalb irgendeines der Handlungssysteme entgegenkommen, in denen sie aktiv sind. Sie können, kurz gesagt, nicht nur vertrauen oder nicht vertrauen, sondern vor allem mit dem Vertrauen spielen, das in Expert*innensysteme gesetzt werden muss, wenn Modelle genutzt werden.

Die strategische Organisationsanalyse ist eine flexible Perspektive. Sie erlaubt es, unterschiedliche Formen kollektiven Handelns mit Bezug auf die lokalen Bedingungen zu untersuchen, unter denen die Akteur*innen handeln. Flexibel ist die Perspektive insofern, als sie konzeptionell offenlässt, wo diese Handlungsbedingungen ihren Ursprung haben und wie sie vermittelt werden, also zum Beispiel über Regeln oder über Technik. Die lokale Ordnung eines Handlungssystems kann zum Beispiel aus dem Handlungsfeld einer formalen Organisation stammen, aus dem einer Profession, aus dem, das sich in einem Verband organisierter Patient*inneninteressen bildet, oder aus anderen Handlungsfeldern. Forscher*innen können mit der strategischen Organisationsanalyse untersuchen, welche Bedeutung die lokale Ordnung aus der formalen Organisation für die Spiele in der Organisation hat, müssen dabei die Bedeutung anderer Handlungsfelder aber nicht ausblenden. In solchen Untersuchungen können gleichzeitig Einflüsse aus verschiedenen lokalen Ordnungen berücksichtigt werden, egal ob diese innerhalb von formalen Organisationen bestehen, in interorganisatorischen Feldern, innerhalb von Beratungs- oder Entwicklungsprojekten oder in ganz anderen geordneten Kollektiven, zum Beispiel in den durch wenige formale Regeln und vor allem durch Normen der Reziprozität geprägten Communities von Open Source Software. Der zentrale Stellenwert, den das konkrete Handlungsproblem in der Theorie für die Herausbildung eines Handlungssystems einnimmt, erlaubt es dabei, Akteur*innen zu identifizieren, die auf den untersuchten sozialen Prozess Einfluss haben: all diejenigen, die für die Lösung des Handlungsproblems notwendig sind. Wenn Forscher*innen das Konzept der Spiele nutzen, um den Umgang mit Software zu untersuchen, können sie außerdem all die Bedingungen des Handelns berücksichtigen, die Akteur*innen tatsächlich in ihren Handlungen beeinflussen, auch wenn sie ihren Ursprung in völlig anderen Handlungssystemen haben. So können formale Organisationsregeln (für den Umgang mit Software und für alles andere) ebenso betrachtet werden wie die informellen Regeln, nach denen in der Praxis gehandelt wird. Aber nicht nur diese: Für die vorliegende Forschungsfrage ist zentral, dass auch die „technischen“ Regeln berücksichtigt

werden können, also die Vorgaben für die Auswahl und Interpretation von Unsicherheiten im Prozess des Umgangs mit Software, die in den Modellen, auf denen Software basiert, gemacht werden. Mit der strategischen Organisationsanalyse können all diese Bedingungen als Teil der Spiele aufgenommen werden, wenn Akteur*innen sich darauf in ihrem Handeln beziehen.

3.2 Vorgehen bei der Untersuchung des Umgangs mit Software

In dieser Arbeit entwickle ich ein Konzept für die soziologische Untersuchung sozio-technischer Phänomene, in denen Software eine zentrale Rolle spielt. Dass das Soziale komplex ist und voller Unsicherheit steckt, setze ich dabei als Grundannahme voraus. Dass Software komplex ist, habe ich in Kapitel 2 ausgeführt. Dabei habe ich nachgezeichnet, wie Software durch die Komplexität und die Unsicherheiten des Sozialen, die ihre Entwicklung und Weiterentwicklung prägen, eine gewisse Eigenständigkeit entwickelt. Ich habe argumentiert, dass Software daher bei der Untersuchung sozio-technischer Phänomene als eigener Einflussfaktor neben dem Handeln der Akteur*innen und den sozialen Bedingungen berücksichtigt werden muss. Was in der ersten Hälfte dieser Arbeit konzeptionell hergeleitet worden ist, soll auf den folgenden Seiten operationalisiert werden. In der Soziologie finden sich bereits eine Vielzahl von Ansätzen, mit denen das Handeln von Akteur*innen unter sozialen Bedingungen untersucht werden kann. Zu beantworten ist im Folgenden die Frage, wie genau die Rolle der Software in so einer Untersuchung berücksichtigt werden kann.

In Kapitel 2 habe ich dargestellt, welche soziale Komplexität mit der technischen Komplexität von Software einhergeht: Software ist das Produkt verschiedener, in soziale Systeme eingebetteter, oft konfliktreicher und voneinander abhängiger Modellierungsprozesse. Sie besteht aus Modellen sozialer Phänomene, in denen bestimmte Aspekte der Unsicherheit, die diese Phänomene auszeichnet, ausgeblendet werden, um den Umgang mit Software in Bezug auf einen bestimmten Zweck zu kontrollieren. Zwecke, Aspekte und Formen der Unsicherheitsreduktion werden in jedem Modellierungsprozess ausgehandelt. Bestehende Modelle beeinflussen diese Aushandlungen, ohne sie zu determinieren. Auf die gleiche Weise beeinflusst Software die sozialen Prozesse, in denen Regeln zu ihrer Nutzung entwickelt werden (die auch als Modelle betrachtet werden können, die jedoch nicht Teil der Software werden) und schließlich auch die „eigentlichen“ Nutzungsprozesse. Die Art und Weise, wie Software genutzt wird, beeinflusst ihre Weiterentwicklung, wodurch auch die reine Nutzung in

den Zyklus der Modellierungsprozesse eingebunden wird. Im Ergebnis zeigt sich Software aus soziologischer Perspektive als ein Artefakt mit rekursiver Struktur. Durch diese Struktur werden verschiedene Kombinationen von Modellen in jedem Prozess performativ, in dem in irgendeiner Form mit Software umgegangen wird. Ich habe vorgeschlagen, dass Forschende herausfinden können, auf welche Art und mit welchen Folgen für das Soziale Modelle performativ werden, indem sie bei der Untersuchung von sozialen Prozessen darauf achten, welche Kontrollversuche in den Unsicherheitsreduktionen der Modelle stecken und wie die Akteur*innen in den Prozessen mit diesen Kontrollversuchen umgehen. Im Folgenden werde ich ausführen, wie die in 3.1 vorgestellten Theorieansätze genutzt werden können, um konkrete Forschungsfragen zu bearbeiten. Dabei werde ich auch auf die Grenzen der vorgeschlagenen Vorgehensweise eingehen.

3.2.1 Identifikation relevanter Modellelemente

In Kapitel 2 ist ausgeführt, dass jede Software eine unüberschaubare Zahl an potentiell performativen Modellen enthält, so dass weder alle Unsicherheitsreduktionen noch alle Wechselwirkungen allgemein rekonstruiert werden können. Soziolog*innen können diese Komplexität im Rahmen von soziologischer Forschung beherrschen, indem sie sich auf die in der Einleitung formulierte Grundannahme besinnen, dass Software im Sozialen nur durch den Umgang der Akteur*innen mit ihr bedeutsam werden kann. Für die soziologische Analyse ist an der Software damit nur das bedeutsam, was im untersuchten Anwendungskontext von Akteur*innen aufgegriffen wird. Statt die vielfach verschränkten Modelle aus den Bauteilen, die in der Software kombiniert sind, und ihre wechselseitigen Abhängigkeiten allgemein zu rekonstruieren, können nur die Modelle und deren Beziehungen zueinander untersucht werden, die beim Umgang mit Software[8] in Bezug auf die Forschungsfrage relevant werden. Was sich untersuchen lässt, hängt also vom konkreten Forschungsgegenstand und der konkreten Forschungsfrage ab. Steht beides fest, können die Modelle und ihre Performativität aus den erhobenen Daten rekonstruiert werden.

Dazu werden im ersten Schritt die zu untersuchenden sozialen Prozesse festgelegt und das Handlungsproblem bestimmt, bei dessen Lösung die zu untersuchende Software zum Einsatz kommt. Wenn Akteur*innen in diesen Prozessen

[8] Falls bei der Forschungsfrage nicht Software, sondern noch nicht in Code überführte Modelle, wie z. B. technische Standards, im Mittelpunkt stehen, werden diese als Ausgangspunkt genutzt.

Elemente der Software nutzen, nehmen sie die angebotenen Möglichkeiten zur Unsicherheitsreduktion an und lagern gleichzeitig Kontrolle aus (in der Regel ohne dies zu reflektieren). Die Modelle, die im Prozess performativ werden, müssen also um diese Elemente herum aufgespannt werden. Widerstände im Prozess, die von den Akteur*innen direkt artikuliert werden oder sich in dauernden Konflikten rund um die Softwarenutzung zeigen, geben Hinweise darauf, welche der Unsicherheitsreduktionen umstritten sind, aber nicht umgangen werden können. Die Verbindung von angenommenen und abgelehnten Elementen gibt Hinweise darauf, auf welcher Ebene der Software die relevanten Modelle zu suchen sind. Diese Modelle müssen identifiziert werden, wenn der soziale Prozess im Zentrum der Forschungsfrage steht, aber zu Beginn der Untersuchung unklar ist, welche Modelle die Performativität verursachen, welche die Forscher*in interessiert. Die strategische Organisationsanalyse sensibilisiert in diesem Schritt für das Geflecht der sozialen Prozesse und Handlungsprobleme, die für den Umgang mit Software direkt bedeutsam sind. Nicht nur der zentrale Nutzungsprozess, sondern auch all die verketteten Spiele, die sich um ihn entfalten, werden in die Analyse aufgenommen.

Falls in der Forschungsfrage umgekehrt die Modelle festgelegt sind und nach ihren Auswirkungen auf den Umgang mit Software gesucht wird, ergeben sich die vorgesehenen Anwendungskontexte bereits aus dem Modell, da dieses – in wie abstrakter Form auch immer – zur Kontrolle einer bestimmten Menge von sozialen Prozessen geschaffen ist. Auch in diesem Fall ist die Suche nach den im Prozess genutzten Elementen (des Modells) und den Widerständen im Prozess der Ausgangspunkt der Untersuchung. Anhand dieser Elemente können Forscher*innen identifizieren, welche Unsicherheitsreduktionen und Angebote zur Kontrollverschiebung angenommen und welche abgelehnt werden.

3.2.2 Rekonstruktion von Modellen über soziale Systeme der Konstruktion

Bei der Untersuchung des Umgangs mit Software wird sichtbar, welche Aspekte der Unsicherheitsreduktion so miteinander verbunden sind, dass sie im sozialen Prozess nur gemeinsam angenommen oder abgelehnt werden können, wenn die Software ihren Zweck erfüllen soll. Widerstände im Prozess weisen darauf hin, dass es Widersprüche zwischen den Unsicherheitsreduktionen mehrerer Modelle gibt, die beim Umgang mit Software aneinandergebunden sind. So eine Verbindung kann entweder darin bestehen, dass mehrere Modelle gemeinsam in die Software eingebettet sind, oder darin, dass die sozialen Regeln, durch die der

Umgang mit Software kontrolliert werden soll, nicht mit den technischen Regeln der Softwaremodelle vereinbar sind. Beide können gemäß den Ausführungen in 3.1.1 als Modelle betrachtet werden, die in sozialen Systemen konstruiert worden sind, an denen die Akteur*innen, die die Modelle einsetzen, in der Regel nicht beteiligt sind.

Verfolgt man so die Formen der Unsicherheitsreduktion, die sich im Prozess beobachten lassen, zu den sozialen Systemen zurück, denen sie entstammen, können die Modelle in ihrer Gesamtheit rekonstruiert werden. Dazu wird auf die Erkenntnis zurückgegriffen, dass die Modelle zusammenhängende Formen der Unsicherheitsreduktion enthalten. Welche Aspekte der Unsicherheit in den Modellen wie reduziert werden, wurde innerhalb der sozialen Systeme ausgehandelt, in denen die Modelle entstanden sind. Die Unsicherheitsreduktionen spiegeln wider, welche Zwecke durch den Umgang mit den Modellen erfüllt werden sollen. Mit Hilfe von Informationen über die Modellierungsprozesse in den sozialen Systemen der Konstruktion lassen sich diese Zwecke und die damit zusammenhängenden Kontrollversuche rekonstruieren. Diese erlauben es dann, aus den beobachteten Formen der Unsicherheitsreduktion auf die für die Fragestellung relevanten performativen Modelle zu schließen.

Die (im Rahmen einer konkreten Untersuchung konstruierten) performativen Modelle dienen dazu, bei der abschließenden Analyse sowohl soziale als auch technische Bedingungen des Umgangs mit Software in systematischer Weise berücksichtigen zu können. Die Suche nach den performativen Modellen wirkt als Filter, mit dessen Hilfe die für konkrete Forschungsinteressen bedeutsamen Ausschnitte der Software identifiziert, nachverfolgt und gedeutet werden können. Wenn die sozialen Systeme, in denen sie produziert worden sind, identifiziert sind, lässt sich nachvollziehen, wohin Kontrolle über welche Aspekte der sozialen Prozesse ausgelagert wird, in denen Akteur*innen mit der Software umgehen. Da soziale Systeme mitsamt ihren Regeln und Ressourcen über längere Zeiträume stabil sind, versetzt das hier geschilderte Vorgehen Forscher*innen in die Lage, Zusammenhänge zwischen verschiedenen Modellierungsprozessen und dem Umgang mit Software nachzuvollziehen, ohne alle Prozesse direkt begleiten zu müssen. Dies ist bei den meisten Fragestellungen von Vorteil, in denen es um den Umgang mit Software geht, die bereits vor Beginn der Untersuchung existiert. Bei solchen Fragestellungen besteht keine Chance, alle Aushandlungen, die für die relevanten Modelle bedeutsam sind, direkt zu beobachten.

3.2.3 Untersuchung der Performativität durch die Analyse von Spielen

Bei der Identifikation performativer Modelle wird die strategische Organisationsanalyse genutzt, um das Handlungsfeld festzulegen, in dem der Umgang mit Software untersucht werden soll. Der Umgang mit Software kann im Anschluss an die Rekonstruktion der Modelle wiederum als Teil des Handlungsfelds analysiert werden. Dazu werden die für das Handlungsproblem zentralen Akteur*innen, Positionen und relevanten Ungewissheitszonen identifiziert und der Umgang mit Software als Ergebnis der im Handlungsfeld stattfindenden Spiele interpretiert. Durch die Untersuchung der im Handlungsfeld miteinander verketteten Spiele kann sichtbar gemacht werden, inwiefern welche Modelle Teil der Spielregeln sind. Es kann herausgearbeitet werden, von wem die Modelle unter welchen Bedingungen als Ressourcen eingesetzt werden. Schließlich kann gezeigt werden, inwieweit sich der Umgang mit Software auf die von den Modellen angebotenen Unsicherheitsreduktionen, auf die aus der Annahme resultierende Verlagerung von Kontrolle an ein anderes soziales System oder auf von den Modellen völlig unabhängige strategische Erwägungen in Bezug auf das Handlungsfeld zurückführen lässt.

3.2.4 Reichweite und Beschränkungen der Vorgehensweise

Durch die vorgestellte Vorgehensweise können Soziolog*innen bei der Untersuchung des Umgangs mit Software sowohl die Komplexität des sozialen Kontexts berücksichtigen, in den dieser Umgang eingebettet ist, als auch der Komplexität der möglichen Einflüsse Rechnung tragen, die über Software in diesen sozialen Kontext hineinwirken. Es kann untersucht werden, welche Zusammenhänge einige Modelle in Software zwischen einer überschaubaren Anzahl an sozialen Prozessen herstellen, wie und wohin einige Formen von Kontrolle durch den Umgang mit Software verschoben werden, und welche Auswirkungen einige Strukturen sozialer Systeme, die Modelle produzieren, auf den Umgang mit Software haben können. Sie erlaubt es vor allem, mit der eigenen Arbeit an die Ergebnisse anderer Studien anzuknüpfen, in denen die Entwicklung und Nutzung bestimmter Softwaremodelle untersucht worden ist. Damit unterstützt sie die langfristige Entwicklung einer Wissensbasis über bestimmte Modelle, die häufig in Software eingebettet werden.

Allgemeine Aussagen über die konkrete Performativität bestimmter Formen von Software können ohne eine solche Wissensbasis nicht getroffen werden.

Dies ergibt sich aus der Komplexität der Problemstellung, der Menge an möglichen relevanten Forschungsgegenständen und dem fragmentarischen Stand der Forschung: Wenn Soziolog*innen verallgemeinerbare Aussagen über die konkrete Performativität bestimmter Formen von Software treffen wollen, müssen sie nicht nur einen Zusammenhang zwischen bestimmten Modellen und bestimmten Formen des Umgangs mit Software aufzeigen können, sondern auch nachvollziehbar begründen, weshalb diese Umgangsformen nicht ebenso überzeugend auf andere der vielen in die Software eingebetteten Modelle oder andere soziale Bedingungen zurückgeführt werden sollten. Für solche Aussagen bedarf es umfangreicher Forschungsarbeiten, in denen Entwicklung und Nutzung der gleichen (Form von) Software aus verschiedenen Perspektiven untersucht worden sind. Bisher existieren solche Arbeiten in Ansätzen nur zu ERP-Systemen, insbesondere SAP. Diese sind das Ergebnis eines Forschungsprogramms zur Biographie von Artefakten, das mit den gleichen Argumenten zur Komplexität des Gegenstands motiviert wird, die ich hier dargestellt habe (Williams und Pollock 2012).

Die vorgeschlagene Vorgehensweise bietet Anhaltspunkte dafür, wie einige der notwendigen Eingrenzungen der Untersuchung mit anderen zusammen hängen (z. B. genutzte Softwareelemente, Modelle und passende soziale Systeme der Konstruktion), nimmt die Forscher*innen jedoch nicht aus der Verantwortung, eigene Entscheidungen zu treffen. Andere notwendige Eingrenzungen (welche Prozesse, welche Art von Modellen) bleiben völlig den Forschenden überlassen. Insgesamt kann die Vorgehensweise Forscher*innen dafür sensibilisieren, welche Aspekte bei der Untersuchung von sozio-technischen Phänomenen mit Software berücksichtigt werden müssen, ihnen jedoch nicht zentrale Entscheidungen im Forschungsprozess abnehmen. Zwei methodische Hinweise lassen sich aus den Ausführungen ableiten:

(1) Aus einer soziologischen Perspektive muss eine empirische Untersuchung des Umgangs mit Software, in der die Software komplett abstrakt behandelt wird, kritisch hinterfragt werden. Wenn Forscher*innen sich unmittelbar auf (vermutete) Veränderungen der Handlungsfähigkeit der Nutzer*innen konzentrieren, ohne am konkreten Artefakt zu überprüfen, worauf sich diese vermeintlichen Veränderungen gründen und welche Nebenfolgen sie haben, werden sie der Performativität von Software im Sozialen nicht gerecht. Statt die Frage nach den Details der Technik als Thema für Informatiker*innen abzutun, müssen Soziolog*innen, die Software für einen relevanten Gegenstand der Forschung halten, methodische Zugänge finden, die dem konkreten Forschungsinteresse angemessen sind und es erlauben, technische und soziale Bedingungen sowie den Umgang der Akteur*innen damit im konkreten Fall

gleichzeitig aufzunehmen. In solchen Zugängen müssen die Interpretationen und Handlungsmöglichkeiten der Akteur*innen sowie die asymmetrische Verteilung von Handlungsressourcen berücksichtigt werden.

(2) Performative Modelle und die über sie vermittelten Kontrollversuche liegen auf höheren Abstraktionsebenen als der detailreiche Code und können sich auf verschiedene Programme oder Softwareebenen verteilen, die beim Umgang mit Software zusammenwirken. Die Rekonstruktion von Modellen aus dem zusammengefassten Code aller zur Laufzeit aktiven Programme ist – von Zugangsproblemen abgesehen – kaum praktikabel. Aus reinen Codeanalysen lassen sich kaum Erkenntnisse über die Modelle und ihre Abhängigkeiten untereinander gewinnen. Erfolgversprechender erscheint die Analyse von Dokumenten aus einzelnen Phasen der Softwareentwicklung wie Spezifikationen, Datenmodellen u. a. Dieser stehen in der Praxis ebenfalls oft Zugangsprobleme im Weg. Selbst wenn diese ausgeräumt sind, muss jedoch immer berücksichtigt werden, dass jedes dieser Dokumente nur einen Teil der Unsicherheitsreduktionen beschreibt, die beim Umgang mit Software performativ werden, da es in ein Geflecht vor- und nachgelagerter Modellierungsprozesse eingebunden ist. Mit der vorgeschlagenen Vorgehensweise wird somit keine Methodik zur Untersuchung von Modellen festgelegt, aber auf die Grenzen bestimmter methodischer Ansätze aufmerksam gemacht.

3.3 Forschungsrahmen der empirischen Untersuchung

Im verbleibenden Teil der vorliegenden Arbeit demonstriere ich die Anwendung der bisher ausgeführten Überlegungen auf eine empirische Untersuchung. Entsprechend der in 3.2 dargestellten Vorgehensweise wird der Analyserahmen ausgehend vom empirischen Gegenstand und der Forschungsfrage konstruiert.

Gegenstand der Untersuchung ist der Umgang mit der standardisierten Krankenhaussoftware eMed im Rahmen der OP-Planung in einem großen deutschen Universitätsklinikum mit mehreren Abteilungen, in denen chirurgische Eingriffe durchgeführt werden. Die Software eMed ist ein Krankenhausinformationssystem (KIS), das auf der standardisierten Organisationssoftware SAP basiert. Für die Fallstudie greife ich auf die zu SAP verfügbare soziologische Wissensbasis zurück und demonstriere damit, wie die vorgeschlagene Vorgehensweise dazu genutzt werden kann, um bestehende Forschung zu erweitern. Dabei thematisiere ich in der Fallstudie einen im Rahmen der soziologischen Forschung zu SAP bisher unterbelichteten Aspekt: den Zusammenhang zwischen den verschiedenen

Versuchen, die Prozesse in Organisationen über Software und organisatorische Regeln zu kontrollieren, und den Handlungsmöglichkeiten, die Nutzer*innen haben, um sich diesen Kontrollversuchen zu entziehen. Am ausgewählten Fall werden besonders die in Kapitel 2 betonten systemischen Einbettungen dieser Kontrollversuche sichtbar.

3.3.1 Besonderheiten des Forschungsgegenstands

In Krankenhäusern treffen sehr unterschiedliche gesellschaftliche Anforderungen aufeinander (Iseringhausen und Staender 2012): Krankenhäuser sind Organisationen, die Menschen „bearbeiten" und dadurch in der Wahl ihrer Mittel unter besonderer Beobachtung der Gesellschaft stehen (Klatetzki 2010). Sie sollen eine „*wohlfahrtsgesellschaftliche Infrastrukturfunktion*" (Bode 2010) im Bereich Gesundheit erfüllen, akut erkrankte Patient*innen behandeln, der medizinischen Profession Raum für die Berufsausübung und die Ausbildung des Nachwuchses bieten und medizinische Forschung ermöglichen (Rohde 1974). Für all diese Aufgaben brauchen sie Geld. Krankenhäuser gelten daher als professionelle Organisationen (Mintzberg 1991), in denen mehrere konkurrierende Formen der Kontrolle aufeinandertreffen. Strukturell prallen so die klassische hierarchische Kontrolle der Organisation und die kollegiale und disziplinär geordnete Kontrolle der Profession aufeinander. In ersterer, der bürokratischen Ordnung, werden Über- und Unterordnung und korrekte Verfahren durch formale Regeln der Organisation legitimiert (Weber 2005 [1922]). In letzterer, der professionellen Ordnung der Medizin, dominieren Ärzt*innen sowohl in der Beziehung zu allen anderen medizinischen Dienstleister*innen als auch bei der Definition des legitimen Fachwissens (Freidson 1963). Dieses Wissen befähigt seine Träger*innen, mit anerkannter Autorität zu entscheiden welche Eigenschaften einer Patient*in als Symptome einer Krankheit und welche als medizinisch unauffällig gelten, welche Behandlung bei welcher Diagnose angezeigt ist, wann Patient*innen als geheilt gelten und so weiter (Abbott 1988). Die zentralen Leistungsprozesse von Krankenhäusern werden von den Mitgliedern der medizinischen Profession erbracht, Regeln für die Leistungserbringung werden größtenteils innerhalb der Profession, für das Krankenhaus also von Ärzt*innenverbänden, geregelt. Diese Regeln, in denen Symptome mit Diagnosen und Diagnosen mit Behandlungen verknüpft werden, gestehen den einzelnen Akteur*innen große Ermessensspielräume innerhalb eines stark standardisierten Rahmens an Handlungsmöglichkeiten zu: Eine Vielzahl an medizinischen Leitlinien, unüberschaubare Forschungsergebnisse und die Grundhaltung, dass jede Patient*in individuell zu betrachten ist, führen dazu,

dass Erfahrung und Intuition von Ärzt*innen als Entscheidungsgrundlage eine große Rolle spielen (Vogd 2004b, S. 23–28). Da die Kontrolle und Sanktionierung des professionellen Handelns Fachwissen voraussetzt, auf das die Profession ein gesellschaftlich legitimiertes Monopol besitzt, stehen professionelle und organisatorische Formen der Kontrolle in Krankenhäusern in Konkurrenz (Freidson 1963): Manager*innen können formale Regeln für die Patient*innenbehandlung erlassen, weil die Regelung zentraler Prozesse der Organisation zu ihren legitimen Aufgaben gehört, aber Ärzt*innen können von diesen Regeln jederzeit legitimerweise abweichen und sich dabei auf ihre professionelle Entscheidungshoheit berufen. Auseinandersetzungen um die Entscheidungshoheit zwischen Management und Ärzt*innen sind damit in der Struktur des Krankenhauses angelegt. In Universitätskliniken treffen zusätzlich zwei Formen professioneller Organisationen zusammen, nämlich die der Medizin und die der Wissenschaft. Organisationsinterne Aushandlungen können dadurch gleichzeitig durch Bezugnahme auf die Hierarchie innerhalb der Organisation, in die medizinische Profession oder die Wissenschaft beeinflusst werden.

In Krankenhäusern kommt häufig standardisierte Organisationssoftware zum Einsatz. Diese stellt den zweiten Teil des in dieser Arbeit untersuchten Forschungsgegenstands dar. Durch ihre Struktur und ihren Nutzungskontext eignet sich standardisierte Organisationssoftware besonders gut, um sowohl die in der soziologischen Forschung bisher größtenteils unterbelichtete dritte Ebene der Komplexität von Software (s. Abschnitt 2.1.3) als auch die systemische Einbettung aller Formen des Umgangs mit Software systematisch zu untersuchen: Sie ist langlebig und wird im Laufe ihrer Lebensdauer nicht nur nacheinander in unterschiedlichen Versionen, sondern auch gleichzeitig in unterschiedlichen Varianten der gleichen Version genutzt. Diese Varianten sind Folge einer kuratierten Flexibilität, die sicherstellen soll, dass möglichst viele Kund*innen die Software als Lösung für ihre speziellen Probleme akzeptieren, ohne dass dadurch grundlegende technische Strukturen verändert werden müssten. Standardisierte Organisationssoftware wird fast immer von einer formalen Organisation produziert und nicht wie andere Formen von Software, in einer Open-Source-Gemeinschaft freiwilliger Entwickler*innen. Daher lässt sich das primäre soziale System, in dem die Entwicklung stattfindet, innerhalb der Herstellerinnenorganisation der Software verorten. Die strategischen Interessen dieser Organisation sind für das Handeln in diesem sozialen System bedeutungsvoll; so werden zum Beispiel die Vielzahl an Umgangsformen und Problemen, die sich innerhalb der verschiedenen Anwendungskontexte entwickeln, bei der Weiterentwicklung berücksichtigt, aber in einer Weise, die systematisch auf die Interessen der Herstellerin abgestimmt ist (Pollock und Williams 2009). Standardisierte Organisationssoftware unterstützt eine

Vielzahl von Praktiken innerhalb formaler Organisationen. Viele dieser Praktiken sind nicht nur innerhalb der Organisation formal reglementiert, sondern müssen oft zusätzlich gesetzlichen Vorgaben, technischen und organisatorischen Standards, den Regeln von Branchenverbänden oder ähnlichen organisationsexternen Reglements folgen. Daher orientieren sich Herstellerinnen in ihren Produktentwicklungsstrategien nicht nur an bekannten oder vermuteten Nutzungspraktiken, sondern berücksichtigen auch die Einflüsse verschiedener Regulierungsinstanzen, die die unterstützten Praktiken reglementieren. Brita Hohlmann (2007, S. 16) gibt Beispiele zu den Anforderungen, die aus verschiedenen Umwelten an die Funktionalität von ERP-Systemen gestellt werden. Als relevante Regulierungsinstanzen in solchen Umwelten sind zum Beispiel Regierungs- und Nichtregierungsorganisationen zu betrachten, die Steuer-, Umwelt-, Arbeits- oder andere unternehmensrelevante Gesetze erarbeiten, Industriestandards aushandeln oder ähnliches. Darüber hinaus ist auch die Kompatibilität mit anderen für Organisationen relevanten Softwareprodukten wichtig, weil standardisierte Organisationssoftware sich an einen möglichst breiten Kund*innenkreis richtet. Bei dieser Art von Software handelt es sich insgesamt um ein technisch komplexes Artefakt, das unter komplexen Bedingungen entsteht.

Die Komplexität der untersuchten Organisation und der untersuchten Software erfordern, dass klar eingegrenzt wird, welche Art von performativen Modellen für eine Untersuchung ausgewählt werden soll. Für die vorliegende Arbeit konzentriere ich mich auf performative Modelle, die ich Modelle des Organisierens nenne. Diese Modelle definiere ich im folgenden Abschnitt genauer.

3.3.2 Fokussierte Modelle und Fragestellung

Als Modelle des Organisierens bezeichne ich performative Modelle, die Arbeitsprozesse in Organisationen so strukturieren, dass durch die Zusammenarbeit der Akteur*innen in diesen Prozessen ein fixer, im Laufe der Modellbildung ausgehandelter Zweck erreicht werden kann. Einem Modell des Organisierens liegt eine Reihe zusammenhängender Annahmen darüber zu Grunde, welche Art von Akteur*innen in Organisationen wie und wozu zusammenarbeiten. Auf diesen Annahmen basieren alle Eigenschaften des Modells. Zu diesen Eigenschaften zählen eindeutige Definitionen dieser Akteur*innen, ihrer Beziehungen zueinander und zu ihrer relevanten Umwelt und der Objekte, mit denen sie umgehen. Auch die Regeln für diesen Umgang lassen sich als Eigenschaften des Modells verstehen. In einem Modell des Organisierens wird damit eine abstrakte Vorstellung davon, was und wie organisiert wird, mit konkreten Strukturelementen,

die in Software umgesetzt werden können, verknüpft. In dieser Hinsicht ähnelt es einem organisationalen Archetyp (Greenwood und Hinings, C. R. 1993). In einem organisationalen Archetyp sind Deutungsmuster mit strukturellen Attributen und Prozessen einer Organisation zu einem konsistenten Ganzen verbunden. Wie organisationale Archetypen lassen sich Modelle des Organisierens nur empirisch aus der Interpretation der Auseinandersetzung der Akteur*innen mit ihren Elementen rekonstruieren (a.a.O., S. 1076). Ihre Einzelheiten sind zwar vollständig ausformuliert, dies geschieht aber in unterschiedlichen Modellierungsprozessen. Die Formulierungen sind somit Ergebnisse unterschiedlicher Aushandlungen. Keine dieser Aushandlungen allein bringt die Modelle des Organisierens hervor. Die Modelle, die bei der Nutzung performativ werden, lassen sich damit nur auf ein Geflecht von Modellierungsprozessen und damit Aushandlungen zurückführen. In der vorliegenden Arbeit untersuche ich die Praktiken des Umgangs mit Software bei der OP-Planung. In diesen Praktiken werden Modelle des Organisierens performativ, weil sich die Akteur*innen beim Umgang mit Software mit den Datenstrukturen, formalen Prozessvorlagen (Templates) und Berechtigungen für bestimmte Funktionen auseinandersetzen, die in die Software eingebettet sind.

Modelle des Organisierens bilden eine überschaubare Teilmenge aller in standardisierte Organisationssoftware eingebetteten Modelle: Standardisierte Organisationssoftware soll alle Arbeitsprozesse in einer Organisation informationstechnisch integrieren und es möglich machen, die Zusammenarbeit in der Organisation so zu kontrollieren, dass sie zu einem einheitlichen Organisationsziel beiträgt. Die Integration verschiedener Arbeitsprozesse und deren Ausrichtung auf das gleiche Ziel ist daher der übergeordnete Zweck der Software. Wie in Abschnitt 2.1 ausgeführt, wird dieser zu Beginn des Softwareentwicklungsprojekts ausgehandelt und dann in den verschiedenen Modellierungsprozessen der Entwicklung als unhinterfragte Rahmenbedingungen übernommen. Nur in Ausnahmefällen wird der Zweck der Software an sich während der Softwareentwicklung grundsätzlich hinterfragt. Da die Integration aller Arbeitsprozesse mit Blick auf ein einheitliches Ziel eine zentrale Vorgabe der Softwareentwicklung ist, ist davon auszugehen, dass diese einheitliche Ausrichtung auch bei der Modellierung einzelner Arbeitsprozesse bedeutsam ist. Diese einheitliche Ausrichtung kommt im Modell des Organisierens zum Ausdruck.[9] Mehr noch: Die Software kann den vorgegebenen Zweck nur erfüllen, wenn es gelingt, alle Modelle, die

[9] Damit ist nicht gesagt, dass standardisierte Organisationssoftware nur Modelle des Organisierens enthalte oder dass nur diese performativ werden könnten. Dies wäre ein Trugschluss. Standardisierte Organisationssoftware enthält wie andere Software auch Modelle von Nutzer*innen und ihren Kompetenzen, Modelle von wirtschaftlichen Zusammenhängen, Modelle menschlicher Beziehungen und viele andere Modelle. In dieser Arbeit konzentriere

in Teilprozessen geschaffen werden, in einem einheitlichen Modell des Organisierens zusammenzuführen. Enthält eine standardisierte Organisationssoftware mehrere unterschiedliche Modelle, so ist zu erwarten, dass sie nicht in ein und demselben Softwareentwicklungsprojekt entstanden sind – zumindest nicht, wenn dieses an seinem Ende als erfolgreich bewertet worden ist. Als Ursprung verschiedener Modelle des Organisierens kommen damit die langfristige Weiterentwicklung, die Entwicklung von Varianten oder die Einführung der Software in einzelne Organisationen in Frage, nicht jedoch beliebige einzelne Modellierungsprozesse im gleichen Lebenszyklus. Die Anzahl der möglichen Modelle des Organisierens, die beim Umgang mit der Software vermutlich eine Rolle spielen können, ist daher durch die Zahl der untersuchten Versionen, Varianten und Organisationen begrenzt. In der vorliegenden Arbeit werden zwei Versionen der gleichen Variante von SAP in einem Krankenhaus in ihrem Einfluss auf konkrete sozialen Prozesse in diesem Krankenhaus analysiert. Da die Software in fast allen Bereichen der Sanusklinik eingesetzt wird und eine vollständige Untersuchung aller Formen des Umgangs mit eMed weder im Rahmen der Dissertation möglich noch notwendig ist, um die Anwendung des Konzepts zu demonstrieren, das den Kern dieser Arbeit ausmacht, habe ich mich aus pragmatischen Gründen auf einige der zentralen Prozesse in der Chirurgie konzentriert. Die Fragestellung der empirischen Untersuchung lautet daher:

> Welche Rolle spielen in eMed eingebettete Modelle des Organisierens für den Umgang mit der Software bei der OP-Planung der Sanusklinik?

3.3.3 Datenerhebung und -auswertung

Zur Beantwortung der Forschungsfrage dient eine Einzelfallstudie, die auf den methodologischen Grundannahmen der Grounded Theory basiert (Strübing 2019). Ziel ist die Konstruktion einer gegenstandsbezogenen Theorie, die auf den im Forschungsprozess erhobenen Daten und Interpretationen beruht und auf diesem Prozess sowie den ihn leitenden theoretischen Vorannahmen basiert (a.a.O.). Zur Einordnung der Vorgehensweise werde ich daher zuerst die zentralen theoretischen Vorannahmen der Untersuchung explizieren und dann die

ich mich jedoch auf Modelle des Organisierens, weil diese besonders geeignet scheinen, um zu analysieren, wie Organisationssoftware externe Einflüsse in Organisationen vermittelt.

Datengrundlage und grundlegende Informationen zu den Bedingungen der Datenerhebung angeben.

Drei theoretische Vorannahmen standen zu Beginn der Untersuchung fest. Die erste lautete, dass Software über konkrete technische Strukturen im Kontext der Anwendung performativ wird. Diese Annahme wurde in den vorangegangenen Teilen der vorliegenden Arbeit erläutert und spiegelt sich in der Forschungsfrage. Die zweite Annahme lautete, dass der Umgang mit Software als Ausprägung von Praktiken betrachtet werden kann, als wiederkehrende, überindividuell geteilte, regelgeleitete und situativ adaptierbare Handlungsweisen, an denen Akteur*innen sich in ihrem Handeln orientieren (vgl. Abschnitt 2.2). Die dritte Annahme war, dass diese Praktiken von den strukturellen Bedingungen der sozialen Kontexte beeinflusst sind, in denen sie stattfinden, weil die Akteur*innen, die diese Praktiken entwickeln, umsetzen und verändern, diese Bedingungen in ihrem Handeln berücksichtigen müssen. In Organisationen sind solche strukturellen Bedingungen zum Beispiel formale und informelle Organisationsregeln und die aus ihnen abgeleiteten Machtstrukturen. In Einklang mit der Forschungsfrage stand die Organisation als bedeutsamer sozialer Kontext von Beginn an im Fokus der Untersuchung.

Die Fallstudie basiert vor allem auf qualitativen Daten. Die Daten zum Umgang mit Software wurden in der Sanusklinik erhoben. Der größte Teil der Daten zur Rekonstruktion der Modelle des Organisierens stammt aus öffentlich zugänglichen Quellen über eMed, vor allem der Softwaredokumentation für Administrator*innen und verschiedenen Studien über SAP und Krankenhausinformationssysteme. Die Hauptphase der Datenerhebung in der Sanusklinik erstreckte sich über einen Zeitraum von 12 Monaten in den Jahren 2013 und 2014. Erste Daten zur Software wurden parallel dazu erhoben, der größte Teil der Datenerhebung zur Software erstreckte sich jedoch auch auf den späteren Analyseprozess und wurde erst 2018 abgeschlossen. Die Datenauswahl folgte der Logik des theoretischen Samplings (Glaser und Strauss 2017). Für die Erhebung der Daten zum Umgang mit Software führte ich leitfadengestützte Interviews[10] mit vier Chirurg*innen aus verschiedenen Abteilungen, zwei Pflegeleitungen und zwei Sekretär*innen, die alle für Aspekte der OP-Planung mit eMed verantwortlich waren. Darüber hinaus wurden auch der OP-Koordinator, ein Anästhesist und die für das OP-Modul Verantwortliche aus der IT-Abteilung interviewt. Die Interviews fanden in zwei Phasen statt, wobei die Erkenntnisse aus der vorläufigen Auswertung früherer Interviews (mit zwei Chirurgen, einer Pflegeleitung und

[10] Die Länge der Interviews variierte zwischen 20 Minuten und zwei Stunden mit einem Durchschnitt von etwa 45 Minuten.

der IT-Verantwortlichen) in die Fragen für die späteren Interviews einflossen. Die Auswertung der ersten Interviews ergab, dass die Variation der Planungspraktiken und Softwarekonfigurationen zwischen den chirurgischen Abteilungen zu hoch war, um im Rahmen der Studie fundierte Aussagen über alle Abteilungen treffen zu können. Daher wurde die Untersuchung auf zwei in relevanten Punkten (Größe, Notfallquote und Nutzungszeit) vergleichbare Abteilungen eingeschränkt, die hier als allgemeine und spezielle Chirurgie bezeichnet werden (s. Kapitel 4). In einer dritten Phase wurde ein weiteres Interview mit der IT-Verantwortlichen geführt und der OP-Planer an einem Arbeitstag begleitet. Dabei wurde auch der OP-Planer im Tagesverlauf ein weiteres Mal interviewt. Darüber hinaus wurden im Zuge der Begleitung auch Beobachtungsprotokolle von verschiedenen Abstimmungsgesprächen mit der Pflegekoordination, Anästhesist*innen und verschiedenen Mitgliedern chirurgischer Abteilungen erstellt. Weitere Beobachtungsprotokolle wurden zur Softwarenutzung bei der OP-Planung, der anästhesistischen und pflegerischen Dokumentation und der Koordination von Änderungen im OP-Plan über eMed erstellt. Die Befragten waren Mitglieder verschiedener an der Operationsplanung beteiligter Gruppen und wurden so ausgewählt, dass alle zentralen Positionen innerhalb des Prozesses (Chirurg*innen mit und ohne Planungsberechtigung, Pflegekräfte) abgedeckt waren und auch die wichtigsten vermittelnden (OP-Koordination, Anästhesie) und externen Teilnehmer*innen des Prozesses (Verwaltungspersonal) Berücksichtigung fanden. Schließlich wurde eine standardisierte Befragung aller Nutzenden des OP-Moduls mit Planungsberechtigung durchgeführt. Vorgehen und weiterführende Ergebnisse dieser Befragung sind an anderer Stelle dokumentiert (Engelmann und Ametowobla 2017; Engelmann et al. 2017).

Zur Analyse der in der Sanusklinik installierten Variante der Software diente zusätzlich zu den Daten zum Arbeitsprozess auch das klinikspezifische Benutzerhandbuch für das OP-Modul für Pflege, Chirurgie und Anästhesie. Für die Analyse der generischen Form der Software wurde die öffentlich zugängliche Dokumentation der Module genutzt, die über die Website von SAP verfügbar ist (SAP AG 2001, o. J.a, o. J.b, o. J.c).

Zur Beantwortung der Forschungsfrage wurde wie in Abschnitt 3.2 vorgegangen. Die Ergebnisse werden im folgenden Kapitel 4 dargestellt.

Die Rolle von eMed in der OP-Planung der Sanusklinik

4

Dieses Kapitel ist der Vorstellung der Fallstudie gewidmet. Die Ausführungen entsprechen den drei Phasen der Analyse, die im vorangegangenen Abschnitt 3.2 beschrieben worden sind: In der ersten Phase wurde das zentrale Handlungsproblem, das beim Umgang mit der Software bearbeitet wird, und die für diesen Umgang relevanten Elemente der Software genauer bestimmt. In der zweiten Phase wurden die Modelle, die sich um diese relevanten Softwareelemente aufspannen, rekonstruiert. Zu diesem Zweck wurde auf eine Kombination aus Analyse der Softwaredokumentation und Sekundärdaten über Entwicklungsprozesse und die sozialen Systeme, in denen diese Elemente entwickelt worden sind, zurückgegriffen. In der dritten Phase wurden diese Analyseergebnisse genutzt, um die verketteten Spiele zu verstehen, die um den Einsatz von eMed bei der OP-Planung gespielt werden. Der Aufbau des folgenden Kapitels spiegelt diese drei Phasen grob. Nach einer Einführung in den gesellschaftlichen Kontext, in dem die OP-Planung mit eMed in der Sanusklinik betrachtet werden muss, werden zuerst die untersuchten Planungsprozesse und die für die Nutzung relevanten Elemente der Software vorgestellt. Im zweiten Schritt werden die relevanten Modelle des Organisierens aus den sozialen Systemen rekonstruiert, die als Ursprünge dieser Modelle ausgemacht worden sind. Im Anschluss folgt die Analyse der Spiele, in denen sich die Modelle des Organisierens als performativ erwiesen haben. Dabei wird dargestellt, wo Akteur*innen eMed anders als vorgesehen nutzen und sich damit den Kontrollvorstellungen widersetzen, die in der Software angelegt sind. Die Spielanalyse zeigt auch, wie beim Umgang mit eMed Kontrolle im Handlungsfeld verteilt wird. Am Ende des Kapitels kehre ich zur Forschungsfrage zurück und diskutiere in der Gesamtschau auf die vier vorgestellten Spiele, welche Rolle die in eMed eingebetteten Modelle des Organisierens für den Umgang mit der Software bei der OP-Planung in den zwei untersuchten Abteilungen der Sanusklinik spielen.

D. Ametowobla, *Zur Soziologie der Software*,
https://doi.org/10.1007/978-3-658-37256-9_4

4.1 Hintergrund: Die Sanusklinik und ihr Krankenhausinformationssystem

Wie in Kapiteln 2 und 3 ausgeführt, ist der Umgang mit Software abhängig von sozialen Bedingungen, die über den engen Rahmen der beobachteten Nutzungsprozesse hinausreichen. Vor der Darstellung der untersuchten Prozesse der OP-Planung werden daher zuerst grundlegende Entwicklungen im Gesundheitssystem und in der Sanusklinik skizziert, die für ein Verständnis des Falls notwendig sind.

4.1.1 Diagnosis Related Groups und die Ökonomisierung der Krankenhäuser

Krankenhäuser sind Organisationen, in denen medizinische und ökonomische Anforderungen und an diesen ausgerichtete Formen der Verhaltenskontrolle permanent ausbalanciert werden müssen (vgl. 3.3.1.). Diese für Krankenhäuser typische Balance hat sich in Deutschland in den letzten Jahrzehnten stark in Richtung Ökonomie verschoben: Bis in die 1970er Jahre war das Gesundheitssystem bedarfswirtschaftlich organisiert, was bedeutet, dass in den Krankenhäusern das Primat der Medizin galt und *„Vergütungsregelungen […] dazu [dienten], den Kliniken versorgungsnotwendige Mittel zuzuführen“* (Iseringhausen und Staender 2012, S. 188). Seitdem ist die Steigerung der Kosteneffizienz des Krankenhausbetriebs in den Vordergrund gerückt. Dadurch haben sich sowohl die Beziehungen zwischen den medizinischen Bereichen des Krankenhauses als auch die zwischen Medizin und Administration neu geordnet (Bode 2010): Einerseits hat sich in vielen Krankenhäusern die interne Hierarchie zwischen den verschiedenen medizinischen Abteilungen verschärft, in der zentrale „Funktionsbereiche“, wie zum Beispiel die Chirurgie, die Abläufe vorgeben, denen andere, wie zum Beispiel die Anästhesie, als nachgeordnete Abteilungen folgen müssen (Iseringhausen und Staender 2012, S. 189). Andererseits wird die Autonomie der Mediziner*innen durch größere Macht der Geschäftsführung bzw. der Verwaltungseinheiten beschnitten (a.a.O., S. 191), weil die finanziellen Mittel, über die sie verfügen, durch geänderte Gesetze zur Krankenhausfinanzierung knapper geworden sind.

Die Krankenbehandlung ist ein Arbeitsfeld mit unbestimmter Technologie und gesellschaftlich umstrittener und damit in der Praxis abstrakter Zieldefinition (Klatetzki 2010): wann ein Mensch gesund ist, wann er oder sie behandelt werden sollte und wie ist hochgradig interpretationsbedürftig. Die Antwort auf

diese Fragen unterscheidet sich von Mensch zu Mensch und von Situation zu Situation. Unter diesen Umständen ist die Verwaltung von Krankenhäusern nach einfachen Kosten-Nutzen-Betrachtungen in einem bedarfswirtschaftlich orientierten Gesundheitssystem schwierig, denn der Nutzen einer Behandlung lässt sich nicht ohne weiteres verallgemeinern oder eindeutig berechnen (Bode 2010, S. 69; Vogd 2016). In Deutschland wird seit dem Jahr 2003 das so genannte DRG-System genutzt, um dieses Problem zu lösen. Krankheitsbilder und dafür vorgesehene Behandlungen werden dabei in standardisierte Fallgruppen einsortiert, die Diagnosis Related Groups, kurz DRGs, genannt werden. Behandlungen werden nicht individuell nach Aufwand bezahlt, sondern nach einem festgelegten Satz, der pauschal anhand der Fallgruppe festgelegt ist (Feißt und Moltzberger 2016). Die sogenannten Fallpauschalen ersetzen den Anteil des Krankenhausbudgets[1], der früher auf Basis der tatsächlich durch eine Behandlung entstandenen Kosten berechnet wurde, durch Entgelte, die auf Basis der Menge an bearbeiteten Fällen berechnet werden. Dadurch ist eine Dynamik entstanden, die das Primat der Medizin bei der Entscheidung über die Behandlung unter Druck durch die Logik der Ökonomie setzt:

> *„Indem an die ursprünglich von Ingenieuren erfundenen statistischen Mittel seitens der Politik über das Recht ein Preis angeheftet wurde, konnten jetzt unterschiedliche Akteure die DRGs als Waren in Hinblick auf Gewinn- und Verlustchancen kalkulieren. Auf dem Papier hat man jetzt ein Abrechnungskonstrukt mit dem sich nun rechnen und abschätzen ließ, welche Patientengruppe und welche Behandlungsprozeduren Profit versprechen." (Vogd 2016, S. 284)*

Die DRGs ordnen den in den Krankenhäusern erbrachten medizinischen Leistungen einen eindeutigen ökonomischen Wert zu. Die Ergebnisse der Arbeit im Krankenhaus werden eindeutig berechenbar und können dadurch einfacher ökonomischen Kalkülen unterworfen werden. *„[D]ie produktive und emanzipative Kraft der ökonomischen Kalküle [liegt darin][…], divergierende Zweck-Mittel-Perspektiven in ein sozial plausibles Arrangement zu überführen"* (Vogd 2016). Weil durch die DRG eine Möglichkeit besteht, die Arbeit im Krankenhaus mit Zahlen zu bewerten, ohne Bezug auf mehrdeutige und umstrittene Ziele nehmen zu müssen, lassen sich ökonomische Kalküle mit DRGs einfach in eindeutige Organisationsregeln übersetzen. Durch solche Regeln können Entscheidungssituationen innerhalb des Krankenhauses mit Bezug auf die Wirtschaftlichkeit

[1] Das Krankenhausbudget beinhaltet in Deutschland immer auch einen fixen Posten, der nicht für die Behandlungskosten, sondern für den Unterhalt der Gebäude u.ä. gedacht wird. Dieser Anteil spielt in der vorliegenden Arbeit keine Rolle und wird daher nicht weiter ausgeführt.

(berechenbar anhand der DRGs) gelöst werden, ohne dass die Beteiligten die Widersprüche zwischen den konkurrierenden Anforderungen, denen Krankenhäuser immer ausgesetzt sind, bearbeiten müssten (Heimer 1999).

Wie Werner Vogd (2016) konstatiert, führt diese Entscheidungsstrategie in der Situation, in der sich das Gesundheitssystem seit den 1970er Jahren befindet, nicht nur zu einer Vielzahl an unterschiedlichen Auseinandersetzungen zwischen allen Akteur*innen des Gesundheitssystems um die DRGs (statt um adäquatere Krankenbehandlung zu insgesamt niedrigeren oder gleichen Kosten) und zu einer Entfremdung der Beteiligten von ihrer Arbeit, die formal der Gesundheit dienen soll (a.a.O. 299 f.). Sie führt auf Dauer auch systematisch zu einer Fehlverteilung der Mittel nach medizinischen, volkswirtschaftlichen und auch organisatorischen Kriterien, die durch immer weitere Arten der Formalisierung repariert werden soll (a.a.O., S. 291 ff.).

Ein Element dieser Dynamik der fortschreitenden Formalisierung sind immer umfassendere und stärker integrierte Krankenhausinformationssysteme (KIS). Die Menge und Komplexität der Dokumentation, die die Formalisierung mit sich bringt, lässt sich ohne Informationstechnik in Krankenhäusern kaum verwalten. In der Wissenschaft wird der Begriff Krankenhausinformationssystem als *„Oberbegriff für die Gesamtheit aller an den administrativen und klinischen Prozessen der Krankenversorgung beteiligten Systemen eines Krankenhauses“* (Gocke 2011, S. 151) verwendet. Diesem Verständnis nach sind sowohl Menschen als auch Maschinen Teil des KIS. In der Praxis wird jedoch als das KIS eines Krankenhauses meist nur die Gesamtheit der Informationssysteme bezeichnet; weder die Nutzer*innen noch die angeschlossenen medizinischen Geräte, die z. B. Diagnosedaten ins KIS einspeisen, sind dabei mitgemeint. In der vorliegenden Arbeit verwende ich den Begriff des Krankenhausinformationssystems in der weniger umfassenden Bedeutung, so wie ihn auch die Akteur*innen in meiner Fallstudie verwendet haben. Wenn hier von Krankenhausinformationssystem oder KIS geschrieben wird, ist also nur die Software gemeint. Herstellerinnen integrierter Krankenhausinformationssysteme wie der in der Fallstudie untersuchten Software eMed versprechen, dass ihre Produkte die neue Eindeutigkeit in der Krankenbehandlung steuerbar machen, indem sie die Informationen zusammenführen, die in den einzelnen Bereichen des Krankenhauses gesammelt und produziert werden. Diese Form von Software ist auf das neue, stärker ökonomisch und administrativ orientierte Krankenhaus und die standardisierte Krankenbehandlung zugeschnitten. Solche KIS werden auch von etablierten Herstellerinnen von integrierter Standardsoftware für Organisationen angeboten. In der Sanusklinik, deren OP-Planung in dieser Arbeit untersucht wird, wird mit eMed ein KIS genutzt, das auf der standardisierten Organisationssoftware der Firma SAP beruht.

4.1.2 Die chirurgischen Abteilungen in der Sanusklinik

Die Sanusklinik ist ein großes deutsches Universitätsklinikum mit über 7000 Mitarbeiter*innen und über 1000 Betten. Als Uniklinik weist sie gleichzeitig die organisatorischen Eigenheiten einer Hochschule und die eines Krankenhauses der Vollversorgung auf. Dies bedeutet zum Beispiel, dass viele der leitenden Mitarbeiter*innen der Sanusklinik Professor*innen sind, dass die Verbindung von Forschung und Lehre zu den Zielen gehört, die in der Organisation offiziell verfolgt werden sollen, und dass im Arbeitsalltag die Kooperation einer Vielzahl von medizinischen Expert*innen organisiert werden muss.

In der Sanusklinik gibt es mehr als zehn chirurgische Fachabteilungen und über vierzig Operationssäle an mehreren Standorten auf dem weitläufigen Gelände. Jeder der Fachabteilungen sind einige Operationssäle zugeordnet, in denen im Regelfall die Operationen stattfinden, für die die jeweilige Abteilung verantwortlich ist. Neben dem zentralen OP-Bereich, in dem mehr als die Hälfte der Säle liegen und in dem die meisten Fachabteilungen ihre Patient*innen operieren, gibt es kleinere Cluster von Sälen, in denen eine oder zwei Abteilungen ihr Tagesprogramm abarbeiten, sowie einzelne „Außenstellen“. Diese Säle sind einzeln über den Campus verteilt. Sie werden nur im Ausnahmefall für sterile Operationen genutzt und dienen in der Regel der Durchführung kleinerer Eingriffe unter Lokalanästhesie.

Beschäftigte, die Tätigkeiten im OP-Bereich ausführen, gehören verschiedenen Berufsgruppen an: Chirurg*innen, Anästhesist*innen, OP- und Anästhesiepflegekräfte führen gemeinsam Operationen durch und sind damit für das Kerngeschäft verantwortlich, Angehörige weiterer Berufsgruppen übernehmen Unterstützungsaufgaben wie Transport oder Reinigung. Die formelle Aufsicht über den Bereich obliegt der Person, die die Stelle der OP-Koordination besetzt.

Alle Ärzt*innen und Pflegekräfte sind hochgradig spezialisiert und müssen nach der beruflichen Grundausbildung (Medizinstudium bzw. Pflegeausbildung) zusätzliche Qualifikationen erwerben, bevor sie sich als vollwertige Mitglieder eines OP-Teams selbstständig an Operationen beteiligen dürfen. Da die Sanusklinik ein Lehr- und Ausbildungskrankenhaus ist, finden sich hier neben voll ausgebildeten und erfahrenen Ärzt*innen und Pflegekräften auch viele, die formal für die Weiterbildung notwendige oder nicht formal vorgeschriebene, aber für den weiteren beruflichen Aufstieg nützliche Erfahrung mit komplexen Operationen sammeln müssen. Unter den Chirurg*innen herrscht eine relativ hohe Fluktuation: mit Berufserfahrung aus einer renommierten Klinik ist es oft einfacher, an anderen Krankenhäusern aufzusteigen und eine Stelle als Abteilungsleiter*in zu

finden, als in die Leitungsebene der chirurgischen Abteilungen der Sanusklinik aufzurücken, in der viele Stellen von Habilitierten besetzt werden.

Chirurg*innen sind den Abteilungen nach Spezialisierung zugeordnet. Jede Abteilung wird von einer Chefärzt*in geleitet. Diese ist formal zwar dem Vorstand der Sanusklinik untergeordnet, faktisch jedoch keinerlei direkten Weisungen unterworfen, wenn es um interne Angelegenheiten ihrer Abteilung geht. Die Chefärzt*in trifft zwar grundlegende medizinische und organisatorische Entscheidungen über ihre Abteilung, überlässt aber die alltäglichen Aufgaben, die mit der Leitung der Abteilung einhergehen, in der Regel einer oder mehreren ausgewählten Oberärzt*innen, den sogenannten leitenden Oberärzt*innen. Jeder chirurgischen Abteilung ist eine Abteilung der OP-Pflege zugeordnet, an deren Spitze eine erfahrene Pflegekraft steht. Diese Position, die diese Pflegekraft besetzt, heißt in der Sanusklinik „OP-Leitung“, und diese Positionsbezeichnung wird im Alltag auch genutzt, um die Leiter*innen der Abteilungen der OP-Pflege zu bezeichnen. Pflegerische und chirurgische Abteilungen sind einander zwar zugeordnet, aber entsprechen sich nicht eindeutig: Einige Abteilungen der OP-Pflege versorgen mehr als eine chirurgische Abteilung, andere sind personell getrennt, arbeiten aber unter der gleichen Leitung. Außerdem wird zwischen den pflegerischen Abteilungen bei Engpässen Personal ausgetauscht, was zwischen den chirurgischen Abteilungen niemals geschieht. Die Anästhesie ist als übergreifende Dienstleistungsabteilung für die gesamte Klinik organisiert, Anästhesist*innen betreuen also nicht nur Operationen, sondern setzen z. B. auch Narkosen bei diagnostischen Untersuchungen oder Geburten. Jeder großen chirurgischen Abteilung ist eine eigene Einheit der Anästhesie zugeordnet. Die Leiter*in dieser anästhesistischen Einheit wird Blockleiter*in genannt (Abbildung 4.1).

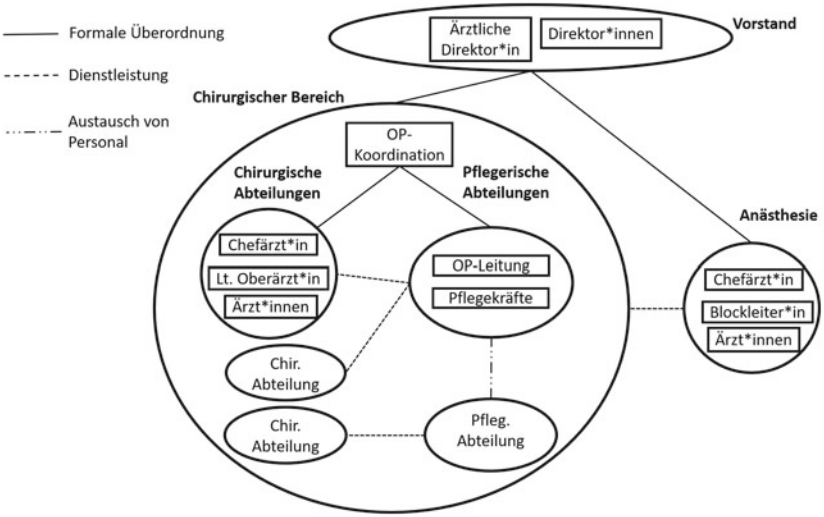

Abbildung 4.1 Chirurgie der Sanusklinik. (Eigene Darstellung)

4.1.3 Vorgeschichte: Einführung von eMed und Reorganisation

Im Jahr 2006 entscheidet der Vorstand der Sanusklinik, dass die IT vereinheitlicht werden soll. Eine Projektgruppe aus Mitgliedern verschiedener klinischer und administrativer Abteilungen und Mitarbeiter*innen des zentralen IT-Service wird beauftragt, ein Krankenhausinformationssystem auszuwählen, das möglichst viele der Anforderungen der administrativen und klinischen Bereiche erfüllen soll. Das Team entscheidet sich für das KIS eMed, einerseits, weil es standardisierte Module für die meisten Arbeitsprozesse anbietet, andererseits, weil es den IT-Expert*innen der Sanusklinik viele Möglichkeiten bietet, organisationsspezifische Anpassungen für die Software selbst zu programmieren. Die Einführung des KIS beginnt 2007 in den administrativen Abteilungen und wird im folgenden Jahr schrittweise auf die klinischen Abteilungen ausgeweitet.

Die Vereinheitlichung der IT stellt nur eines der Ziele der Softwareeinführung dar. Das KIS ist Teil einer größeren Reorganisation, die sowohl die Effizienz der Arbeitsprozesse als auch die Qualität der Dokumentation erhöhen soll. Die Dokumentation der Krankenbehandlung ist für die Klinik in zweierlei Hinsicht

bedeutsam: juristisch, weil alle Aspekte der Krankenbehandlung strengen rechtlichen Regeln unterliegen und unvollständige oder falsche Dokumentation hohe Haftungsrisiken mit sich bringt. Finanziell, weil die Kliniken von den Kostenträger*innen (in der Regel Krankenkassen) nur die Teile der Behandlung abrechnen können, die formal dokumentiert sind. Die Reorganisation der chirurgischen Abteilungen ist in dieser Hinsicht besonders wichtig, denn Operationen gehören zu den kostspieligsten und gewinnträchtigsten Leistungen, die im Krankenhaus erbracht werden. Im Zuge der Reorganisation wird mit dem OP-Bereich eine neue organisatorische Einheit oberhalb der einzelnen chirurgischen Abteilungen geschaffen. Die chirurgischen Abteilungen sollen durch die Zusammenführung in einer eigenen organisatorischen Einheit zum Informationsaustausch, zur Zusammenarbeit und zum sparsamen Umgang mit den kostspieligen OP-Ressourcen bewegt werden.

Vor der Einführung arbeiten die Mitarbeiter*innen der einzelnen chirurgischen Abteilungen größtenteils unabhängig von denen aller anderen chirurgischen Abteilungen. Jede Abteilung verfügt über eigene Säle, eigene Pflegekräfte und ein eigenes Budget. Wenn das Tagesprogramm in einer Abteilung abgearbeitet ist, stehen ihre Säle leer. Wenn an einem Tag zu viele Operationen anfallen, werden die, die an diesem Tag nicht mehr erledigt werden können, auf einen anderen Tag verschoben. Ausschließlich die Verantwortlichen in den Abteilungen entscheiden darüber, wer Zugang zu „ihren“ Sälen hat. Die aktuellen OP-Pläne kennen außerhalb der Abteilung nur die Mitarbeiter*innen der Anästhesie, die übergreifend an Operationen in allen chirurgischen Abteilungen teilnehmen. Die für die Einführung des KIS verantwortliche Mitarbeiterin des IT-Service erinnert sich: *„Das war damals unbedingt so gewünscht, weil man sollte sich nicht gegenseitig […] in die Karten kucken“*. Der Betrieb von Operationssälen ist teuer, Operationen bringen oft hohe Erlöse und die Sanusklinik hat keinen Mangel an Patient*innen, die operiert werden müssen. Für Patient*innen, die lange auf Operationen warten müssen, für Mitarbeiter*innen einzelner Abteilungen, die ständig Überstunden machen, um den Rückstand abzuarbeiten, und für den Vorstand der Sanusklinik, aus deren Budget Säle und Personal finanziert werden, ob sie genutzt werden oder nicht, ist die Autonomie der Abteilungen ein Problem.

Über das OP-Modul des KIS wird eine einheitliche Plattform für die OP-Pläne geschaffen. Die Pläne werden zwar weiterhin innerhalb der Abteilungen erstellt, sind aber über eMed für alle Mitglieder des OP-Bereichs sichtbar. Die verschiedenen Programme, die in den Abteilungen für die OP-Planung eingesetzt werden, werden abgeschaltet, die Planung in eMed wird verpflichtend. Auch die

Dokumentation der Operationen, bei der wichtige Arbeitsschritte und verwendete Materialien und Medizinprodukte festgehalten werden, muss nun in eMed erfolgen.

An der Spitze des gemeinsamen OP-Bereichs wird eine neue Stelle geschaffen, die der OP-Koordination. Die Inhaber*in dieser Position soll die Zusammenarbeit zwischen allen für Operationen relevanten Abteilungen koordinieren, soll dafür sorgen, dass die Ressourcen im neu geschaffenen OP-Bereich effizient genutzt und bedarfsgerecht zwischen den Abteilungen verteilt werden. Außerdem soll sie sicherstellen, dass alle Arbeiten vorschriftsmäßig dokumentiert werden. Sie besitzt dazu über eMed Zugriff auf die OP-Pläne und –Dokumentationen aller Abteilungen. Einsicht in alle Pläne haben durch die neue Software auch die übrigen Mitglieder der chirurgischen Abteilungen. Unter diesen gilt es jedoch als bekannt, dass die OP-Pläne in eMed nur eingeschränkt den tatsächlichen Stand der Planung wiedergeben. Kurzfristige Planänderungen, wie sie zum Beispiel bei Notoperationen geschehen, können in der allgemein verfügbaren Ansicht nicht abgebildet werden. Auch Verschiebungen in den Startzeiten, die entstehen, wenn Operationen länger dauern oder früher enden, als ursprünglich geplant, sind in der Übersicht nicht erkennbar. Zur Zeit der Erhebung besetzt Dr. Mosser[2], ein Mediziner mit Leitungserfahrung, aber kein Chirurg, die Position der OP-Koordination. Er soll auf der einen Seite langfristig die Zusammenarbeit und den Informationsaustausch in und zwischen den chirurgischen Fachabteilungen so verbessern, dass Einnahmen steigen und Kosten gesenkt werden. Dazu berichtet er dem für die Krankenversorgung verantwortlichen Vorstandsmitglied über das Geschehen im OP-Bereich und berät es bei strategischen Entscheidungen. Auf der anderen Seite vermittelt er im Alltag zwischen den Abteilungen, die für Operationen notwendige Leistungen erbringen und wirkt darauf hin, dass die Mitglieder des OP-Bereichs eMed wie vorgesehen für Planung und Dokumentation nutzen. Er organisiert den Austausch von Personal und Sälen zwischen Abteilungen, die Engpässe und Überhänge haben, zum Beispiel weil eingeplantes Personal ausfällt oder an einem Tag weniger Operationen anfallen als üblich. Außerdem achtet er darauf, dass die formalen Regeln des Klinikvorstands eingehalten werden.

Die Mitglieder des Vorstands verlassen sich nicht allein auf den Einfluss des OP-Koordinators, sondern setzen auch auf finanzielle Anreize, um die Abteilungen zur effizienteren Nutzung von Ressourcen und zur Nutzung von eMed zu bewegen: Auch nach der Reorganisation bleibt die Budgethoheit bei den chirurgischen Abteilungen. Die Abteilungsleiter*innen handeln mit dem Vorstand der Sanusklinik das Jahresbudget ihrer Abteilung aus. Der Vorstand setzt die formalen

[2] Alle hier angegebenen Namen und Bezeichnungen aus der Fallstudie sind Pseudonyme.

Regeln fest, die für die Verteilung des Budgets gelten, und trifft die Entscheidung über die Budgetverteilung im Einklang mit diesen Regeln. Das Budget der Abteilungen wird an die Einnahmen geknüpft, die diese für das Krankenhaus generieren.[3] Auch die Kosten, die die Arbeit in einer Abteilung nachweislich verursacht, werden berücksichtigt.[4] Grundlage für die Berechnung der Einnahmen und Kosten sind die Daten aus eMed, die von der Controllingabteilung überprüft und (unter anderem) für die Verwendung in den Budgetverhandlungen aufbereitet werden.

4.2 OP-Planung und -Dokumentation mit eMed

Das Modul eMed OP für die Planung und Dokumentation der Prozesse im OP-Bereich wird 2009 in einer einzigen Abteilung, der allgemeinen Chirurgie, eingeführt. Diese fungiert als Pilotabteilung. In Vorbereitung auf die Pilotphase findet der größte Teil des Customizing des OP-Moduls in der allgemeinen Chirurgie statt. Chirurgie und Pflege werden im Customizingteam durch Mitglieder der allgemeinen Chirurgie und der ihr zugeordneten Abteilung der OP-Pflege repräsentiert. Bei den Anpassungen des Standardsystems in Bezug auf die OP-Planung und –Dokumentation orientieren sich die am Customizing beteiligten daher stark an den Arbeitsabläufen, die in der allgemeinen Chirurgie bereits vor der Einführung von eMed etabliert waren. Die Details dieser Abläufe werden in 4.2.2 beschrieben.

Die flächendeckende Einführung von eMed OP in der Chirurgie dauert bis 2013. Einzelne Einführungsprozesse beginnen damit, dass die Verantwortliche für eMed OP aus dem IT-Service Kontakt mit der Person aufnimmt, die für die operative Leitung der Abteilung zuständig ist. In der Regel ist dies die leitende Oberärzt*in. Diese entscheidet dann, wer sich auf Seiten der Abteilung wie an der Einführung beteiligt und ob die Abteilungsleiter*in in den Einführungsprozess einbezogen werden soll.

[3] Der Vorstand nutzt das Budget auch dafür, die Abteilungsleiter*innen zur Einhaltung anderer Regeln der Organisation zu motivieren. So werden Abteilungen, die ihre OP-Berichte nicht innerhalb der vom Vorstand vorgegebenen Frist von sechs Wochen vervollständigen, Anteile der Einnahmen aus den betreffenden Operationen abgezogen.

[4] Weicht die Liegezeit einzelner Patient*innen, also die Zeit, über die diese für eine bestimmte Behandlung stationär betreut werden, zum Beispiel deutlich von der durchschnittlichen Liegezeit für diese Behandlung ab, so bezahlen die Kostenträger*innen nicht die gesamte Fallpauschale, sondern ziehen einen Betrag ab. Diese Abzüge werden an die Abteilung weitergereicht, in der die Patient*in stationär aufgenommen war.

*Fr. Senger (IT-Service): Die Chefs haben sich sehr unterschiedlich involviert. Es gibt sehr konservative Chefs, die das überhaupt nicht interessiert, die wollen nur, dass das funktioniert und machen, sprechen nur über ihren Oberarzt sozusagen und es gibt aber auch Chefärzte, die da sehr wohl dran Anteil nehmen. […] normalerweise reden die [Chefärzt*innen, DA] mit irgendwem im [IT-Service] nicht unmittelbar.*

Bevor das Modul in einer neuen Abteilung eingeführt wird, können Repräsentant*innen der Abteilung eigene Anpassungen an die Software mit den für die Einführung Verantwortlichen im IT-Service aushandeln. Mögliche Anpassungen sind jedoch von vorneherein begrenzt, erstens durch die Standardsoftware selbst, zweitens durch die im Customizing unter den Beteiligten ausgehandelten allgemeinen Rahmenbedingungen des Softwareeinsatzes und drittens durch die Parameter, die durch formale Organisationsregeln als verbindlich für diesen Einsatz festgesetzt worden sind. Die Abteilungen machen unterschiedlich stark von dieser Möglichkeit Gebrauch; im Ergebnis unterscheiden sich die Varianten vor allem in Details der Benutzer*innenoberfläche sowie darin, welche Angaben zum Eingriff bei der Anmeldung von Operationen verpflichtend sind. Wer zum Beispiel eine Operation in der Augenchirurgie anmelden will, muss die zu operierende Körperseite zwingend mit angeben. In der Neurochirurgie hingegen sind Angaben zur Körperseite optional. Auch die Berechtigungen zum Umgang mit dem System sind sehr unterschiedlich verteilt. Die Rechtevergabe schwankt zwischen restriktiv (nur wenige ausgewählte Mitglieder der Abteilung und die Pflegeleitung können im System planen) und permissiv (alle Chirurg*innen und Sekretär*innen der Abteilung und die zugeordneten Pflegekräfte haben Planungsrechte).

4.2.1 Das allgemeine Handlungsproblem in der OP-Planung

Das allgemeine Handlungsproblem der OP-Planung ist die Zuordnung von Patient*innen mit Operationsbedarf zu Operationszeiten und Ressourcen, die für die Operation nötig sind. Zu den Ressourcen zählen qualifiziertes Personal, Operationssäle, für den Eingriff notwendige Arbeitsmittel wie Röntgengeräte, Spezialbesteck oder Lagerungsvorrichtungen sowie Verbrauchsmaterialien wie Blutkonserven, Medikamente, Implantate und ähnliches. Bei der OP-Planung handelt es sich also auf den ersten Blick vor allem um ein Koordinationsproblem: Der OP-Plan für einen Tag soll sicherstellen, dass einerseits alle Patient*innen, die operiert werden müssen, möglichst adäquat behandelt werden. Der korrekte Eingriff muss dazu von Mediziner*innen und Pflegekräften mit der passenden

Qualifikation durchgeführt werden. Notwendige technische Geräte und Materialien müssen zum richtigen Zeitpunkt am richtigen Ort zur Verfügung stehen. Zur adäquaten Behandlung gehört auch die Vermeidung unnötiger Wartezeiten vor, während und nach der Operation sowie die den Bedürfnissen der Patient*in entsprechende Nachsorge, also zum Beispiel eine Unterbringung auf der Intensivstation im Anschluss, falls dies medizinisch geboten ist. Andererseits soll durch die OP-Planung auch sichergestellt werden, dass die Kapazitäten des Krankenhauses optimal ausgelastet werden. Dies bedeutet, dass der OP-Plan so gestaltet werden soll, dass alle Operationssäle während des Tages kontinuierlich gefüllt sind und alle Mitglieder des OP-Bereichs ihre Arbeitszeit produktiv im Sinne ihrer jeweiligen Aufgabenbeschreibung nutzen können.

In allen Krankenhäusern, in denen regelmäßig Operationen durchgeführt werden, findet irgendeine Art von OP-Planung statt. Wie diese Aufgabe konkret ausgestaltet ist, unterscheidet sich aber von Krankenhaus zu Krankenhaus. In der Sanusklinik variieren Arbeitsschritte, Verantwortliche, Vorgaben, zeitliche Zusammenhänge, Verbindlichkeit der Ergebnisse und andere Spezifika des Planungsprozesses bereits zwischen den Abteilungen stark. Dennoch lassen sich allgemeine Aussagen zur OP-Planung treffen, die unabhängig von den konkreten Praktiken überall dort gelten, wo Operationen zum Alltag gehören und zumindest einige der Ressourcen begrenzt sind, die zu ihrer Durchführung notwendig sind:

Damit eine Operation auf den OP-Plan gesetzt werden kann, muss zuerst bei einer Patient*in konkreter OP-Bedarf festgestellt werden. Dazu ist eine Untersuchung durch eine Chirurg*in nötig. Konkreter OP-Bedarf liegt vor, wenn feststeht, dass die Patient*in in einem definierten Zeitraum innerhalb der nächsten Tage operiert werden soll. Bei Patient*innen, deren Operationen Wochen oder Monate im Voraus geplant wurden, wird erst dann von konkretem OP-Bedarf gesprochen, wenn sie im Krankenhaus aufgenommen und von einer Chirurg*in untersucht worden sind. Jeden Tag wird eine Gruppe Patient*innen mit konkretem OP-Bedarf ausgewählt, die am Folgetag operiert werden soll.[5] Es wird festgelegt, in welcher Reihenfolge die Operationen stattfinden sollen, was im Detail zu tun und wer dafür verantwortlich ist. Der OP-Plan des Folgetags mit mindestens diesen Informationen (Reihenfolge, Patient*innen, genauer Eingriff, Verantwortliche) wird erstellt und auf irgendeine Art in der Gruppe der Beteiligten bekannt gemacht.

[5] Der Planungszyklus kann abhängig vom Krankenhaus variieren, die genaue Zeitlichkeit (also ob täglich, jeden zweiten Tag etc. für den Folgetag oder für weiter entfernte Tage geplant wird) ändert jedoch nichts an den grundsätzlichen Konturen des Problems.

Die zentrale Unsicherheit des Handlungsproblems liegt in der Frage, welche Operationen anstehen werden. Aus den logistischen Anforderungen der OP-Planung ergeben sich weitere potentielle Unsicherheiten, die zu großen Teilen durch das Krankenhaus vorstrukturiert werden. Stellen- und Dienstpläne, Materiallager und Betriebszeiten für OP-Säle, Hierarchien und Sanktionen bilden den Rahmen, in dem sich die Unsicherheiten der OP-Planung entfalten.[6] Das Verhältnis zwischen dem Operationsaufkommen und den durch diese Rahmenbedingungen vorgegebenen grundsätzlichen Kapazitäten, mit der jede Planende arbeiten kann, macht die grundlegenden Unsicherheiten des Handlungsproblems zu organisatorischen Ungewissheitszonen, die in den konkreten Handlungssystemen jeweils leicht unterschiedlich aussehen können.

In der Sanusklinik bildet eine chirurgische Abteilung mit den ihr zugeordneten Pflege- und Anästhesieteams ein solches Handlungssystem, da die Spiele, die die Mitglieder dieser Teams und der Abteilung spielen, untereinander verkettet sind. Die Akteur*innen dieser drei Gruppen sind es, die die tägliche Planung direkt beeinflussen. Was in anderen Bereichen der Klinik geschieht (z. B. Transportdienst, Lager etc.) haben die Planenden bei der Planung nicht im Blick; diese Bereiche funktionieren nach eigenen Regeln – für die Planung ist nur wichtig, dass sie meistens so funktionieren wie erwartet.

Die dargestellten Grundelemente des Handlungsproblems der OP-Planung werden in den einzelnen Handlungssystemen der Sanusklinik in unterschiedlichen Praktiken und damit auf unterschiedliche Weise bearbeitet. In allen wird der OP-Plan des Folgetages jeden Morgen von (mindestens) den Oberärzt*innen im Dienst besprochen. Große Unterschiede zwischen den Abteilungen gibt es in drei Punkten: Erstens, welche Informationen der vorläufige Plan bereits enthält, zweitens, wie er genau zu Stande kommt und drittens, welche Funktion seine Besprechung hat, also ob sie allein der Information der Anwesenden dient oder Inhalte des Plans dort nur diskutiert oder auch (regelmäßig oder nur in Ausnahmefällen) verändert werden. Vor oder nach dieser Besprechung werden weitere Informationen in den Plan eingefügt, die für die Vorbereitung der Operationen wichtig sind. Dazu gehören zum Beispiel der zugeordnete OP-Saal, besondere Instrumente oder Materialien, die Art der Narkose, die Lagerung, also wie die Patient*in auf dem Operationstisch platziert werden soll, und die vorgesehene Nachsorge. Den fertigen Plan erhalten Anästhesie und OP-Pflege, in der Sanusklinik formell bis zum Mittag des Vortags. In beiden Abteilungen wird auf

[6] Eine ausführliche Analyse des Handlungsproblems, die logistische ebenso wie mikropolitische Aspekte, deren Interdependenzen und die Herausforderungen, die daraus für die Entwicklung maßgeschneiderter Software entstehen, deutlich macht, findet sich in Egger und Wagner (1993).

Basis der Informationen des Plans das Personal für den jeweiligen Tag eingeteilt. Eine Anästhesist*in besucht am Vortag die Patient*innen, um die Narkose zu besprechen, um festzulegen, welche Medikamente gegeben werden und um die schriftliche Einwilligung für den Eingriff einzuholen.[7] Die eingeteilten Pflegekräfte beschaffen direkt vor der OP die benötigten Materialien, bauen sie im Saal auf und informieren die Mediziner*innen, dass ihre Operation bald ansteht. Die Anästhesist*in bestellt die Patient*in beim Transportdienst, damit diese genau dann im OP ankommt, wenn es Zeit für die Vorbereitung ist. Im OP-Vorraum wird die Patient*in dann von Pflegekräften vorbereitet und von der Anästhesist*in in Narkose gelegt. Erst wenn sie schläft, wird sie in den OP-Saal gefahren. Aus Sicht der Chirurg*innen beginnt die Operation zu diesem Zeitpunkt (mit dem Schnitt) und endet, sobald die Wunde verschlossen ist (mit der Naht). Für die Planung müssen jedoch auch die Vor- und Nachbereitungszeiten (Ausleitung der Narkose, Überführung der Patient*in, Saalreinigung) mitbedacht werden.

Die in den Abteilungen der Sanusklinik eingesetzten Varianten von eMed unterscheiden sich in den zu Beginn von 4.2 dargestellten abteilungsspezifischen Details. Das Modell der OP-Planung in eMed bildet die allgemeinen Grundelemente des Handlungsproblems für alle einheitlich ab. Um bei der Analyse der Spiele die Modelle des Organisierens als technische Bedingungen berücksichtigen zu können, wird nun als erstes vorgestellt, wie der Prozess der OP-Planung in diesem Modell aussieht.

4.2.2 Modell der OP-Planung in eMed nach dem Customizing

Das Modell der OP-Planung, das beim Customizing für alle Abteilungen festgelegt worden ist, ist für diese Arbeit aus Interviews und dem Nutzerhandbuch der Klinik rekonstruiert worden. Der Prozess folgt in diesem Modell in weiten Teilen der Prozessvorlage aus der generischen Version von eMed und wird im Folgenden beschrieben (vgl. auch Abbildung 4.2).

[7] Dieser Vorgang wird Prämedikation genannt.

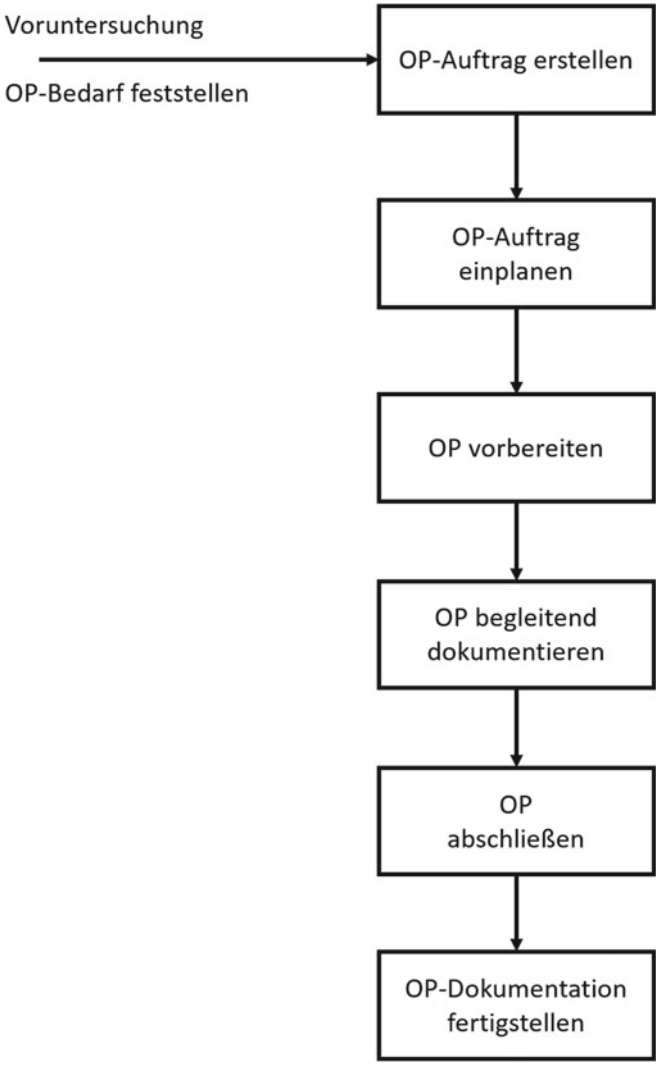

Abbildung 4.2 Prozess der OP-Planung mit eMed. (Eigene Darstellung)

Eine Nutzer*in (die Veranlasser*in) fordert eine Operation für eine Patient*in an, indem sie in eMed einen OP-Auftrag für die zuständige chirurgische Abteilung (zuständige Organisationseinheit) erstellt. In diesem Auftrag hält sie fest, welcher Eingriff aufgrund welcher Diagnose durchgeführt werden soll. Sie fügt alle Informationen hinzu, die sie über die Patient*in erhoben hat und die für die Operation relevant sein könnten, und speichert den Auftrag in der Software. Die Veranlasser*in kann zwar einen Termin für die Operation vorschlagen, die Entscheidung darüber, ob der Terminvorschlag angenommen wird, liegt jedoch bei der Planer*in. Die Software prüft, ob alle Pflichteingaben für einen OP-Auftrag vorhanden und alle Eingaben im richtigen Format sind. Ist dies der Fall, speichert die Software den Auftrag als bestätigt. Der Auftrag erscheint nun in der Auftragsübersicht der zuständigen Abteilung und kann dort von der Planer*in gesehen werden. Die Planer*in legt sowohl den Termin als auch die vermutliche OP-Dauer (die sogenannte Planzeit), den Saal und alle anderen Daten fest, die in der Abteilung als Pflichteingaben für den OP-Plan vorgegeben, im Auftrag aber noch nicht enthalten sind. Der so vervollständigte OP-Auftrag wird automatisch in den Plan eingefügt und repräsentiert die Operation nun im System.

Der OP-Plan kann in eMed auf unterschiedliche Arten dargestellt werden: Die Planer*in kann die aktuell geplante Belegung der OP-Säle und alle angeforderten Operationen des Tages gleichzeitig einsehen und OP-Aufträge der Reihe nach Sälen zuordnen. Diese Planungsansicht ist für die meisten Mitglieder der Chirurgie nicht einsehbar, sie dient allein der Erstellung und Bearbeitung des Plans (Abbildung 4.3).

In der Plantafel, einer Kalenderansicht des ausgewählten Tages, werden die OP-Säle als Spalten dargestellt, die geplanten Operationen als Blöcke. Die Ausdehnung der Blöcke korrespondiert mit der bei der Planung angegebenen Operationszeit (Abbildung 4.4). Die OP-Säle, die in den Gebäuden der Sanusklinik nebeneinander liegen (vgl. 4.1.2), werden auch auf der Plantafel gemeinsam dargestellt; das Team, das am Customizing beteiligt war, hat sich gegen eine Gesamtansicht entschieden, die alle 40 Säle der Klinik auf einmal zeigt, da auf einem regulären Bildschirm die Pläne für maximal acht Säle gleichzeitig repräsentiert werden können. Die Plantafel stellt die Auslastung aller OP-Säle im Tagesverlauf grafisch dar. Die Planer*in kann die Blöcke, die Operationen repräsentieren, auf der Zeitleiste oder zwischen den Spalten, die für Säle stehen, verschieben und die Plantafel somit direkt zur Planung nutzen. Laut der Dokumentation der Standardversion von eMed ist die Plantafel vor allem für die Nutzung durch die Person vorgesehen, die für die gleichmäßige Auslastung der OP-Ressourcen verantwortlich ist. In der Sanusklinik wäre dies die OP-Koordinator*in.

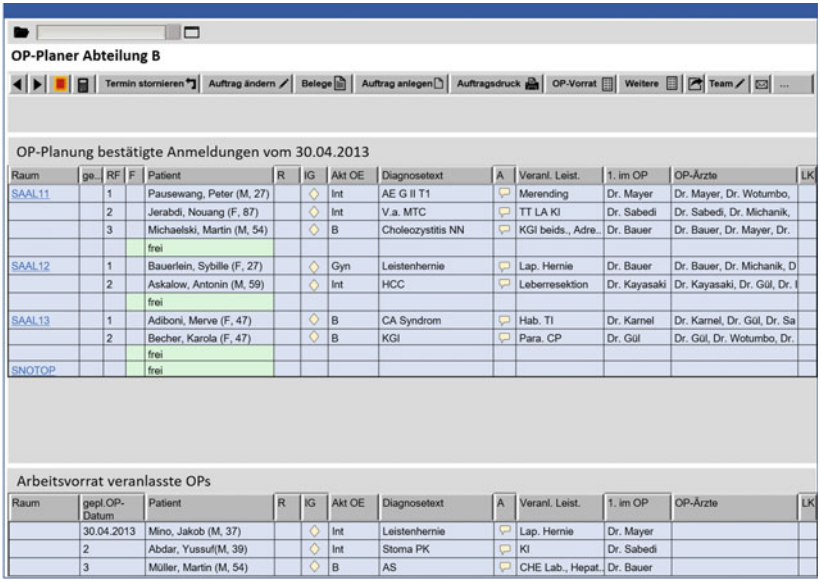

Abbildung 4.3 Planungsansicht mit angeforderten Operationen. (Eigene Darstellung angelehnt an Benutzerhandbuch)

Die Farbe der Blöcke zeigt an, ob eine Operation noch aussteht, bereits begonnen wurde oder schon abgeschlossen ist. Wenn laufende Operationen die geplante Zeit überschreiten, kann das zwar aus der Farbe der Blöcke gefolgert werden, die Ausdehnung der Blöcke in der Kalenderansicht verändert sich jedoch nicht. Die Plantafel ist für alle Mitglieder des OP-Bereichs sichtbar; bearbeiten kann diese Ansicht jedoch nur, wem die Abteilung Planungsrechte in eMed eingeräumt hat.

Ähnliches gilt auch für das laufende OP-Programm, wie die Listenansicht des OP-Plans in eMed bezeichnet wird. Hier sind die Operationen entsprechend der geplanten Reihenfolge untereinander aufgelistet (Abbildung 4.5). Im laufenden OP-Programm zeigen zwei Spalten an, in welchem Stadium die Operation gerade ist. In allen drei Ansichten werden die Operationen durch eine kleine Auswahl der Daten aus dem OP-Auftrag symbolisiert, die jede Abteilung selbst festgelegt hat.

Das laufende OP-Programm soll Nutzer*innen ermöglichen, von der Übersicht über die anstehenden Operationen aus die Ansicht zu öffnen, in der sich die

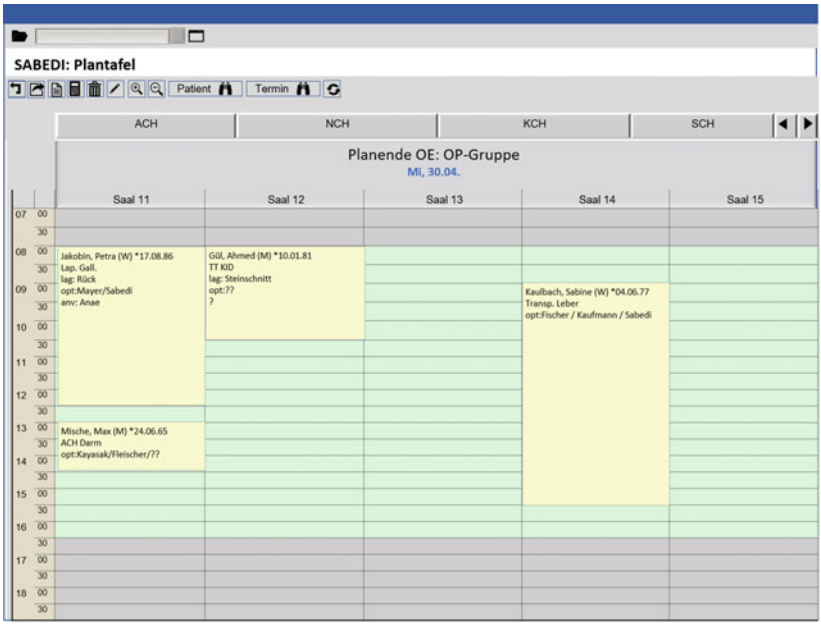

Abbildung 4.4 Plantafel. (Eigene Darstellung angelehnt an Benutzerhandbuch)

OP-begleitende Dokumentation einer Operation erstellen lässt. In der Sanusklinik wird das laufende OP-Programm so genutzt: Zu Beginn der OP-Vorbereitung öffnet die dafür verantwortliche Pflegekraft den Auftrag für die anstehende OP, indem sie in der Liste auf die Zeile klickt, in der diese Operation repräsentiert ist. Darüber ruft sie ein Protokollformular auf, in dem sie während der Operation alle vorgeschriebenen Zwischenschritte dokumentiert. Die Operation wird als begonnen markiert, was in allen Ansichten des Plans sichtbar ist. Nach Abschluss der Operation benutzen auch Chirurg*innen und Anästhesist*innen die Spalte mit dem OP-Auftrag als Einstieg, um die jeweiligen medizinischen Dokumentationen in dafür vorgesehene Formulare einzutragen. Alle Dokumente werden Teil der Krankenakte der Patient*in.

Das laufende OP-Programm ist laut der Dokumentation von eMed nicht für die Planung entwickelt worden, sondern soll eigentlich einen bestehenden Plan repräsentieren. Hier können jedoch auch Veränderungen am Plan vorgenommen

werden, soweit sie die angezeigten Parameter betreffen (z. B. kann der zugeordnete OP-Saal nach dem Öffnen des OP-Auftrags verändert werden, solange die Operation noch nicht begonnen hat). Auch Daten der geplanten Operation, die nicht in der Ansicht angezeigt werden, können von der Ansicht des laufenden OP-Programms aus geändert werden. Um dies zu tun, müssen Nutzer*innen jedoch immer erst den OP-Auftrag öffnen, indem sie auf die Zeile in der Ansicht klicken, in der dieser Auftrag repräsentiert ist. Darüber hinaus brauchen Nutzer*innen für solche Änderungen die passenden Berechtigungen. In den meisten Abteilungen besitzen alle Mitarbeiter*innen die Berechtigung, das laufende OP-Programm zu sehen, aber nur wenige die Berechtigung, OP-Aufträge zu verändern.

Abbildung 4.5 Laufendes OP-Programm. (Eigene Darstellung angelehnt an Benutzerhandbuch)

Der OP-Plan ist kein eigenständiges Element von eMed, sondern nur eine Ansicht, in der eine Sammlung einzelner Elementen auf eine bestimmte Art und Weise angezeigt wird. Das, was im Umgang mit eMed als OP-Plan behandelt wird, besteht in der Software aus mehreren auf unterschiedliche Arten dargestellten Listen von OP-Aufträgen. Der OP-Auftrag erweist sich als das erste relevante Softwareelement, das in allen mit der OP-Planung verbundenen Praktiken bedeutsam wird. Von der Beauftragung bis zur Abrechnung der Operation ändert sich zwar der Status, den der OP-Auftrag in eMed hat, der Auftrag bleibt aber an sich als Element erhalten. Dieses Element *OP-Auftrag* ist die im Customizing erstellte Variante der Datenstruktur *Klinischer Auftrag* im OP-Modul der Sanusklinik. Die Datenstruktur *Klinischer Auftrag* wiederum ist in eMed definiert, also das Produkt des Softwareentwicklungsprozesses, der außerhalb der Sanusklinik (und ohne Beteiligung ihrer Mitglieder) stattgefunden hat. Über klinische Aufträge wird die Behandlung von Patient*innen im Krankenhaus angestoßen, geplant und dokumentiert. Insbesondere werden die Daten aus den Voruntersuchungen der Patient*innen, die Plandaten der Operation und die Daten ihrer Dokumentation zusammen im klinischen Auftrag gespeichert und sind über diesen für Nutzer*innen gemeinsam verfügbar. Der klinische Auftrag fungiert

damit als Verbindungsglied zwischen den verschiedenen Arbeitsprozessen rund um Operationen.

Bei der Untersuchung des Umgangs mit eMed in der OP-Planung zeigt sich, dass die technische Verbindung zwischen OP-Planung und OP-Dokumentation die zentrale Ursache für Veränderungen der Praxis mit eMed im Vergleich zur Praxis vor der Einführung des KIS ist. Während OP-Planung und OP-Dokumentation vorher technisch und organisatorisch nur lose gekoppelt stattfanden, koppelt der OP-Auftrag aus Sicht der Nutzer*innen im OP-Bereich beide Aufgaben nun eng aneinander. Diese enge Kopplung ist eine Folge des Zwecks, für den der OP-Auftrag im KIS vorgesehen ist: Durch den OP-Auftrag wird dafür gesorgt, dass alle Daten, die für die Planung, Durchführung, Dokumentation und Abrechnung einer Operation im Krankenhaus als relevant erachtet werden, nach der ersten Eingabe im KIS gespeichert und an alle weitergereicht werden, die mit einem dieser Vorgänge zu tun haben.

Die Operation selbst wird in der Software als eine Menge medizinischer *Leistungen* modelliert, ebenso wie alle begleitenden medizinischen Maßnahmen, wie Voruntersuchungen, Narkose, Pflegetätigkeiten und andere. *Leistungen* repräsentieren einzelne Tätigkeiten der medizinischen Akteur*innen und stellen damit ein Modell der Aufgaben dar, die diese Akteur*innen erfüllen, wenn sie Patient*innen behandeln. Diese Aufgaben sind in eMed doppelt modelliert: Medizinische Tätigkeiten werden in eMed gleichzeitig als *Prozeduren* und *Leistungen* abgebildet. Während Prozeduren der Dokumentation dienen, sind *Leistungen* für die Abrechnung gedacht. Über zwei unterschiedliche Softwareelemente werden in diesem Fall zwei unterschiedliche Sichtweisen auf die gleiche Tätigkeit repräsentiert. Beide Elemente werden zusammen generiert, d. h. stößt eine Nutzer*in zum Beispiel bei der Beauftragung einer Operation die Erstellung einer neuen *Leistung* an, so erzeugt die Software automatisch die dazugehörige *Prozedur.* Im weiteren Verlauf der Arbeit werde ich auf diese Doppelstruktur nur über das Element der *Leistungen* Bezug nehmen, denn die Sichtweise, die in den *Leistungen* repräsentiert ist, erwies sich bei der Untersuchung als besonders relevant für das Verständnis des beobachteten Geschehens im Fall. *Leistungen* werden nicht nur in Verbindung mit Operationen, sondern bei der gesamten Behandlung der Patient*innen erbracht. *Leistungen* und *Klinische Aufträge* stellen also unterschiedliche Modelle dar, die aber in der Software verknüpft sind.

Der *OP-Auftrag* ist in eMed an ein Element gebunden, das als *Fall* bezeichnet wird. Der *Fall* verknüpft alle Daten, die im Laufe des Behandlungsprozesses entstehen, miteinander und mit den anderen Daten, die im Krankenhaus über die Patient*in gespeichert sind (z. *B.* Adresse, Versicherungsnummer etc.). Durch den *Fall* sollen alle Aspekte des Aufenthalts einer Patient*in im Krankenhaus

von Beginn an für die Organisation nachvollziehbar werden. *Leistung, Fall* und *OP-Auftrag* sind die Softwareelemente, die zentral für die Praktiken in und um die OP-Planung in der Sanusklinik sind. Der *OP-Auftrag* ist eng an die eigentliche OP-Planung geknüpft. Durch Statuswechsel (von „beauftragt" zu „geplant", „laufend", „abgeschlossen") bildet er die unterschiedlichen Stadien der Operation im Krankenhaus ab. Der *Fall* integriert die OP-Planung in die Modelle der übrigen medizinischen und administrativen Arbeitsabläufe der Organisation: Er verbindet die verschiedenen klinischen Aufträge (z. B. einen OP-Auftrag, aber auch Aufträge für Voruntersuchungen, Röntgen etc.), die im Rahmen eines Krankenhausaufenthalts erfüllt werden, miteinander und mit Patient*innendaten, die für die Verwaltung des Aufenthalts gebraucht werden. Über die *Leistung* schließlich werden die finanziellen Aspekte der Behandlung im Krankenhaus mit den prozessualen und medizinischen Sichten verknüpft.

4.2.3 Die Praxis der OP-Planung in der Sanusklinik

Theoretisch entsprechen die Arbeitsschritte und Anforderungen der OP-Planung für die allermeisten Operationen[8] denen, die in 4.2.1 beschrieben werden: Auswahl einer Gruppe von Patient*innen mit konkretem OP-Bedarf, Festlegung der genauen Eingriffe, Zuordnung von Sälen und Personal, Festlegung der Reihenfolge der Operationen an einem Tag, Bekanntmachung des Plans. Dadurch erscheint die OP-Planung auf den ersten Blick als eine vorhersehbare Aufgabe. In der Praxis setzen jedoch die Patient*innen und ihre unberechenbaren Krankheitsverläufe der Vorhersehbarkeit Grenzen: Patient*innen erscheinen beispielsweise am Vortag der Operation nicht wie geplant zur Einweisung oder erscheinen unerwartet in kritischem Zustand. Der Zustand von Patient*innen verschlechtert sich über Nacht und sie können daher gar nicht oder aber müssen viel schneller als geplant operiert werden. Den Chirurg*innen bietet sich nach dem Schnitt ein

[8] Eine Ausnahme bilden Fälle, in denen Patient*innen ohne Aufklärung direkt nach der Einlieferung in die Klinik operiert werden müssen, wodurch das Vorgespräch mit der Anästhesist*in und die Einwilligung entfallen. In der Praxis stellen solche Situationen aber die Ausnahme dar, denn auch die meisten dringenden Operationen werden an Patient*innen durchgeführt, die vor dem Eingriff noch bei Bewusstsein sind und in der Regel schon auf einer der Stationen der Klinik liegen.

anderes Bild des Körperinneren, als die Diagnostik vermuten ließ. Kurz: Ausfälle, Notfälle[9] und Terminverschiebungen aus verschiedensten Gründen sind Teil der Routine der OP-Planung. Der Umgang mit ihnen gehört ebenso zu diesem allgemeinen Handlungsproblem wie die logistischen Anforderungen, die sich aus dem „ungestörten" Ablauf ergeben.

Solche kurzfristigen Veränderungen des OP-Plans werden von der Planer*in mit der zuständigen Leiter*in der zugeordneten Teams der Pflege und Anästhesie abgesprochen. Wenn die Veränderung beschlossen ist, werden alle telefonisch informiert, die an der Operation beteiligt sind. Anders als bei den am Vortag geplanten Operationen wird die Änderung den übrigen Chirurg*innen weder aktiv mitgeteilt, noch haben diese Einfluss auf die Entscheidung. Sie erfahren davon nur, wenn sie sich den OP-Plan des aktuellen Tages in eMed ansehen und ihnen die Veränderung auffällt. Kleinere Verschiebungen im Tagesverlauf können aufgefangen werden: Die Leiter*in der OP-Pflege behält das Geschehen in den Sälen im Tagesverlauf im Blick und sorgt dafür, dass die für eine Operation eingeteilten Pflegekräfte mit der Vorbereitung beginnen, sobald der OP-Saal bereit ist, und alle anderen Beteiligten telefonisch informiert werden, wenn ihr Einsatz ansteht. Kurzfristig neu in den Plan aufgenommene Operationen führen aber dazu, dass andere Operationen auf einen Folgetag verschoben werden oder vorerst ganz aus der Planung fallen. Ausfälle gehen jeweils in den Planungsprozess eines Folgetags ein, ein neuer Termin wird also nicht automatisch festgelegt.

Dass die OP-Planung grundsätzlich ein Prozess voller Unsicherheit ist, ist bei allen Beschäftigten im OP-Bereich akzeptiert, denn *„wir sind eben nicht bei der Post und auch nicht im Büro"* (OP-Leitung allgemeine Chirurgie). Wie damit umgegangen werden sollte und welche Ursachen im Einzelfall verantwortlich sind, ist Gegenstand regelmäßig wiederkehrender Konflikte innerhalb der und zwischen den verschiedenen Berufsgruppen. Diese drehen sich vor allem um die Frage, wer für Verschiebungen im Plan und deren Folgen wie das verlängerte

[9] Für die Analyse des Planungsprozesses gelten als Notfälle hier die Operationen, die kurzfristig, d. h. am gleichen Tag oder am Folgetag, in einen bereits bestehenden OP-Plan eingefügt werden müssen. Was ein Notfall ist, lässt sich nicht allgemein festlegen, sondern hängt trotz bestehender medizinischer Leitlinien endgültig von der Entscheidung der beurteilenden Chirurg*in ab. Die Chance, etwas als Notfall zu definieren, gilt als eine wichtige professionelle Machtressource der Medizin (Freidson et al. 1979, S. 101 f.). Die auch strategische Nutzung der Notfalldefinition wurde bei der Untersuchung als potentiell für die Fragestellung relevantes Element erkannt. Im Laufe des Forschungsprozesses zeigte sich jedoch, dass rein strategische Notfalldefinitionen nicht dazu beitragen, die Rolle von eMed für die Praktiken der OP-Planung in den untersuchten Abteilungen der Sanusklinik zu verstehen. Der Notfall wird daher als ein Aspekt der grundlegenden Problematik der OP-Planung mit zu den durch Patient*innen verursachten Unsicherheiten gezählt.

Leiden von Patient*innen, deren Operationen kurzfristig verschoben werden, oder unerwartete Überstunden, die den Mitarbeiter*innen abverlangt werden, verantwortlich ist und ob diese hätten vermieden werden können. Klar ist, dass nicht alle Unsicherheiten der Planung von Patient*innen ausgehen: eine Pflegekraft, die für den Weg zu einem entfernten Saal 60 statt wie erwartet 20 Minuten braucht, Anästhesist*innen, die Anfragen auf Unterstützung beim Schichtwechsel nicht weitergeben, Chirurg*innen, die erst kurz vor dem Schnitt bekannt geben, dass der gleich beginnende Eingriff deutlich komplexer wird als im Plan angegeben – wie sehr der Tagesablauf im OP-Bereich mit dem übereinstimmt, was im OP-Plan festgelegt worden ist, hängt davon ab, wie gut die Planenden solche Störungen verhindern oder ausgleichen können. Die Analyse der Spiele in Abschnitt 4.4 wird zeigen, dass der Umgang mit eMed darauf großen Einfluss hat.

Das medizinische Personal im OP-Bereich schätzt es, dass eMed einen direkten Zugriff auf Patient*innendaten ermöglicht, arbeitet aber insgesamt ungern mit der Software. Sie wird als langsam und umständlich wahrgenommen und ist nur in den Sekretariaten überwiegend beliebt. Chirurg*innen, Pfleger*innen und Anästhesist*innen empfinden die Benutzer*innenoberflächen als unübersichtlich und die Prozessvorlagen als unnötig komplex; viele der Informationen, die die Software ihnen präsentiert oder von ihnen verlangt, erscheinen ihnen überflüssig für die Erfüllung ihrer Aufgaben. Die Reaktionen der Software auf Eingaben finden sie oft nicht nachvollziehbar.

> *Dr. Schmidt (Oberarzt Anästhesie): Für mich ist das ein Programm, was ich als faschistoid empfinde. […] Das Hauptproblem ist ja, dass es so viele Informationen sind, die ich gar nicht brauche. Das ist gar nicht darauf beschränkt, was für mich nötig ist.*

Die Verpflichtung, bei der Planung und Dokumentation eMed einzusetzen, wird vor allem als Zusatzbelastung empfunden, die die Beschäftigten von ihrer eigentlichen Arbeit an den Patient*innen ablenkt. Chirurg*innen und Pflegekräfte haben in der Sanusklinik jedoch keine alternativen Möglichkeiten, um Pläne und medizinische Dokumente im Alltag mit all den Akteur*innen zu teilen, die darüber informiert sein müssen, wenn die Behandlungsabläufe funktionieren sollen. Sie können die Regeln der Sanusklinik, die eine Nutzung von eMed vorschreiben, nicht umgehen, wenn sie ihr Handlungsproblem lösen wollen. Keine andere Software wird von allen Akteur*innen genutzt, die koordiniert werden müssen, damit Operationen durchgeführt werden können. Bei der Koordination ganz auf Software zu verzichten ist erst recht nicht vorstellbar, denn die Akteur*innen sind räumlich über den Campus der Sanusklinik verteilt und müssen eine Menge an Detailinformationen austauschen, um sich zu koordinieren. Der Umgang mit der ungeliebten Software stellt immer noch die beste Möglichkeit dar, um

das Handlungsproblem zu lösen. Vor allem die Planungsdaten und Teile der operationsbegleitenden Dokumentation sind jedoch extrem unzuverlässig; jede Abteilung hat eigene informelle Regeln dafür entwickelt, wie die Interpretationsspielräume, die den Nutzer*innen bei der Dateneingabe bleiben, auszulegen sind und welche Regelverstöße (v. a. die Eingabe formal korrekter, aber inhaltlich fehlerhafter Daten) wie sanktioniert werden. Einige dieser informellen Regeln werden bei der Analyse der Spiele in Abschnitt 4.4 vorgestellt.

Die Stelle der OP-Koordinator*in ist im Rahmen der Reorganisation entstanden, in der die formale Stellenstruktur des OP-Bereichs an das Modell der Organisation in der Standardversion von eMed OP angepasst worden ist. Der aktuelle OP-Koordinator nutzt allerdings nur wenige der für seine Stelle vorgesehenen Funktionen der Software. Eines der Hauptargumente der Softwareherstellerin für eMed OP ist, dass dieses Produkt sich dazu eigne, einen OP-Bereich zentral zu überblicken und zu steuern. Diese Behauptung verweist er in den Bereich der Phantasie. Statt eMed OP nutzt der OP-Koordinator den Plan der Anästhesist*innen, den diese in einer eigenen Software erstellen und selbstständig pflegen, um sich einen Überblick über das zu verschaffen, was im OP-Bereich vor sich geht.

Die Abteilung Anästhesie hat die formale Erlaubnis, eine andere Software als eMed für ihre internen Planung einzusetzen, weil die Standardversion von eMed keine Möglichkeit vorsieht, die OP-Pläne, die nur Operationen abbilden, mit den geplanten diagnostischen Untersuchungen, die ebenfalls von der Anästhesie betreut werden, in einer Gesamtübersicht zu vereinen. Die Blockleiter*innen, also die Leiter*innen der anästhesistischen Einheiten, die den einzelnen chirurgischen Abteilungen zugeordnet sind, benutzen nur diese Software, um den Überblick über das Geschehen im OP-Bereich und die Auslastung ihrer Mitarbeiter*innen zu behalten. Die Daten aus den OP-Aufträgen in eMed und kurzfristige Änderungen, von denen sie auf andere Weise erfahren, übertragen sie per Hand in das eigene System. Diese Übertragung verursacht erheblichen Arbeitsaufwand. Die Anästhesist*innen nehmen diesen jedoch in Kauf; die Software, mit der die Anästhesist*innen ihre Pläne erstellen, bietet im Gegensatz zu eMed eine Gesamtübersicht aller OP-Säle an. Diese zeigt zwar weniger Details zu den laufenden Operationen an, stellt aber dafür aktuelle Veränderungen in der Planung grafisch dar. Da die Anästhesist*innen in ihrer Software selbst eintragen, wann Operationen begonnen und beendet werden, ist die Übersicht in der Planungssoftware der Anästhesist*innen unabhängig von den verschiedenen Planungs- und Dokumentationspraktiken der chirurgischen Abteilungen.

Die Ausführungen der Abschnitte 4.1 und 4.2 geben die Erkenntnisse aus der ersten Analysephase wieder (vgl. Abschnitt 3.2.1 zur Beschreibung des Vorgehens). Es zeigt sich, dass die OP-Dokumentation seit der Einführung von eMed eng mit dem Handlungsproblem der OP-Planung verbunden ist. Das Modell der OP-Planung aus der Standardversion von eMed OP ist zu großen Teilen übernommen worden, wobei sich OP-Auftrag, Leistung und Fall als zentrale Elemente herauskristallisieren. Widerstände zeichnen sich vor allem daran ab, dass Akteur*innen im OP-Bereich Daten, die die OP-Planung vereinfachen könnten oder für eine vollständige Planung sogar unabdingbar wären, falsch oder gar nicht eingegeben. Sie werden aber auch daran erkennbar, dass die Mitarbeiter*innen der Anästhesie großen Aufwand aufbringen, um nicht mit eMed planen zu müssen, und dass der OP-Koordinator es ablehnt, die Überblicksfunktion der Software zu nutzen. Die zentralen Elemente und beobachteten Widerstände bilden den Ausgangspunkt für die Rekonstruktion der Modelle des Organisierens.

4.3 Modelle des Organisierens im Krankenhausinformationssystem

Über Modelle des Organisierens wird versucht, die Arbeitsprozesse in Organisationen auf einen einheitlichen Zweck auszurichten. Die Widerstände im Umgang mit der Software, die diese Modelle enthält, zeigen an, wenn Akteur*innen ihre Praxis nicht auf diesen Zweck ausrichten wollen oder können. Letzteres kann eine Folge von Widersprüchen zwischen verschiedenen Modellen des Organisierens in der Software sein oder aus Widersprüchen zwischen diesen Modellen und formalen Regeln der Organisation entstehen, die ebenfalls Versuche darstellen, Kontrolle über den Umgang der Organisationsmitglieder mit der Software auszuüben.

Durch Analyse der Online-Softwaredokumentation von eMed lassen sich einige der grundlegenden Zusammenhänge zwischen den Datenstrukturen und Prozessvorlagen rund um die zentralen Elemente für die OP-Planung und OP-Dokumentation rekonstruieren. Die interne Struktur des Krankenhaussystems lässt sich auf drei technische Ebenen abstrahieren. Diese sind (s. auch Abbildung 4.6):

(1) Der generische Softwarekern der Standardsoftware, die die Firma SAP für Organisationen produziert. Dieser wird als *ERP-Kern* bezeichnet. (ERP steht hier als Abkürzung für Enterprise Resource Planning, eine häufig genutzte Bezeichnung für standardisierte Organisationssoftware).

(2) Die Erweiterungsmodule für den Gesundheitssektor, die ebenfalls von der Firma SAP entwickelt werden. Diese sind unter der Abkürzung *IS-H* bekannt.
(3) Eine Reihe an Ergänzungsmodulen für den Einsatz im Krankenhaus, die zusammen *i.s.h.med* genannt werden. Diese krankenhausspezifische Ergänzung ist kein Produkt von SAP, sondern wird von einem anderen Unternehmen angeboten. Erst durch i.s.h. med wird die gesamte Software zum Krankenhausinformationssystem.

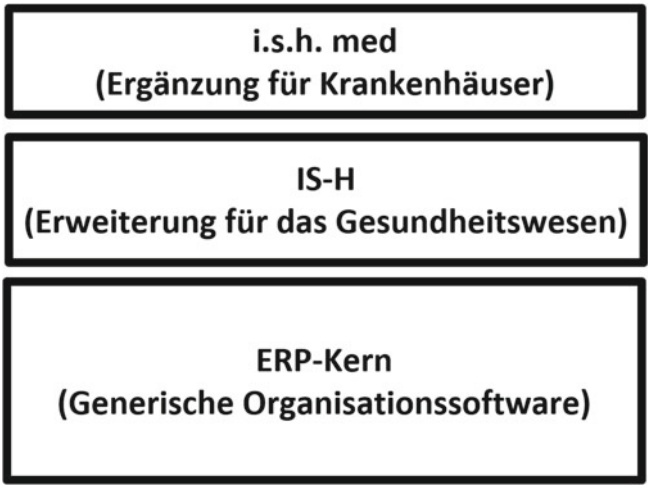

Abbildung 4.6 Technische Ebenen des Krankenhausinformationssystems. (Eigene Darstellung)

Es zeigt sich, dass die Art, wie Unsicherheiten in den Modellen der Elemente OP-Auftrag, Leistung und Fall reduziert wird, innerhalb jeder der drei technischen Ebenen von eMed einer konsistenten Logik folgt. Diese Logiken werden in den folgenden drei Abschnitten beschrieben. Die Beschreibung wird auch zeigen, dass die Logiken der drei technischen Ebenen nicht vollständig miteinander vereinbar sind. Viele der Datenstrukturen und Prozessvorlagen, die beim Umgang mit der Software genutzt werden, lassen sich nicht eindeutig einer Ebene zuordnen, sondern setzen sich aus Elementen zusammen, die ihren Ursprung auf verschiedenen Ebenen haben. Dadurch entstehen Abhängigkeiten zwischen den Elementen, die

die Ebenen mit ihren in sich konsistenten Logiken überschreiten. Die Zusammenhänge zwischen den Elementen folgen daher in einigen Arbeitsprozessen mehr der einen, in einigen mehr der anderen Logik. Modelle des Organisierens lassen sich aus den Ausschnitten der Dokumentation nicht direkt rekonstruieren. Zu diesem Zweck wurden Informationen über die sozialen Bedingungen, unter denen die Software entwickelt worden ist, zur Analyse hinzugezogen. Die technischen Ebenen können aus dieser „biographischen" Perspektive (vgl. 2.2.3.2) leicht als Produkt verschiedener sozialer Systeme identifiziert werden. Diese sozialen Systeme sind relativ stabil, d. h. es handelt sich um dauerhafte Beziehungs- und Interaktionszusammenhänge, nicht nur um isolierte Softwareentwicklungsprojekte in einzelnen Organisationen. Aus ihrer Historie und Entwicklung lässt sich in Umrissen erkennen, welche Art der Kontrolle mit ihren Produkten über Organisationen ausgeübt werden soll.

4.3.1 ERP-Kern: Die Ebene der generischen Organisationssoftware

Enterprise Resource Planning-Systeme (ERP-Systeme) sind darauf ausgelegt, Informationen über die unterschiedlichen Arbeitsprozesse in einem Unternehmen zu integrieren (Davenport 1998). Die Integration erfolgt einerseits darüber, dass die Software Daten über die Arbeitsprozesse in einheitlicher Form verwaltet und in einer gemeinsamen Datenbank oder einer vergleichbaren Struktur so ablegt, dass eine einheitliche Datenbasis entsteht, auf der beim Umgang mit der Software in den Arbeitsprozessen zurückgegriffen wird. Andererseits werden die Arbeitsprozesse über Prozessvorlagen integriert, die vorgeben, welche Arbeitsschritte aufeinander folgen sollen und welche Daten in jedem Schritt in die gemeinsame Datenbasis eingegeben werden müssen und können. Eine gemeinsame Benutzer*innenoberfläche sorgt für ein einheitliches Erscheinungsbild, wodurch Nutzer*innen bei allen Arbeitsprozessen, die durch die Software unterstützt sind, auf ähnliche Art und Weise mit der Software umgehen können.[10]

[10] Diese Beschreibung entspricht einer der technischen Grundstrukturen der Software, der Architektur nach dem Drei-Schichten-Modell, demzufolge Datenhaltungs-, Anwendungs- und Präsentationsschicht konzeptionell und technisch so getrennt sind, dass jede Schicht nur mit den beiden jeweils nächsten über wohldefinierte Schnittstellen Informationen austauschen kann (Ludewig und Lichter 2013, S. 431). Die Anwendungsschicht wird im untersuchten ERP-System als „business logic" bezeichnet (Mormann 2016, S. 30) und enthält nicht nur Modelle von Geschäftsprozessen in Form von Prozessvorlagen, sondern auch Modelle anderer Organisationsaspekte (a.a.O., S. 54 ff.). Da den Geschäftsprozessen jedoch für die

Das Ergebnis jeder Aktion in der Software, bei der Daten verändert werden, wird unmittelbar gespeichert und ist über die gemeinsame Datenbasis für alle Elemente der Software verfügbar (Hohlmann 2007). Damit enthält die gemeinsame Datenbasis – so die Theorie – ein genaues Abbild aller von der Software abgebildeten Prozesse der Organisation in Echtzeit. Ausgehend davon wird ERP-Systemen eine zweite Funktion zugeschrieben: Sie sollen dieses vollständige Abbild der Informationen über die Organisation zentral verfügbar machen, um dadurch Manager*innen die Steuerung der Organisation zu ermöglichen. ERP-Systeme dienen also sowohl der Integration als auch der Darstellung der Geschehnisse, die an verschiedenen Stellen einer Organisation zu unterschiedlichen Zeitpunkten stattfinden, in einer zentralen Ansicht. Diese zentrale Darstellung soll durch die Zusammenführung aller prozessproduzierten Daten möglich werden.

Was heute als Kernfunktion von ERP-Systemen betrachtet wird, ist ein Ergebnis der in den 1960er Jahren begonnenen Automatisierung der industriellen Produktion mit Hilfe von Mikroelektronik (Pollock und Williams 2009). ERP-Systeme haben ihren Ursprung in Software zur Lagerverwaltung, die allmählich erweitert wurde, um immer weitere Teile des Produktionsprozesses zu integrieren. In den Anfangsjahren der Entwicklung standen Planung und Steuerung des Produktionsprozesses im Vordergrund. Arbeitsprozesse jenseits der Fertigung wurden erst später integriert, indem die Software nach und nach um entsprechende Funktionen erweitert wurde. Der Fokus des modellierten Anwendungsbereichs verschob sich von der Fertigung auf die Organisation als Ganzes.[11] Mit der Erweiterung des Anwendungsbereichs veränderten sich auch die technischen Charakteristika: Die Softwareunterstützung verschiedener Aufgabenbereiche der Organisation wurde in fachspezifische Module aufgeteilt, deren technische Integration (also aneinander anschließende Prozessvorlagen und gemeinsame Datenbasis) die fachliche Integration (also aneinander anschließende Arbeitsprozesse mit geteilten Informationen) sicherstellen sollte. Ein Kennzeichen von ERP-Systemen wurde ihre Multifunktionalität (Klaus et al. 2000). Auch

Integration der Praktiken in der Organisation zentrale Bedeutung zukommt, werden sie hier herausgestellt.

[11] Ein Umstand, der auch in der Entwicklung der Bezeichnungen für diese Klasse von Software zum Ausdruck kommt: Vorläufer der standardisierten Organisationssoftware von heute wurden als Systeme für „Materials Requirements Planning“ (MRP), „Manufacturing Resource Planning“ (MRP II) und „Computer-Integrated Manufacture“ (CIM) bezeichnet. Damit wird als Anwendungsgebiet eindeutig die Fertigung genannt, während der vom Industrieanalysten Gartner geprägte Begriff „Enterprise Resource Planning“ (ERP) den Bezug zur Produktion hinter sich lässt (Pollock und Williams 2009, S. 22–24).

der ERP-Kern des Krankenhausinformationssystems eMed ist ein Ergebnis dieser Entwicklung.

Gleichzeitig wurden die Systeme immer stärker als Werkzeug von Manager*innen dargestellt (und wahrgenommen). Zweck dieses Werkzeugs war es, dass Manager*innen alles, was in der Organisation geschieht, mit der Software überblicken, nach erprobten Vorlagen umgestalten und steuern können sollten. Diese Verschiebung – von einem Hilfsmittel für die Automatisierung der Produktion zum Werkzeug für Organisationsgestaltung – lässt darauf schließen, dass sich einige der Zwecke, für die die Software konstruiert worden ist, und einige der Unsicherheitsreduktionen in den Modellen, die im Entwicklungsprozess eingesetzt worden sind, über die Zeit verändert haben. Diese Veränderungen spiegeln sich aber nur partiell im Code wider, zumindest nicht bei den größten Anbietern: *„[T]he ERP offerings of the top five ERP vendors still use the kernel of MRPII functionality [also den Code, der die Kernfunktionen des rein produktionsorientierten Vorgängersystems erzeugt, D.A.] for the manufacturing planning portion of their systems (Chung and Snyder 2000)*“ (Pollock und Williams 2009, S. 25). Werden Modelle in frühen Phasen des Entwicklungsprozesses verändert, diese Veränderungen aber in den Phasen, in denen Code produziert wird, nicht nachvollzogen (z. B. weil Codeteile aus alten Versionen ohne Anpassung in neue Versionen übernommen werden), so bleibt das performative Modell des Organisierens von den Kontrollversuchen geprägt, für die Software ursprünglich entwickelt worden ist.

Das in der Fallstudie als Teil des Krankenhausinformationssystems untersuchte ERP-System der Firma SAP ist eines der Produkte dieser fünf erfolgreichsten Anbieter*innen. Grundlegend für das Modell des Organisierens in dieser Software ist die „ablaufgesteuerte Prozessorientierung“ (Rolf et al. 1998): Das Modell der Organisation besteht dabei aus einer statischen Grundstruktur, der sogenannten *Aufbauorganisation,* und einer dynamischen Ebene, der sogenannten *Ablauforganisation.*

Im Modell der *Aufbauorganisation* sind alle Organisationseinheiten (OE) inklusive der dort angesiedelten Planstellen abgebildet. Die Elemente der Aufbauorganisation sollen funktionalen Einheiten der Organisation entsprechen. Der genaue Zuschnitt solcher Einheiten hängt dabei von der konkreten Organisation ab. Organisationseinheiten in der Software können sowohl Werksstandorte repräsentieren als auch Abteilungen, Arbeitsgruppen oder ähnliches. Technisch gesehen enthält das Modell der Aufbauorganisation im ERP-System nicht nur die eine Abbildung der statischen Grundstruktur der Organisation, die hier beispielhaft genannt ist und in der Organisationseinheiten aufgabenspezifisch bestimmt werden. Sie besteht zusätzlich noch aus mehreren andere Modellen. In diesen

anderen Modellen wird die statische Grundstruktur der Organisation zum Beispiel aus einer buchhalterischen Perspektive dargestellt. In diesem Fall bilden Kostenstellen die Organisationseinheiten im Modell und die gesamte Aufbauorganisation ist anders strukturiert als im ersten Fall. Unterschiede zwischen verschiedenen Perspektiven ergeben sich zum Beispiel, wenn die Ausgaben mehrerer Abteilungen über eine gemeinsame Kostenstelle abgerechnet werden oder Mitarbeiter*innen der gleichen Arbeitsgruppe ihre Kosten je nach Aufgabe über unterschiedliche Kostenstellen abrechnen. Beim Customizing werden die unterschiedlichen Modelle aufeinander abgebildet. Zur Laufzeit, also wenn Organisationsmitglieder die Software nutzen, wird dann jederzeit automatisch der richtige Bezug, z. B. zwischen der Abteilung, ihren Kostenstellen, Mitarbeiter*innen und zugeordneten Aufgaben, hergestellt (vgl. SAP AG 2001, D.8.16.2.1, S. 8–32).

Das Modell der *Ablauforganisation* setzt sich aus den Modellen der einzelnen Geschäftsprozesse zusammen. Die Software enthält Prozessvorlagen, also generische Kombinationen aus Elementen, die einzelne Prozessschritte (z. B. Wareneingang, Warenprüfung, Warenannahme) repräsentieren sollen. Im Modell jedes Prozessschritts sind notwendige und mögliche Eigenschaften definiert (z. B. beim Wareneingang Warentyp, Menge, Datum, Lieferant*in etc.), ebenso wie mögliche Verantwortlichkeiten (z. B. beim Wareneingang eine Person, die die Ware entgegennimmt) und eine Reihenfolge (z. B. beim Wareneingang immer vor der Warenprüfung). Auch die Prozessvorlagen müssen beim Customizing bearbeitet werden, zum Beispiel, indem entschieden wird, welche Geschäftsprozesse mit welchen Einzelschritten in der jeweiligen Organisation überhaupt zum Einsatz kommen (ob z. B. Wareneingang, Warenprüfung und Warenannahme einzeln unterschieden werden oder alles, was nach der Anlieferung von Waren geschieht, in einem Prozessschritt abgebildet werden soll), welche der Eigenschaften übernommen werden (z. B. welche Warentypen in der Organisation angenommen werden) und wer konkret für welche Schritte verantwortlich sein soll (ob es z. B. eine Lagerist*in gibt oder die Ware bei einer Pförtner*in abgegeben wird). Vor der Inbetriebnahme wird außerdem jeder Schritt in den Prozessen innerhalb der Ablauforganisation einem oder mehreren Elementen der Aufbauorganisation eindeutig zugeordnet und kann damit zur Laufzeit in der Organisation „lokalisiert“ werden (SAP AG 2001).

Der ERP-Kern von SAP soll durch diese Strukturen alle statischen und dynamischen Elemente der Organisation und alle Veränderungen der Beziehungen zwischen ihnen zu jeder Zeit vollständig abbilden. Um diesem Anspruch gerecht werden zu können, wird eine unüberschaubare Menge an Einzeldaten produziert und gespeichert. Während für die Integration der Arbeitsprozesse jeweils nur eine

kleine Auswahl dieser Daten notwendig ist, ist die gewaltige Datenmenge, die von der Software produziert wird, immer dann ein Problem, wenn sich Nutzer*innen einen Überblick über einen Aspekt des Geschehens in der gesamten Organisation verschaffen wollen. Dies ist im ERP-Kern über Berichte möglich, die den Nutzer*innen eine vorstrukturierte Auswahl der Daten aus der von ihnen gewählten Perspektive zeigen (SAP AG 2001). Die Auswahl, Ausgestaltung und Neuanlage solcher Berichte ist ein wichtiger Teil der meisten Customizingprozesse.

Aufbau- und Ablauforganisation werden zusätzlich über sogenannte Module in betriebswirtschaftliche Funktionsbereiche eingeteilt, zum Beispiel Materialverwaltung, Personalverwaltung, Controlling o.ä. (mehr dazu bei Hohlmann 2007). Module bündeln Funktionalitäten und stellen funktionsspezifische Standardberichte bereit. Abbildung 4.7 zeigt die Architektur des ERP-Kerns mit den angegliederten Modulen.

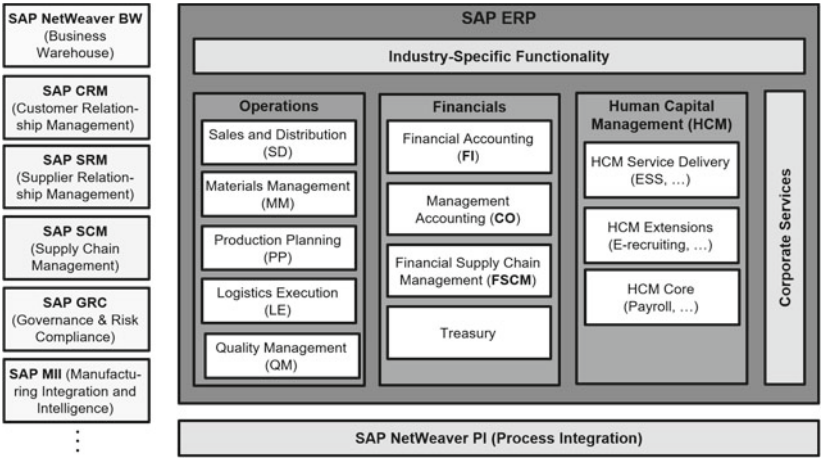

Abbildung 4.7 ERP-Kern *(Boeder und Groene 2014, S. 15)*

Bei der Entwicklung der ersten Version ihrer ERP-Software setzte die Firma SAP ihr Produkt durch eine bis dahin ungewöhnliche zweite Sicht auf die Organisation von den Produkten der Konkurrenz ab, die auch in der später entwickelten Version R/3 (und der in der Fallstudie untersuchten Weiterentwicklung mit der Bezeichnung ERP) noch eine zentrale Position einnimmt: Alle Informationen in der gemeinsamen Datenbasis werden einerseits als Elemente der Arbeitsprozesse modelliert, die sie hervorbringen, andererseits als Elemente der Finanzflüsse

innerhalb der Organisation (Hohlmann 2007). Durch die parallele Modellierung dieser beiden unterschiedlichen Sichten ist es möglich, das Abholen von Materialien aus dem Lager sowohl als Bewegung von Gütern aus einer Abteilung in die andere zu betrachten als auch als Verschiebung von finanziellen Werten zwischen internen Kostenstellen. In beiden Fällen kann konkret festgestellt werden, welche Mitarbeiter*in der Organisation die Aktion durchgeführt hat oder an ihr beteiligt war. Jeder Arbeitsschritt kann damit (je nach Modellierung) einem oder mehreren Mitarbeiter*innen zugerechnet werden. Jede Tätigkeit in der Organisation, die in irgendeiner Form in der Software abgebildet ist, wird dadurch automatisch finanziell bewertet. Theoretisch, d. h. entsprechend der Vorstellung vom Umgang mit der Software, die in dem Modell zum Ausdruck kommt, lässt sich über den ERP-Kern die gesamte Organisation in Echtzeit aus der Perspektive des Rechnungswesens darstellen.

Dieses finanzielle Modell der Organisation erlaubte es den Entwickler*innen der ersten Version von SAP, viele der unterschiedlichen Geschäftsprozesse nach einer einheitlichen Logik zu integrieren, nämlich der des Rechnungswesens (Mormann 2016). Bei der Entscheidung, die Daten, mit denen das Geschehen in der Organisation abgebildet werden sollte, in Echtzeit zu verarbeiten, orientierten sich die Entwickler*innen allerdings nicht an fachlichen Anforderungen von Vertreter*innen des Rechnungswesens, sondern nur an den vorhandenen technischen Möglichkeiten: *„Die SAP-Entwickler boten […] eine technische Lösung für ein organisatorisches Problem an, das bislang überhaupt nicht als Problem identifiziert worden war“* (a.a.O., S. 59). Im Rechnungswesen wurden die finanziellen Werte, die im Unternehmen umgesetzt werden, traditionell im Nachhinein zu festen Zeitpunkten bilanziert. Eine ständige Betrachtung der Wertveränderungen als „Finanzströme“ war den Vertreter*innen des Rechnungswesens fremd (a.a.O.). Dieses für die Funktionalität von ERP-Systemen heute zentrale Merkmal lässt sich eher aus einer Dominanz der naturwissenschaftlich-technisch orientierten Entwickler*innen gegenüber den Repräsentant*innen der Anwender*innen in den frühen Auseinandersetzungen um die Modelle der ersten SAP-Version erklären als aus genuinen Anforderungen der intendierten Nutzer*innen (a.a.O.).

Insgesamt ist das ERP-Kernsystem das Ergebnis einer Vielzahl teilweise höchst kontroverser Aushandlungen zwischen Akteur*innen innerhalb und zwischen sozialen Systemen (Mormann 2016; Hohlmann 2007; Pollock und Williams 2009). Aus diesen Auseinandersetzungen ist eine Software hervorgegangen, in deren Modellen der Versuch zum Ausdruck kommt, „die“ Organisation „an sich“, also alle (in der Vorstellung der an der Entwicklung Beteiligten) relevanten Aspekte aller Arten von Organisationen aus mehreren Perspektiven mit Hilfe einer möglichst hohen Anzahl an Detailinformationen darzustellen. Die

Funktionsbereiche des ERP-Kerns, die in jeder konkreten Variante der Software enthalten sind, die in irgendeiner konkreten Organisation genutzt wird, umfassen das Betriebsmodul, das Finanzmodul und das Personalmodul (Boeder und Groene 2014). Erweiterungen[12] gibt es sowohl für zusätzliche Aufgabenbereiche wie die Integration von Daten, die von Zuliefer*innen übermittelt werden, als auch für verschiedene Branchen, wie Gesundheitswesen oder Einzelhandel. Solche branchenspezifischen Erweiterungen fügen Vorlagen für die Aufbau- und Ablauforganisation sowie für typische Geschäftsprozesse hinzu, verändern aber die Strukturen des ERP-Kernsystems nicht, sondern nutzen sie für die als allgemein angesehenen Aufgaben. Das Modell des Organisierens im Kernsystem bleibt unabhängig von der konkreten Installation erhalten. In diesem Modell des Organisierens besteht eine Organisation aus wohldefinierten, hierarchisch geordneten statischen Einheiten wie zum Beispiel Abteilungen oder Kostenstellen. Diese nehmen veränderbare Einheiten aus der Umwelt, wie zum Beispiel Materialien und Vorprodukte, auf und reichen diese einander so lange in klar vorbestimmten Schritten weiter, bis sie wieder an die Umwelt abgegeben werden. Die Prozesse, in denen Mitglieder von Organisationseinheiten auf diese Art die Produkte der Organisation herstellen, sind vollständig transparent und können mit Hilfe der von der Software erhobenen und gesammelten Daten einfach von Manager*innen gesteuert werden. Der Zweck, auf den dieses Modell des Organisierens die Organisation ausrichten soll, ist die kosteneffiziente, fehlerfreie und zentral vom Leitungspersonal der Organisation beherrschbare Produktion standardisierter Güter. Das Modell des Organisierens im Kern entspricht damit einem kapitalistischen Ideal informationstechnisch integrierter Industrieproduktion.

4.3.2 IS-H: Die Ebene der Erweiterung der generischen Software für das Gesundheitswesen

Die Erweiterung von SAP ERP für das Gesundheitswesen basiert auf der von SAP entwickelten Komponente IS-H, die um zusätzliche, von anderen Firmen entwickelte Module (wie i.s.h.med) erweitert werden kann. Nach Peter Haas und Klaus Kuhn (2011) lässt sich ein KIS aus Anwender*innensicht in drei logische Teilsysteme zerlegen, die über ein *Kommunikationssystem* miteinander und mit einem *elektronischen Zentralarchiv* in Verbindung stehen: Das *administrative Informationssystem* beinhaltet alle Funktionen, die für die Verwaltung der

[12] Eine Übersicht über verbreitete Erweiterungen findet sich in Boeder und Groene (2014, S. 18).

Organisation und die Unterstützung der logistischen Prozesse notwendig sind. Das *medizinische Informationssystem* dient der Organisation und Dokumentation der Behandlung und der Pflege und das *Patient*innenendatenverwaltungssystem* führt die administrativen, medizinischen und organisatorischen Daten zusammen, die die Mitarbeiter*innen des Krankenhauses brauchen, um Behandlungen durchzuführen, abzurechnen und im Einklang mit bestehenden Gesetzen nachzuweisen. Im Krankenhausinformationssystem der Sanusklinik erfüllt IS-H die Funktionen des *Patient*innendatenverwaltungssystems,* des *administrativen Systems* und des *Kommunikationssystems.* Die in IS-H enthaltenen Modelle werden also beim Umgang mit der Software relevant, wenn durch das, was mit der Software getan wird, administrative Arbeitsprozesse berührt werden (also sehr häufig, wie sich noch zeigen wird). Die Elemente, über die diese administrativen Arbeitsprozesse im Modell des Organisierens von IS-H abgebildet sind, lassen sich auf zwei Quellen zurückführen: Erstens enthält IS-H Datenstrukturen und Prozessvorlagen, die eigens für Krankenhäuser entwickelt wurden. Diese krankenhausspezifischen Modelle repräsentieren zum Beispiel Elemente der Behandlung und sind im Zuge der Entwicklung von IS-H eigens für die Software modelliert worden. Zweitens finden sich in IS-H auch Datenstrukturen und Prozessvorlagen, die auf generischen Elementen aus dem ERP-Kern aufbauen. Diese Datenstrukturen und Prozessvorlagen sind das Ergebnis einer Spezifizierung der generischen Elemente, die ebenfalls im Zuge der Entwicklung von IS-H durchgeführt worden ist. Ein Beispiel ist die Organisationsstruktur: In IS-H ist das Grundmodell aus dem ERP-Kern konkretisiert, indem eine Hierarchie von Organisationseinheiten bereitgestellt wird, wie sie im Krankenhaus erwartbar sind: Im Modell der Klinik finden sich als Organisationseinheiten mehrere Fachabteilungen (wie z. B. die allgemeine Chirurgie) und deren Unterabteilungen (z. B. allgemeinchirurgischen Ambulanz, Station und Intensivstation). Diesen Organisationseinheiten sind, ebenso wie im ERP-Kern, dann bauliche Einheiten zugeordnet, Kostenstellen und Leistungen, die dort erbracht werden können. Durch solche Spezifizierungen nehmen die Entwickler*innen von IS-H einen Teil der Aufgaben vorweg, die in Krankenhäusern beim Customizing erledigt werden müssten, wenn dort das ERP-System von SAP ohne die Erweiterung für das Gesundheitswesen eingesetzt werden sollte. Die Mitarbeiter*innen eines Krankenhauses, in dem das ERP-System von SAP und IS-H gekauft worden ist, müssen beim Customizing zum Beispiel nicht mehr selbst definieren, dass jede ihrer chirurgischen Abteilungen eine eigene Ambulanz, eine Station und eine Intensivstation hat, sondern die chirurgischen Abteilungen nur noch konkret benennen und die auswählen, in denen es abweichend von der Regel keine eigene Ambulanz (o.ä.) gibt. Die Unterschiede zwischen den Organisationen des

Gesundheitssektors und denen der Industrie, an denen der ERP-Kern sich orientiert, kommen vor allem im Modell der medizinischen Arbeitsprozesse und in dem Modell der Patient*in zum Ausdruck.

Die medizinischen Arbeitsprozesse werden in IS-H vor allem durch die Kombination von Behandlungselementen modelliert, die *Prozeduren* genannt werden (vgl. auch die Erläuterung dazu auf S. 172). Prozeduren werden im Modell eines medizinischen Arbeitsprozesses im Anschluss an eine *Diagnose* durchgeführt. Prozeduren werden in der Software gleichzeitig auch als Leistungen abgebildet, die von den Organisationseinheiten erbracht und später mit den Kostenträger*innen abgerechnet werden. Wie das ERP-Kernsystem enthält auch IS-H damit parallele Datenstrukturen, die es erlauben, die medizinischen Arbeitsprozesse im Krankenhaus sowohl (als Prozeduren) aus einer (bei der Softwareentwicklung als solche konstruierten) medizinischen als auch (als Leistungen) aus einer (ebenfalls bei der Softwareentwicklung so konstruierten) buchhalterischen Perspektive zu betrachten. Konkrete Diagnosen, Prozeduren und Leistungen sind den größtenteils gesetzlich vorgeschriebenen Katalogen entnommen, die von den Organisationen der gemeinsamen Selbstverwaltung des Gesundheitssystems und dem Gesundheitsministerium in Anlehnung an internationale medizinische Standards definiert werden. Diagnosen und Prozeduren entstammen dem ICD-10 bzw. dem OPS-Katalog, die beide in Anlehnung an die Klassifizierung der WHO im dem Bundesgesundheitsministerium untergeordneten „Deutschen Institut für medizinische Dokumentation und Information“ entwickelt werden (DIMDI 2019). Leistungskataloge sind je nach dem Behandlungsmodus (stationär/ambulant) und nach Art der Kostenträger (z. B. gesetzliche Versicherung, Privatversicherungen) unterschiedlich (SAP AG o. J.d, D3, S. 98–108) und werden zwischen verschiedenen korporativen Akteur*innen ausgehandelt.[13] Die Abrechnung selbst orientiert sich an dem System der Fallpauschalen (SAP AG o. J.d), das ebenfalls im Detail von den Kollektivakteur*innen des Gesundheitssystems definiert wird und gesetzlich vorgeschrieben ist (Feißt und Moltzberger 2016).

Während in der normativen gesellschaftlichen Vorstellung vom Krankenhaus die Patient*in als Empfänger*in der Behandlung im Mittelpunkt steht (Rohde 1974, S. 234 ff.), spielt sie im Modell des Organisierens in IS-H nur eine Nebenrolle. Im Modell, über das die Patient*in in IS-H abgebildet wird, werden zwar

[13] Zu den Organisationen der Interessenvermittlung im Gesundheitswesen und ihren Beziehungen siehe auch Lange (2016) und Gerlinger (2009).

administrative und medizinische Daten über die jeweilige Patient*in zusammengeführt. Dieses Modell taucht beim Umgang mit der Software aber nur selten auf. Das zentrale Element in der Ablauforganisation von IS-H ist der *Fall.*

Über das Modell des Falls wird der Aufenthalt einer Patient*in im Krankenhaus repräsentiert. Dazu werden deren Bewegungen[14] innerhalb der Aufbauorganisation in einer Datenstruktur, eben dem Fall, zusammengefasst. Über den Fall werden Daten der Patient*in mit Daten über die Arbeitsprozesse verknüpft.[15] Anders als bei den Elementen der Behandlung (Prozeduren/Leistungen) und beim Modell der Patient*in (administrative/medizinische Daten) bildet der Fall in IS-H weder eindeutig eine medizinische noch eine buchhalterische Perspektive auf das Geschehen in der Organisation ab. Es findet sich in IS-H auch kein dem Fall entsprechendes paralleles Element, mit dem der Aufenthalt der Patient*in aus einer anderen Perspektive betrachtet werden könnte, die die des Falls in irgendeiner Weise spiegeln würde. Stattdessen werden die für das Krankenhaus gleichermaßen relevanten, aber unterschiedlichen Perspektiven, die medizinische und die buchhalterische, in einem Element zusammengeführt. Damit entspricht diese Struktur in IS-H den in 4.1.1 beschriebenen Standards der Finanzierung des Gesundheitssystems: Fallkategorien (DRGs) *„gruppier[en] Fälle, die klinisch homogen sind und annähernd die gleichen Ressourcen verbrauchen“* (SAP AG o. J.d). Auch bei der medizinischen und pflegerischen Dokumentation sowie bei den Berichten, die das Krankenhausinformationssystem erzeugt, steht der Fall im Mittelpunkt (SAP AG o. J.d).

Obwohl diese Verbindung von Medizin und Rechnungswesen mit der Modellierung des Falls beabsichtigt ist, zeigt sich bei genauerer Betrachtung, dass im Modell des Falls primär die buchhalterische Perspektive auf die Behandlung abgebildet wird. Fälle dienen in IS-H explizit dem Zweck, die im Krankenhaus erbrachten Leistungen mit den Kostenträger*innen abzurechnen (SAP AG o.J.b). Im Modell des Falls werden die Unsicherheiten des Krankenhausaufenthalts der Patient*in, die beim Umgang mit der Software eine Rolle spielen, also so reduziert, dass die Kosten des Aufenthalts am Ende möglichst vollständig abgerechnet werden können. Andere Aspekte der Unsicherheit des Krankenhausaufenthalts sind im Modell des Falls kaum berücksichtigt. Ebenso sieht es in den

[14] Eine Bewegung liegt vor, wenn sich die Zuordnung zu einer Organisationseinheit ändert (SAP AG o. J.b, „Bewegung (IS-H)“).

[15] Am Fall wird gespeichert, welche Krankheit der Grund für den Aufenthalt war (als Diagnose), wann der Patient zu welchem (vordefinierten) Zweck welcher Organisationseinheit des Krankenhauses zugeordnet war (als Bewegung) und welche Personen dort welche Teile der Pflege oder Behandlung durchgeführt haben.

anderen Modellen aus, die extra für IS-H entwickelt worden sind: Sie modellieren vor allem die Abrechnung und die Verwaltung von Patient*innen, nicht aber Spezifika der medizinischen Arbeitsprozesse. Die Funktionalitäten, die sich in IS-H für die Unterstützung der Arbeitsprozesse finden, entsprechen größtenteils generischen Funktionalitäten für die Produktionsplanung und – steuerung (z. B. Materialeingang, Transport, etc.), die im ERP-Kern bereits enthalten sind. Andere Funktionalitäten von IS-H dienen der Überführung der Prozessdaten in die Module für Controlling und Materialwirtschaft, die Teil des ERP-Kerns sind (SAP AG o. J.d). Die übrigen Aufgabenbereiche des Krankenhauses werden vollständig von den Standardmodulen des ERP-Kerns abgedeckt. Das Modell des Organisierens in IS-H entspricht somit größtenteils dem im ERP-Kern. Es unterscheidet sich vor allem dadurch, dass die Organisation im Modell des Organisierens in IS-H andere Produkte herstellt als die im Modell des Organisierens im ERP-Kern: Die Organisation, die im ERP-Kern repräsentiert ist, ist ein generisches Industrieunternehmen, in der Mitarbeiter*innen Materialien, die Kostenstellen zugeordnet sind, zu standardisierten Produkten zusammenfügen. Diese werden an Kund*innen verkauft, was dem Unternehmen Erlöse einbringt. Die Organisation, die in IS-H repräsentiert ist, gleicht diesem Industrieunternehmen in den Grundzügen. Die Mitarbeiter*innen der in IS-H modellierten Organisation nehmen allerdings nicht nur Materialien und ähnliche Gegenstände aus der Umwelt auf, sondern auch Patient*innen, die genauso wie die Materialien im Industrieunternehmen als Ressourcen für verschiedene Produktionsprozesse dienen. In diesen Produktionsprozessen, die denen des generischen Industrieunternehmens so sehr ähneln, werden keine physischen Gegenstände, sondern standardisierte Fälle hergestellt, wobei die Patient*innen als wichtigstes Vorprodukt von Prozessschritt zu Prozessschritt weitergereicht werden, bis die Fallproduktion abgeschlossen ist. Die fertigen Fälle werden dann für Fallpauschalen an Kostenträger*innen verkauft. IS-H, die Erweiterung des generischen ERP-Systems von SAP für den Gesundheitssektor, weicht fast ausschließlich durch Datenstrukturen und Prozessvorlagen von der generischen Grundform ab, die für das Rechnungswesen im Gesundheitssektor relevant sind. Die als medizinisch deklarierten Elemente gleichen strukturell den generischen Elementen und unterscheiden sich nur in ihren Bezeichnungen von diesen. Die Spezifika der Organisationen im Gesundheitswesen werden damit in IS-H auf Besonderheiten des Rechnungswesens reduziert. Die medizinischen Arbeitsprozesse, die in der Software modelliert sind, werden dadurch einer Kontrolle durch die Logik des Rechnungswesens unterworfen.

4.3.3 i.s.h.med: Die Ebene der Ergänzung für Krankenhäuser

i.s.h.med ist eine Erweiterung von IS-H, die vollständig in die SAP-Software für Krankenhäuser integriert ist, aber nicht von SAP selbst stammt, sondern von einer Reihe an wechselnden Unternehmenspartner*innen (weiter)entwickelt wird (Hoeksma 2009). In der Sanusklinik deckt i.s.h.med primär die Aufgaben eines medizinischen Informationssystems ab.[16] Anders als IS-H, bei dem ein Schwerpunkt auf die Neugestaltung der administrativen Arbeitsprozesse gelegt wurde, werden in i.s.h.med vor allem Arbeitsprozesse neu modelliert, die primär von medizinischem Personal im klinischen Alltag ausgeführt werden (SAP AG o. J.a). Dazu erweitert i.s.h.med Datenstrukturen und Prozessvorlagen, die in IS-H erstmalig definiert oder aus dem ERP-Kern übernommen und konkretisiert worden sind, um zusätzliche Parameter. Solche Parameter ermöglichen es zum Beispiel den Nutzer*innen, Daten nach den Kriterien einzelner Fachdisziplinen strukturiert einzugeben und anzuzeigen (z. B. Anzahl der Geburten bei Patientinnen in der Gynäkologie, aber nicht bei Patient*innen der Neurologie) oder die Leistungserbringung auf bestimmte Mitarbeiter*innen oder Organisationseinheiten zu beschränken (z. B. als Verantwortliche für Operationen keine Mitarbeiter*innen der Psychiatrie zuzulassen) (SAP AG o. J.a). Mit i.s.h.med sollen die medizinischen Prozesse im Krankenhaus aus medizinischer Perspektive abgebildet werden: Nicht Kosten oder Haftungsfragen, sondern das, was Ärzt*innen und Pfleger*innen wissen müssen, um den Patient*innen die richtige Behandlung zukommen zu lassen, soll in der Software modelliert werden. Dieser Anspruch wird an den detaillierten Vorlagen für die Arbeit in einzelnen Fachabteilungen ebenso erkennbar wie an der Modellierung der Zusammenarbeit zwischen diesen. Das zentrale Element dieser Modellierung sind *Klinische Aufträge.*

Mit einem Klinischen Auftrag werden medizinische Informationen zu einer Patient*in zwischen den Organisationseinheiten ausgetauscht, wenn Mitarbeiter*innen aus einer Organisationseinheit bei Mitarbeiter*innen der anderen Organisationseinheit eine Behandlung anfordern. Der Auftrag kann zu Beginn ohne Verknüpfung zu bestimmten *Leistungen* erstellt werden, die aus buchhalterischer Perspektive für den Arbeitsprozess relevant sind, aber nicht aus medizinischer (denn aus dieser stehen die Prozeduren im Vordergrund) (SAP AG o. J.a). Während der Behandlung werden die Daten, die das Personal bei

[16] Die Zuweisung der architektonischen Komponenten von Haas und Kuhn (2011) zu den Systemen IS-H bzw. i.s.h.med ist nicht ganz trennscharf, denn Funktionen wie die medizinische Leistungskommunikation und die fachspezifische Dokumentation werden grundsätzlich von IS-H bereitgestellt und von i.s.h.med genauer ausdifferenziert.

der Erfüllung seiner Aufgaben in das auf die Logik ihrer Fachdisziplin zugeschnittene Teilmodul von i.s.h.med eingibt, mit Hilfe des Klinischen Auftrags miteinander verknüpft und von i.s.h.med automatisch um die administrativ notwendigen Elemente ergänzt. In der Online-Softwaredokumentation von i.s.h.med wird angegeben, dass das medizinische Personal den Arbeitsprozess rein aus der medizinischen Perspektive betrachten kann, wenn es klinische Aufträge nutzt, um die Zusammenarbeit mit Hilfe des Krankenhausinformationssystems zu koordinieren (a.a.O.). In dieser Angabe aus der Online-Softwaredokumentation, die hier beispielhaft genannt wird, um zu verdeutlichen, wie die Software in diesen Dokumenten beschrieben wird, zeigt sich der Zweck, dem i.s.h.med der offiziellen Darstellung zufolge dienen soll. In dieser Ergänzung der standardisierten Organisationssoftware für Krankenhäuser wird die Koordination der unterschiedlichen Teilaufgaben des Behandlungsprozesses mit ihren je speziellen Eigenlogiken in den Mittelpunkt gestellt. Auf dieser technischen Ebene des Krankenhausinformationssystems findet sich keine eigenständige Funktionalität für die administrativen Aspekte des Krankenhauses. Sie ist dezidiert dafür entwickelt worden, die beiden vorgelagerten technischen Ebenen (ERP-Kern und IS-H) zu ergänzen (Siemens AG 2009).

Die Modelle, in denen diese medizinische Perspektive auf die Arbeit im Krankenhaus zum Ausdruck kommen soll, haben nur wenig Auswirkung auf das Modell des Organisierens, das den Umgang mit dem KIS eMed in der Praxis beeinflusst. Der Grund dafür ist, dass das Modell des Organisierens in i.s.h.med nicht nur durch die Modelle bestimmt wird, die bei der Entwicklung dieser technischen Ebene neu geschaffen worden sind, sondern sehr stark durch die Modelle der vorgelagerten Ebenen vorstrukturiert wird. In i.s.h.med allein wird nur ein Teil einer Organisation repräsentiert; grundlegende Elemente, die Teil der Modelle der Arbeitsprozesse in i.s.h.med sind, werden mitsamt ihrer Funktionslogik aus dem ERP-Kern oder aus IS-H übernommen. Beispiele dafür sind die Prozeduren und Leistungen und die Unterteilung der Organisation in Untereinheiten, die in den Klinischen Aufträgen angefordert werden. Solche Elemente von anderen technischen Ebenen sind in den Modellierungsprozessen für i.s.h.med übernommen worden, haben sich im Code abgelagert und steuern zur Laufzeit von eMed viele der Funktionen der Software. Der Klinische Auftrag kann zwar als zentrales Element von i.s.h.med betrachtet werden. Er ist jedoch immer an einen Fall gekoppelt, welcher weiterhin in der Logik von IS-H verbleibt, in der das Krankenhaus als eine Art „Fabrik für Fälle“ vor allem aus buchhalterischer Perspektive dargestellt wird. Der Fall wird auch in i.s.h.med an vielen Stellen genutzt, um Daten zu sortieren, auszuwählen und darzustellen. Von der Anzeige der Patient*innendaten bis zu den medizinischen Berichten, die in i.s.h.med definiert

sind, bleibt die Fallorientierung der Arbeitsprozesse dominant. Damit werden die Unsicherheiten der Krankenbehandlung auch über i.s.h.med immer wieder so reduziert, dass vor allem den Anforderungen an eine vollständige Abrechnung der Arbeit genüge getan wird. Die Orientierung an den Patient*inneninteressen auf medizinisch adäquate Behandlung kommt nur in einzelnen, speziell für i.s.h.med konstruierten Modellen, zum Ausdruck.

Hinzu kommt, dass i.s.h.med eine Ergänzung zu IS-H darstellt und kaum Elemente der anderen Software ersetzt. Wenn eMed in ein Krankenhaus eingeführt wird, müssen alle Aufgaben, die für die Einführung von i.s.h.med notwendig sind, zusätzlich zu den Aufgaben erfüllt werden, die bei der Einführung von IS-H anfallen. Das bedeutet vor allem, dass zusätzlicher Arbeitsaufwand beim Customizing entsteht, wodurch zusätzliche Kosten anfallen und zusätzlicher Bedarf an Aushandlungen (mit dem entsprechenden Konfliktpotential) entsteht. Das Customizing der Elemente aus IS-H bleibt notwendig, um die generischen Aspekte des Krankenhausinformationssystems so weit zu konkretisieren, dass es lauffähig wird. Viele dieser Konkretisierungen sind unabdingbar, wenn die Software genutzt werden soll: Organisationseinheiten unterschiedlicher Art (Abteilungen, Kostenstellen etc.) *müssen* bestimmt, benannt und einander zugeordnet werden, konkrete Mitarbeiter*innendaten *müssen* eingegeben, Nutzer*innenkonten *müssen* angelegt werden und so weiter, damit die Software irgendwelche Funktionen erzeugt. Im Gegensatz zu diesen unabdingbaren Aufgaben sind die Anpassung und Aktivierung der Funktionalitäten von i.s.h.med beim Customizing optional. Jede Entscheidung für eine Funktionalität von i.s.h.med erzeugt dadurch zusätzliche Komplexität beim Umgang mit der Software in einer Arbeitsumgebung, die auch ohne diese Funktionalität bereits hochkomplex ist: Fachspezifische Arbeitsprozesse müssen differenziert vordefiniert, von Nutzer*innen erlernt und regelmäßig so ausgeführt werden, dass die Software ihren Zweck im Arbeitsprozess erfüllen kann.

Beim Customizing in der Sanusklinik ist auf viele der möglichen Funktionalitäten verzichtet worden, wie Abschnitt 4.2.2 gezeigt hat. Die wenigen i.s.h.med-spezifischen Anpassungen sind in den meisten Fällen optional. Entweder haben sich abteilungsspezifische Regeln dafür herausgebildet, oder ihre Interpretation ist den Nutzer*innen freigestellt. Wie im folgenden Abschnitt sichtbar wird, sind solche optionalen zusätzlichen Anforderungen in den Aushandlungen um die Praxis des Umgangs mit dem KIS besonders einfach zu umgehen.

4.4 Das konkrete Handlungssystem der OP-Planung

Während die technischen Strukturen der drei generischen Ebenen des KIS unabhängig von den Gegebenheiten in der Sanusklinik beschrieben werden können, ist die Software, die in der alltäglichen Praxis genutzt wird, das Ergebnis des Customizings und der abteilungsspezifischen Anpassungen. Die Praktiken im Umgang mit eMed unterscheiden sich zwischen den Abteilungen, da zwar alle in die gleiche lokale Ordnung der Sanusklinik eingebunden sind, aber im Alltag relativ unabhängig voneinander arbeiten (vgl. 4.1.2 und 4.1.3). Die einzelnen Abteilungen müssen daher in einer strategischen Organisationsanalyse der OP-Planung als eigene Handlungssysteme betrachtet werden.

OP-Planung ist in allen Abteilungen eine relevante Aufgabenstellung; vor allem das Verhältnis von Routine- und Notfalloperationen und damit die Dynamik des OP-Plans und die generelle Auslastung der verfügbaren OP-Ressourcen variiert. In den Abteilungen für *allgemeine* und für *spezielle* Chirurgie,[17] die im Folgenden verglichen werden, sind mehrere Disziplinen organisatorisch zusammengefasst. Während in einigen Disziplinen dringende Eingriffe eher die Ausnahme darstellen (z. B. Urologie oder Orthopädie), kommen sie in anderen Abteilungen häufiger vor (z. B. Neuro-, Herz-, oder Unfallchirurgie). In einigen Abteilungen fallen regelmäßig mehr Operationen an, als mit den verfügbaren Kapazitäten an einem Tag erledigt werden können, in anderen ist das Verhältnis ausgeglichener. In der Allgemeinen und in der Speziellen Chirurgie gibt es eine ähnliche, relativ hohe Notfallquote. Notfälle sind in beiden Abteilungen häufig auch lebensbedrohlich. Die OP-Pläne sind dementsprechend in beiden Abteilungen sehr dynamisch. Zusätzlich übersteigt das Operationsaufkommen in beiden Abteilungen regelmäßig die Kapazitäten. In beiden Abteilungen darf regelhaft auch außerhalb der Kernzeiten, also vor allem nach dem Ende der Tagschicht um 16 Uhr, operiert werden, allerdings nur in einem gewissen, genau festgelegten Umfang.

Um das Thema Operationen herum bildet sich ein Ensemble von Unsicherheiten unterschiedlichen Ursprungs, unterschiedlicher Relevanz und unterschiedlicher Möglichkeiten, diese zu kontrollieren. Das Operationsaufkommen bildet auch in der Sanusklinik die größte Unsicherheit der OP-Planung. Die Kontrolle darüber, welche Operationen anstehen, liegt außerhalb des OP-Bereichs. Die Informationskanäle, über die Mitarbeiter*innen des OP-Bereichs von neuen Operationen erfahren können, bilden damit eine der zentralen organisatorischen

[17] Die Abteilungsbezeichnungen sind für die Fallstudie verändert worden.

Ungewissheitszonen der Handlungssysteme. Auch die Unsicherheiten der Logistik sind durch die Organisation vorstrukturiert: Wenn im Tagesverlauf mehr Operationen anfallen, als mit den verfügbaren Kapazitäten erledigt werden können, liegt es meist daran, dass mit Personal besetzte OP-Säle fehlen. Die Gründe dafür sind strukturell: Die Sanusklinik hat insgesamt ein hohes Operationsaufkommen, aber nur wenige räumliche Puffer und wenig zusätzliches Personal, das Krankheits- und Urlaubszeiten ausgleichen kann. Ungewissheitszonen liegen damit im Zugang zu Sälen und in der Verfügbarkeit von Personal.

Die Ungewissheitszone „Verfügbarkeit von chirurgischem Personal" wird stark über die Profession mitstrukturiert: In der professionellen Identität von Chirurg*innen ist angelegt, dass sie jederzeit einsatzbereit sind und sich der Autorität der ranghöheren Chirurg*innen weitgehend unterwerfen.[18] Auch in der Sanusklinik ist diese professionelle Identität regelmäßig handlungsleitend; Chirurg*innen mögen unglücklich mit den Ergebnissen der Planung sein, folgen aber in der Regel dem OP-Plan und passen sich damit an die Vorgaben der Planer*innen an. Sie sind gleichzeitig auch die Hauptakteur*innen im Spiel um die Planung, denn die Erstellung des OP-Plans liegt in der Hand der Chirurgie und ist zentral mit allen anderen Spielen in den chirurgischen Abteilungen verbunden.

Anders sieht dies bei Pflegekräften und Anästhesist*innen aus. In der professionellen Hierarchie der Medizin stehen beide Gruppen unter den Chirurg*innen (mit deutlichen Unterschieden in der Distanz); in der Organisation bilden sie außerdem eigenständige Abteilungen mit anderen Spielen im Innen- und Außenverhältnis. Die Mitglieder beider Gruppen berufen sich in der Sanusklinik aktiv auf ihr Recht auf verlässliche Arbeitszeiten und versuchen diese weniger im Planungsspiel, sondern eher durch Spielzüge in anderen Handlungssystemen durchzusetzen, in denen das Machtgefälle zur Chirurgie weniger stark ins Gewicht fällt. Besonders umkämpft ist im Klinikum die Einhaltung der Betriebszeiten der OP-Säle. Weniger als zehn der über 50 Säle dürfen nach dem regulären Ende der Arbeitszeit am späten Nachmittag noch für Operationen genutzt werden. Die Betriebszeiten sollen eine verlässliche Personalplanung in den Servicebereichen, also unter anderem in Pflege und Anästhesie, ermöglichen. In beiden

[18] Zur spezifisch chirurgischen Ausprägung der professionellen Identität vgl. die Arbeiten von Joan Cassell (1987). Professionssoziologische Studien gibt es dazu nicht. Die hier erwähnten Besonderheiten der Chirurgie im Vergleich zu anderen medizinischen Spezialisierungen werden aber in soziologischen Studien thematisiert, in denen Chirurg*innen Teil des Forschungsgegenstands sind (z. B. Kellogg 2009, 2011). Die auch im Fall beobachtete starke Bedeutung der Hierarchie speziell in der Chirurgie zeigt auch Vogds Arbeit zu ärztlichen Entscheidungsprozessen im Krankenhaus (Vogd 2004a, 2004b).

Bereichen gelten vorhersehbare Arbeitszeiten als hohes Gut und ungeplante Überstunden sollen die Ausnahme sein. Diese Ansicht widerspricht der Logik, die in den chirurgischen Abteilungen gefolgt wird. Hier wird mit der Planung eher das Ziel verfolgt, möglichst viele Operationen pro Tag zu absolvieren. Die Betriebszeiten der Sanusklinik sind das formale Ergebnis eines Kompromisses zwischen den Berufsgruppen in diesem dauerhaften Konflikt. An seiner Aushandlung war auch der Vorstand der Sanusklinik beteiligt, für den Arbeitszeitgesetze mehr Gewicht haben als innermedizinische Hierarchien.

Der Konflikt besteht jedoch weiter fort: Die OP-Planung wird ausschließlich von Chirurg*innen durchgeführt. Diese nehmen mögliche Überschreitungen der erlaubten Betriebszeiten bei der Planung deutlich eher in Kauf als Anästhesist*innen oder Pflegekräfte. Der OP-Koordinator greift daher täglich in die Planung in einzelnen Abteilungen ein und setzt geplante Operationen ab, wenn er überzeugt ist, dass sie nicht rechtzeitig beendet werden können. Die Betriebszeiten und seine formale Position, die ihm ein Vetorecht gegen die OP-Pläne der Abteilungen sichert, sind dabei seine Ressourcen. Aus dieser im Kern konfliktären Konstellation zwischen den Berufsgruppen folgt insgesamt, dass die Verfügbarkeit von Pflegekräften und Anästhesist*innen in den Spielen um die Planung eine dauerhafte Ungewissheitszone bildet.

Da die Studie erst mehrere Jahre nach Einführung des Systems in die beobachteten Abteilungen beginnt, können die früheren Abläufe bei der OP-Planung nur aus den geschilderten Erinnerungen der Akteur*innen rekonstruiert werden. Das allgemeine Handlungsproblem der OP-Planung bleibt durch die Softwareeinführung unverändert und ist in beiden Abteilungen gleich: Es muss entschieden werden, wer (aus Chirurgie, Anästhesie und Pflege) was (Patient*in und Eingriff) wann (Tag und Uhrzeit) und wo (in welchem Saal) operiert. In beiden untersuchten Abteilungen haben sich nach Ansicht der Befragten (Abteilungsmitglieder und Außenstehende) im Großen und Ganzen die Konturen dieser Abläufe durch die Einführung von eMed kaum verändert. Die Gründe sind allerdings höchst unterschiedlich.

Die allgemeine Chirurgie ist als Pilotabteilung eng in das Customizing von eMed eingebunden gewesen. Die Planer*in hat ihre bevorzugten Arbeitsabläufe so gut wie möglich in der Software abgebildet und umgekehrt die Software in die Planungspraktiken der Abteilung integriert. Die spezielle Chirurgie hat den entgegengesetzten Weg gewählt: Die Chefärzt*in versteht den Umgang mit Software als reine Verwaltungsaufgabe, vor der ihre ärztlichen Mitarbeiter*innen geschützt werden müssen. Nur im Sekretariat soll mit der neuen Software gearbeitet werden. Die Planung bleibt alleinige Aufgabe der Chirurg*innen, die Ergebnisse

werden an das Sekretariat übergeben und dort in eMed übertragen. Die Planungspraktiken sind so weit wie möglich vom Umgang mit der Software entkoppelt. Auch nachdem die Chefärzt*in den „eMed-Bann“ für ihre Ärzt*innen zwei Jahre nach der Einführung aufhebt, ändert das wenig daran, dass die Planung größtenteils als Abstimmungsprozess zwischen Chirurg*innen betrachtet wird. Dass der fertige Plan nicht nur für die Verwaltung produziert wird, sondern auch für Pflegekräfte und Anästhesist*innen, also für den medizinischen Arbeitsprozess an sich, eine Rolle spielt, scheint weniger wichtig.

Die nun folgende Analyse der Spiele, in die die Modelle des Organisierens einbezogen werden, zeigt, dass viele der Widerstände, die in Abschnitt 4.2.3 angedeutet worden sind, sich nicht einfach als individuelles Fehlverhalten, Technikfeindlichkeit oder Gedankenlosigkeit deuten lassen. Stattdessen zeigen sich darin Reaktionen der Akteur*innen auf die verschiedenen Versuche, die Praktiken im OP-Bereich durch Modelle des Organisierens und organisatorische Regeln zum Umgang mit eMed zu kontrollieren.

4.4.1 Planungsspiele in der Allgemeinen Chirurgie

Die Allgemeine Chirurgie ist eine der größeren chirurgischen Abteilungen der Sanusklinik. Hier arbeiten über 30 Chirurg*innen, eingeteilt in feste Teams, die sich auf unterschiedliche Arten von Operationen spezialisiert haben. Neben einem Grundaufkommen an vorhersehbaren Routineeingriffen gibt es in der allgemeinen Chirurgie viele Notfälle, die entweder sofort oder spätestens am nächsten Tag operiert werden müssen. Dr. Natal, der leitende Oberarzt und Planer der Abteilung, schätzt, dass *„die OP-Planung zu zwei Dritteln nicht so [ist], wie man sie am Vortag gemacht hat“*.

Trotz dieser großen Unvorhersehbarkeit verläuft die OP-Planung sehr routiniert; sowohl die Aufgabenverteilung als auch die Abläufe sind klar geregelt und innerhalb der Abteilung kaum umstritten. Die Verantwortung für die Entscheidungen darüber, wer was wann und wo operiert, ist klar verteilt und folgt der formalen Hierarchie: Die Teamleiter*innen entscheiden, wer operiert, die OP-Leitung, also die leitende Pflegekraft, teilt die OP-Pflegekräfte für die Eingriffe ein, die Blockleiter*in aus der Anästhesie bestimmt, wer welche Narkosen betreut. Dr. Natal, der leitende Oberarzt der Chirurgie, bestimmt, welche Operationen wann und in welchem Saal durchgeführt werden.

Die Kontrolle über das chirurgische Personal liegt durch diese Aufteilung bei den Teamleiter*innen. Sie besetzen damit eine Machtposition im Spiel. Zwar legt eMed fest, dass bei der Planung Operateur*innen eingetragen werden müssen,

sieht aber keine Funktionen für die Personalplanung vor. In das Feld, das die Daten über Operateur*innen in eMed speichert, können keine beliebigen Daten eingegeben werden, sondern es muss ein Eintrag aus einer Liste ausgewählt werden. Was in dieser Liste möglicher Einträge steht, gehört zu den abteilungsspezifischen Anpassungen im Customizing. In der allgemeinen Chirurgie enthält die Liste nur die Bezeichnungen der Teams, nicht etwa die Namen einzelner Chirurg*innen. Welche Mitglieder des Teams letztendlich am Operationstisch stehen sollen, wird innerhalb des Teams entschieden. Diese Entscheidung ist auch nach der Einführung von eMed für Außenstehende nicht transparent. So berührt die Software die Machtposition der Teamleiter*innen nicht.

Diese sorgen dafür, dass für jeden Eingriff die notwendige Zahl an Operateur*innen im OP-Saal erscheint, und entscheiden ohne Einmischung der Abteilungsleitung darüber, wie die Operationen innerhalb ihres Teams verteilt werden. Im Gegenzug akzeptieren sie, dass die restlichen Planungsentscheidungen von Dr. Natal getroffen werden. Im Planungsprozess mit eMed stellt das chirurgische Personal damit faktisch keine Ungewissheitszone dar.[19]

> *Dr. Natal (OP-Planer der Allgemeinen Chirurgie): Die Patienten suchen sich die Teams selber aus. Und das ist auch ganz alleine die Hoheit des Teamoberarztes. Punkt. Da mischen wir uns überhaupt nicht ein. […] Im Prinzip haben alle Oberärzte die gleichen Rechte am SAP, aber die werden natürlich nicht wahrgenommen. Das heißt, da gibt es aber auch gar keine Autoritätsprobleme. In anderen Abteilungen gibt es durchaus Querelen in dem Bereich, wo jeder versucht so 'n bisschen seine Möglichkeiten auszuspielen. Das gibt es bei uns gar nicht. Das Programm wird von mir gemacht oder ich gebe es ab an jemanden, der es macht, und dann macht es der. Und zwar uneingeschränkt.*

Dieses Arrangement entspricht laut der Aussagen von Dr. Natal den Vorstellungen der Chefärzt*in, die die Autorität der Verantwortlichen stützt, ohne ins Alltagsgeschäft einzugreifen.

Auch der Umgang mit Notfällen wird durch diese eindeutige Aufgabenverteilung zur Routinesache: Alle Notfälle werden telefonisch an den Planer Dr. Natal gemeldet. Er allein stimmt sich mit der Blockleiter*in und der OP-Leitung ab, um Personal für die neue Operation zu organisieren, und er allein entscheidet auch darüber, wo diese in den bestehenden Plan eingefügt wird.

---

[19] Das Verhältnis von Planer*in und Teamleiter*innen wurde nicht untersucht. Die Verfügbarkeit von Chirurg*innen der allgemeinen Chirurgie wurde jedoch von keiner der Befragten in irgendeiner Weise problematisiert. Zumindest, soweit es den Umgang mit eMed betrifft, stellt diese Verfügbarkeit im Untersuchungszeitraum und in der rekonstruierten Vergangenheit faktisch keine Ungewissheit der OP-Planung dar, da sie vollständig durch das Arrangement im Handlungssystem kontrolliert und durch die Software auch nicht beeinflusst wird.

Vor der Einführung von eMed kontrolliert Dr. Natal so durch seine Entscheidungshoheit in allen Fragen der Planung allein die Informationskanäle, über die alle für die Planung notwendigen Einzelinformationen zusammengeführt werden: die anstehenden Operationen der Abteilung, das Personal, die verfügbaren Säle.

> *Fr. Lewitscharoff (OP-Leitung der Allgemeinen Chirurgie): Die OP-Planung lief über 'nen Zettel. [Lacht] Der leitende Oberarzt hat das immer schon gemacht. Da haben wir das eigentlich auch immer schon so … Der hatte Voranmeldungen, die bekam er … Also im Prinzip ähnlich wie jetzt. Der macht ja den OP-Plan aus den Voranmeldungen der einzelnen OP-Teams. Und so hat er das vorher auch gemacht. Bloß in Zetteln. Da bekam er für jede Operation, für jede Voranmeldung einer Operation 'nen Zettel von dem OP-Team und daraus wurde dann der OP-Plan erstellt. Auf Papier. Und dann ging das Ganze zu einer Sekretärin von der Anästhesie, die dann den Gesamt-OP-Plan für alle geschrieben hat. So. Und dann hatte man im Prinzip so was [Sie holt einen Ausdruck des aktuellen OP-Plans heraus].*

OP-Leitung und Blockleiter*in der Anästhesie kontrollieren jeweils das Personal ihrer Berufsgruppen und besetzen damit weitere Machtpositionen im Planungsspiel. Pflege und Anästhesie besitzen jeweils eigene Personalpläne, die für die Mitglieder dieser Abteilungen bindend sind. Solange die beiden die Informationskanäle zwischen den Berufsgruppen kontrollieren, besetzen sie auch in ihren internen Spielen Machtpositionen.

4.4.1.1 Widerstände

Die Allgemeine Chirurgie diente bei der Einführung von eMed in den OP-Bereich als Pilotabteilung (vgl. Abschnitt 4.1.3). Da Mitarbeiter*innen der Allgemeinen Chirurgie dadurch am Customizing beteiligt waren, ist die Erwartung naheliegend, dass der Umgang mit eMed den Vorgaben entspricht, die im Rahmen dieses Prozesses festgelegt worden sind. Dies ist jedoch nicht der Fall.

4.4.1.1.1 Das Spiel mit den OP-Aufträgen

Die Koordination zwischen den drei Berufsgruppen im OP ist nur möglich, wenn die OP-Leitung und die Blockleiter*in wissen, für welche Aufgaben sie ihre Mitarbeiter*innen einteilen sollen. Dazu müssen viele Detailinformationen zu anstehenden Operationen zwischen den behandelnden Chirurg*innen, Pflegekräften und Anästhesist*innen ausgetauscht werden, die bei der Operation zusammenarbeiten sollen. Nachdem die „Zettel" durch eMed abgelöst worden sind, geschieht dies über die OP-Aufträge in der Software. Über diese ist es prinzipiell möglich, umfangreiche medizinische Daten zur Patient*in anzugeben. Erzwungen werden aber nur die Eingaben, die für die Abrechnung nötig sind, und die, die von den zentralen Akteur*innen im Rahmen des Customizing als

Pflichteingaben festgelegt wurden, also zum Beispiel Angaben darüber, wie die Patient*in auf dem OP-Tisch gelagert werden soll, ob eine Infektion mit resistenten Keimen vorliegt o.ä. Insgesamt sind es nur sehr wenige Pflichteingaben, mit denen rein medizinische (und nicht gleichzeitig abrechnungsrelevante) Aspekte des Zustands der Patient*in in eMed beschrieben werden. Allein auf Basis der Pflichteingaben lässt sich nicht genau bestimmen, wie komplex und dringlich eine Operation ist.

Dr. Natal setzt durch, dass seine Chirurg*innen nicht zu viele zusätzliche Details in die OP-Aufträge einfügen, denn detaillierte Fallbeschreibungen in eMed sind für ihn Zeitverschwendung. Wenn in einem OP-Auftrag Informationen fehlen, die er für die Planungsentscheidung braucht, wird die Veranlasser*in angerufen. Er legitimiert dieses Vorgehen, indem er sich auf die medizinische Tradition der diskursiven Konstruktion eines Falls aus einer Menge an Einzelinformationen (Atkinson 1995) beruft und die Möglichkeit, medizinische Entscheidungen nach Aktenlage zu treffen, grundsätzlich bestreitet:

> *Dr. Natal (OP-Planer der Allgemeinen Chirurgie): Die wahre Planung findet natürlich individuell am Patienten statt, […] durch seine individuelle Dringlichkeit. Die Organisation, dass das stattfindet, und die Abwägung findet logischerweise nicht im SAP statt. Sondern das ist, ja, orale Koordination. Also* ***dort wird miteinander gesprochen, der Fall dargestellt*** *[…] Die Entscheidungen werden ja von Menschen getroffen. Das SAP ist das Kondensat, das heißt also, damit jeder hinterher drauf zugreifen kann auf einen fixen Punkt, dafür ist das SAP da. Aber […]* ***man nutzt nicht das SAP, um die Entscheidung zu treffen. Weil man im SAP eben nicht lesen kann****, sozusagen* ***wie schwierig ein Patient ist oder so.*** *Das kann man jemandem erzählen, aber man kann sich nicht 'ne halbe Stunde hinsetzen und eingeben, was für Macken ein Patient hat.*

Die medizinischen Daten in den OP-Aufträgen sind damit korrekt, aber ungenügend, um die Arbeit des Planers zu beurteilen oder gar in Frage zu stellen.

4.4.1.1.2 Das Spiel mit den Planzeiten

Auch die Daten, die der Planer selbst eingibt, sind nicht umfangreich und präzise genug, um sie ohne mündliche Absprachen zu verstehen. Besonders auffällig wird dies bei den Planzeiten: Im OP-Plan in eMed sind alle Operationen des Tages mit zwei Stunden Dauer angesetzt, egal, ob es sich um einfache Routineeingriffe oder um sehr aufwändige Kombinationseingriffe o.ä. handelt. Obwohl Operationen in der allgemeinen Chirurgie in der Praxis zwischen 30 Minuten und 16 Stunden dauern, wird ihr Zeitbedarf über den OP-Plan in eMed vereinheitlicht repräsentiert und damit auf eine Weise abgebildet, die für die Koordination bedeutungslos ist. Eine Zeitschätzung für das Tagesprogramm, bei der die real zu erwartende

Varianz der Operationsdauer berücksichtigt wird, ist durchaus möglich. Der Planer Dr. Natal führt so eine Zeitschätzung auch täglich durch. Die Ergebnisse, also geschätzte Operationszeiten für die Eingriffe des Tages, werden nur nicht in eMed eingetragen. Dr. Natal gibt diese abseits von eMed weiter, zusammen mit allen anderen Informationen, die OP-Leitung und Blockleiter*in brauchen, um ihre eigenen Planungen durchzuführen. Dazu treffen sich die beiden Mediziner*innen täglich, um die geplanten Operationen zu besprechen. Die OP-Leitung holt fehlende Informationen meist im Anschluss an dieses Gespräch telefonisch beim OP-Planer ein.

Obwohl Dr. Natal erklären kann, wie er zu einer realistischen Zeitschätzung kommt und diese an die Verantwortlichen der Servicedisziplinen weitergibt, bestreitet die OP-Leitung, dass eine solche überhaupt möglich sei. Auch hier wird wieder auf die Logik der Profession verwiesen:

> *Fr. Lewitscharoff (OP-Leitung der Allgemeinen Chirurgie): Es werden auch keine Zeiten geplant. […] Aber* ***die stimmen auch in anderen Abteilungen nicht*** *[…] Die wissen das vorher manchmal nicht. Man kann zwar manchmal im Laufe der Operation dann erkennen,* ***wenn die erst mal angefangen haben*** *und alles ist gut in Übersicht,* ***dann können die vielleicht schon mal was sagen****, so während der Operation. Aber* ***vor der Operation nicht****. Zu 99,9 Prozent stimmen diese Listen sowieso nicht. […] Auch* ***die Fälle sind unterschiedlich****. Man muss einfach erst mal vor Ort sein und wirklich einmal kucken. […] Es kann immer alles irgendwie Komplikationen geben und Schwierigkeiten […]. Und wie sich das zeigt, ist das auch fast überall so.*

Die OP-Leitung begründet das Vorgehen des Planers hier als folgerichtiges Umgehen mit den allgemeinen Bedingungen, die angeblich für Operationen immer und überall gelten: Patient*innen sind unterschiedlich und Eingriffe verlaufen unvorhersehbar. Dies steht im Einklang mit der Logik des Einzelfalls, die typisch für die medizinische Profession ist. Eine OP-Planung, die eine realistische Einschätzung der OP-Dauer beinhaltet, wird aus Sicht der OP-Leitung nicht von Dr. Natals Planungspraxis verhindert, sondern ist grundsätzlich nicht möglich. Gegenbeispiele aus anderen Abteilungen verweist sie ins Reich der Phantasie.

4.4.1.1.3 Das Stornierungsspiel

Eine weitere Besonderheit des OP-Plans der allgemeinen Chirurgie lässt sich nicht durch Bezug auf die Profession erklären: Operationen, die im Tagesverlauf abgesetzt werden, sollen in eMed storniert werden. Dieses Vorgehen ist im Benutzer*innenhandbuch zum Umgang mit eMed in der Sanusklinik vorgeschrieben, eine Vorgabe, die der OP-Koordinator Dr. Mosser im Interview bestätigt. Werden Operationen in eMed storniert, verschwinden die OP-Aufträge vom Plan und

erscheinen wieder in der Planungsübersicht. OP-Aufträge haben kein Ablaufdatum. Einmal eingeplant, wird in eMed nur ihre Startzeit nach hinten verschoben, solange sie nicht als erledigt markiert werden. Dadurch wechseln OP-Aufträge, die am Vortag eingeplant, aber weder als abgeschlossen markiert noch storniert worden sind, automatisch auf den Plan des nächsten Tags, wenn die in der Software hinterlegte Betriebszeit für den eingeplanten OP-Saal abgelaufen ist.

Dr. Natal storniert verschobene Operationen nicht, sondern fügt bei der täglichen Planung nur neue OP-Aufträge hinzu. Im Ergebnis enthält der OP-Plan der allgemeinen Chirurgie jeden Tag viel mehr Operationen, als bewältigt werden können. Dieses Unterlassen verhindert, dass irgendjemand außer den zentralen Akteur*innen eine Möglichkeit bekommt, den OP-Plan oder auch nur den Stand des Geschehens in den Sälen der allgemeinen Chirurgie zu beurteilen.

> *Fr. Lewitscharoff (OP-Leitung der Allgemeinen Chirurgie): Also für mich persönlich ist das kein Problem. Es ist natürlich für andere. Weil ich es jetzt weiß. Für andere, wie den [OP-Koordinator], für den wird's 'n Problem, weil der weiß das jetzt grad nicht. […] Es wär' einfacher für alle Beteiligten, wenn sie das [d. h. die stornierten OP-Aufträge aus dem Plan] rausnehmen würden. […] Aber … ja.*

Anders als im Fall der fehlenden Planzeiten, die die OP-Leitung ausführlich begründen konnte, indem sie auf die Logik der Profession Bezug nahm, fehlt ihr im Fall der fehlenden Stornierungen eine Erklärung. Der Verzicht auf Stornierungen stellt für die Koordination nach Aussage der OP-Leitung kein großes Problem dar, ist aber auch nicht förderlich. Die Stornierung eines OP-Auftrags ist weder zeitaufwändig noch kompliziert. Vor allem in der Allgemeinen Chirurgie, in der die planenden Akteur*innen sehr routiniert mit eMed umgehen, überrascht es daher, dass der Planer auf Stornierungen verzichtet. Dieses Vorgehen muss nicht ausführlich und explizit legitimiert werden, so wie es bei den anderen unvollständigen Eingaben der Fall war, denn der unübersichtliche Plan in eMed beeinträchtigt die Lösung des Handlungsproblems nicht. Über die Plantafel kann der Verlauf des Geschehens in den OP-Sälen nur verfolgt werden, wenn regelmäßige Abweichungen zwischen Plan und Wirklichkeit nicht vorkommen. Im Tagesverlauf neu angemeldete Notfälle werden auf der Kalenderansicht sogar überhaupt nicht angezeigt. In der allgemeinen Chirurgie sind Notfälle auch bei genauester Zeitschätzung nicht vermeidbar. Pflegekräfte verbringen den ganzen Tag im OP-Bereich und bekommen daher ohne Blick auf den Plan mit, wann die nächste Operation ansteht. Die Mediziner*innen werden telefonisch benachrichtigt, wenn ihre Operationen beginnen. Dass sie den wahren Plan nicht kennen, stört nicht bei der Lösung des Handlungsproblems.

4.4.1.2 Verteilung der Kontrolle in den Spielen mit eMed

Die Modelle des Organisierens in eMed und die Entscheidungen beim Customizing führen dazu, dass der OP-Plan für alle Spieler*innen des Planungsspiels sichtbar ist. Doch enthält der Plan neben den verpflichtenden Daten, die für die Abrechnung notwendig sind, nicht genug medizinische Daten, um ihn auch zu beurteilen. Die Akteur*innen in der allgemeinen Chirurgie koordinieren sich intern über einen Plan, der für Außenstehende unverständlich ist.

Der Wechsel vom „Zettel“ auf den OP-Auftrag schwächt Dr. Natals Kontrolle über die Informationskanäle zu anstehenden Operationen, denn durch eMed sehen alle Chirurg*innen der Abteilung auf einen Blick, welche Operationen für den Tag angemeldet sind: Die OP-Aufträge enthalten detaillierte Angaben zu jedem geplanten Eingriff. Die Plantafel in eMed OP zeigt den OP-Plan in Kalenderansicht, die geschätzte Operationsdauer und auch, welche Operationen schon begonnen haben und welche noch ausstehen. Diese Planungsentscheidungen basieren auf professionellem Fachwissen, das in eMed nicht enthalten ist. Die Chirurg*innen der Abteilung besitzen jedoch solches Fachwissen. Mit den Informationen aus eMed könnten sie die Entscheidungen des Planers Dr. Natal darüber, welche Operationen priorisiert und zu welchen Zeiten sie eingeplant werden, in Frage stellen. Da jedoch nur Pflichtfelder ausgefüllt werden, wird eine solche Kritik in der Praxis verhindert. Die Teamleiter*innen und der Planer können diesen Umgang mit eMed durchsetzen. Dabei können sie sich auf verschiedene Bedingungen stützen, die die Arbeit in der Allgemeinen Chirurgie charakterisieren: das Wohlwollen der Chefärzt*in, die ihren leitenden Mitarbeiter*innen bei der Gestaltung der Arbeitsprozesse innerhalb der Abteilung freie Hand lässt, die medizinische Tradition, der zu Folge Chirurg*innen nur mündlich kommunizierbares Expert*innenwissen haben, mit dem sie unvorhersehbare Einzelfälle behandeln, die professionelle Hierarchie der Medizin, in der erfahrene Chirurg*innen die Entscheidungen erfahrener Chirurg*innen nicht hinterfragen, und die notorische Arbeitsüberlastung der Chirurg*innen, die als offizielle Begründung gilt, warum „überflüssige“ Dateneingaben verhindert werden müssen.

Auch die OP-Leitung und die Blockleiter*in der Anästhesie tragen die Praxis des Umgangs mit eMed mit, durch die die unvollständigen OP-Pläne in der Allgemeinen Chirurgie entstehen. Würden sie nur den OP-Plan zur Koordination nutzen, wären alle Informationen, die dort eingegeben werden, für alle anderen Spieler*innen sichtbar. OP-Leitung, OP-Planer und Blockleiter*in berücksichtigen diese durch eMed neu eingeführte Bedingung beim Umgang mit der Software und tauschen kritische Informationen nur mündlich aus. Die Planung in der

allgemeinen Chirurgie ist eingespielt, die Kontrolle der Ungewissheitszonen aufgeteilt zwischen Akteur*innen, die untereinander nicht konkurrieren: Über die professionelle Hierarchie der Medizin ist die Rangordnung zwischen den drei Akteur*innen klar definiert und über die organisatorische Hierarchie der Sanusklinik sind sie mit der Leitung klar getrennter Bereiche betraut. Die Planungspraxis, die in der Allgemeinen Chirurgie etabliert ist, sichert allen dreien einen Informationsvorsprung gegenüber den Mitgliedern ihrer eigenen Bereiche. Die Kooperation ist also für die OP-Leitung und die Blockleiter*in ebenso vorteilhaft wie für den OP-Planer. Alle drei nutzen die Funktionen der Software, die ihnen bei der Koordination helfen, ohne ihre Position zu gefährden, und verteidigen so die relevanten Ungewissheitszonen, die sie kontrollieren.

Auch Dr. Mosser, der OP-Koordinator, der jeden Tag entscheidet, welche Abteilung die raren zusätzlichen Plätze in OP-Sälen zugeteilt bekommt, kann sich durch eMed ein eigenes Bild vom OP-Plan der Allgemeinen Chirurgie machen. Der Verzicht auf die Stornierung von Aufträgen trifft vor allem ihn: Er kann dadurch nicht über eMed erkennen, welche Operationen tatsächlich noch anstehen und welche abgesetzt sind, ist aber auch nicht in die Kommunikation innerhalb der Abteilung eingebunden. Die Praxis, verschobene Operationen nicht zu stornieren, scheint nicht der Stabilisierung der Machtverteilung innerhalb des Spiels zu dienen. Eine Motivation dafür findet sich im Spiel um die Verteilung der abteilungsübergreifenden Ressourcen des OP-Bereichs.

Durch den Umgang mit eMed, der sich in der Allgemeinen Chirurgie etabliert hat, wird das Handlungsproblem zuverlässig gelöst; der etablierte Planungsprozess sorgt dafür, dass Menschen und Material täglich zur richtigen Zeit am richtigen Ort sind. Über das richtige was, wer, wo und wann entscheiden die medizinische Vernunft und die Umstände, beides interpretiert durch die zentralen Akteur*innen im Spiel.

Tatsächlich gibt es in der allgemeinen Chirurgie keine der Planungsfehler, über die sich Pflegekräfte und Chirurg*innen anderer Abteilungen in der Sanusklinik beschweren: Operateur*innen, die nicht erreichbar sind, wenn die schnittbereite Patient*in in Narkose liegt, leerstehende Säle, vor deren Tür mehrere OP-Teams diskutieren, wessen Operation nun Vorrang hat, oder diagnostische Untersuchungen kurz vor OP-Termin, deren Ergebnis den ganzen Tagesplan umwirft. Die Mitarbeiter*innen nutzen eMed routiniert. Trotz durchgängiger Kritik an der Langsamkeit und der überkomplexen Benutzer*innenoberfläche wird die Software als nützlich bewertet, weil sie Informationen bündelt und zentral bereitstellt: eMed ist „*ein Fixpunkt, auf den alle zugreifen können*“ (Dr. Natal). Die Transparenz, die in der Abteilung vorteilhaft ist, soll die Grenzen der Abteilung jedoch

nicht überschreiten, wozu die Planungspraxis erheblich beiträgt. Die Informationen in eMed zeigen zwar an, welche Patient*innen noch warten und welche schon operiert wurden, lassen aber keine Rückschlüsse darauf zu, wie der Übergang vom ersten in den zweiten Zustand organisiert ist. Genau diese Frage ist Teil des Handlungsproblems, das alle Akteur*innen im OP-Bereich zusammenbringt.

4.4.2 Spiele um die Steuerung des OP-Bereichs

In den abteilungsübergreifenden Spielen im OP-Bereich wird um die Verteilung der Ressourcen, die von der Sanusklinik für Operationen bereitgestellt werden gespielt. Während es sich bei den Planungsspielen in der allgemeinen Chirurgie, die im vorangehenden Abschnitt analysiert worden sind, um Routinespiele handelt, werden in diesem Abschnitt Innovationsspiele rekonstruiert. Das Handlungsproblem dieser Innovationsspiele ist die Steuerung der abteilungsübergreifenden Spiele, über die die Verteilung der Ressourcen stattfindet.

Akteur*innen generieren relevante Ungewissheitszonen entlang der Grundkonflikte des Krankenhauses: Die Ressourcenverteilung soll einerseits medizinischen Anforderungen genügen, also sicherstellen, dass keine Patient*in ohne adäquate Behandlung bleibt. Andererseits sollen die Ressourcen auf eine Weise verteilt werden, die auch ökonomischen Anforderungen genügt. Dies bedeutet, dass im OP-Bereich sparsam mit Arbeitszeit und Sachmitteln umgegangen werden soll. Dieser Grundkonflikt bildet den Hintergrund, vor dem die Akteur*innen im OP-Bereich die relevanten Ungewissheitszonen des Innovationsspiels ausdefinieren. Die erste entsteht rund um die Frage, welche Ressourcen notwendig sind, um die in der Sanusklinik anfallenden Operationen adäquat zu erledigen. Die zweite liegt im Bereich der Verfügungsgewalt über die vorhandenen Ressourcen. Die Chirurg*innen können ihren Status als Repräsentant*innen der Profession einsetzen, um sich in Auseinandersetzungen um die Kontrolle über die erste Ungewissheitszone einen Vorteil zu sichern. Für die Verwaltung, die im Steuerungsspiel von Vorstand und OP-Koordinator*in repräsentiert wird, stellt die klassische Aufgabenteilung zwischen medizinischen und administrativen Abteilungen in Krankenhäusern eine Ressource in Auseinandersetzungen um die Kontrolle der zweiten Ungewissheitszone dar. Durch die Reorganisation und die Einführung von eMed eröffnet der Vorstand ein Innovationsspiel, das die frühere alleinige Kontrolle, die Abteilungsleiter*innen über „ihre“ Säle und „ihr“ Servicepersonal ausgeübt haben, in Frage stellt. Der Vorstand, der formal die Verfügungsgewalt über alle Ressourcen hat, definiert zwei Modi, in denen über Ressourcen verfügt werden soll: Das Jahresbudget der Abteilungen, die Stellenpläne und fest

zugeordnete Säle sind mittel- und langfristig zwischen Abteilung und Vorstand auszuhandeln. Die kurzfristige Umverteilung von Sälen und Servicepersonal von einer Abteilung auf die andere obliegt der OP-Koordinator*in.[20] Das eröffnet neue Arenen und motiviert ein ganzes Ensemble neuer Spiele, in denen auch vorhergehende Konflikte neu ausgetragen werden können.

Zur Beantwortung beider Verteilungsfragen sind Informationen über das Geschehen in den Abteilungen notwendig. Die Abteilungsleiter*innen kontrollieren diese, und eMed soll das ändern. Einige der Spiele, die die Einführung von eMed innerhalb der Abteilungen nach sich zieht, wurden im vorigen Abschnitt 4.4.1 am Beispiel der Allgemeinen Chirurgie diskutiert. Andere werden hier nur angedeutet: Der Vorstand legt fest, dass alle Arbeitsprozesse und Ergebnisse in eMed dokumentiert werden müssen. Eines der Spiele um diese OP-Dokumentation wird weiter unten beschrieben. Mit Hilfe der OP-Pläne soll die OP-Koordinator*in die Übersicht über das Tagesgeschehen im OP-Bereich behalten und feststellen, welche ungenutzten Ressourcen kurzfristig wem zugeteilt werden können. Dieses Innovationsspiel ist zum Beispiel mit dem Stornierungsspiel in der Allgemeinen Chirurgie verkettet. Auch diese Verkettung wird im Folgenden genauer ausgeführt werden. Als Grundlage für Verhandlungen über das Jahresbudget und über langfristige Verschiebungen von Stellen und zugeordneten Sälen wird die OP-Dokumentation zwischen den Abteilungen genutzt. Eines der Spiele, das sich in den Abteilungen daraus ergibt, wird nun als erstes vorgestellt.

Die Abteilungsleiter*innen sind formal verantwortlich für alle Arbeitsabläufe innerhalb ihrer Abteilung. Sie delegieren die operative Leitung zwar größtenteils an die leitenden Oberärzt*innen, aber bleiben als oberste Entscheidungsinstanz auch im Arbeitsalltag präsent, indem sie die operativen Entscheidungen kommentieren oder auch spontan durch eigene Entscheidungen außer Kraft setzen, wann

[20] Die interne Organisation der pflegerischen und anästhesistischen Abteilungen, die für die Umverteilung von Sälen relevant ist, wird hier nicht betrachtet. Die Untersuchung hat gezeigt, dass der zentrale Konflikt um die Steuerung zwischen Chirurgie und Verwaltung liegt und die anderen Berufsgruppen keine eigenen Steuerungsinteressen verfolgen. Anästhesie und übergreifende Pflegeleitung greifen in der Sanusklinik nicht direkt in das Spiel ein, sondern formulieren ihre Ansprüche gegenüber dem OP-Koordinator, der diese dann gegenüber der Chirurgie vertritt. Die zweifellos existierenden Unterschiede zwischen den Interessen der Verwaltung, der Pflege, der Anästhesie und anderer im OP-Bereich tätigen Berufsgruppen stehen hinter dem Grundkonflikt Chirurgie – Verwaltung zurück. Die professionelle Hierarchie lässt es plausibel erscheinen, dass dieser Konflikt auch in anderen Krankenhäusern primär zwischen diesen beiden Gruppen ausgetragen wird; erwartet werden sollten aber klinikspezifische Unterschiede in der Frage, ob und auf welcher Seite andere Berufsgruppen beim Steuerungsspiel mitspielen.

immer es ihnen gefällt. Die Chefärzt*innen setzen durch, dass eMed für die Planung genutzt wird. Widerstand gegen direkte Befehle der Chefärzt*in wird nicht geübt – die Chirurg*innen folgen ihren Anordnungen und beschweren sich höchstens unauffällig bei gleichrangigen Kolleg*innen, wenn sie mit der Entscheidung nicht einverstanden sind. Die Chefärzt*innen zeigen den Chirurg*innen in ihrer Abteilung außerdem, dass sie für das Budget mit verantwortlich sind.

> *Dr. Lessing (leitender Oberarzt): Wir kriegen Abzüge, wenn man 'nen Patienten zu lange liegen lässt, oder zu früh entlässt. […] Das sind ja manchmal gar nicht medizinische Gründe, dass 'n Patient jetzt sieben Tage liegt oder fünf, ja? Manchmal sind's administrative Gründe […] Und dann gibt's auch die volle Summe und dann kriegt man keine Abzüge. […]* ***Der Chef legt da Wert [drauf] und spricht das in den Besprechungen häufig an, dass man da jemanden nicht sofort entlassen kann.*** *Oder geht donnerstags selber einmal Chefvisite und fragt halt, oh, die liegt seit vier Wochen hier, was kann man vielleicht optimieren. […] Das ist ja für beide Seiten gut. Für Patienten, aber auch für die wirtschaftliche Situation.*

Auch die Mitglieder der zugeordneten pflegerischen Abteilungen werden von den Chefärzt*innen angehalten, bei ihrer Arbeit an das Abteilungsbudget zu denken. Die Pflegekräfte sind zwar formal eigenständig organisiert, sie sind der Chirurgie aber sowohl durch die professionelle Ordnung der Medizin als auch durch die Organisation des Arbeitsalltags untergeordnet: Chirurg*innen sind sowohl für die Planung von Operationen verantwortlich als auch für deren Durchführung und geben daher häufig Anweisungen, die die Pflegekräfte befolgen müssen. Einerseits unterstützen Pfleger*innen die Vorgabe, dass in eMed geplant werden soll, indem sie selbst eMed für die OP-begleitende Dokumentation einsetzen und dafür darauf bestehen, dass nur Operationen durchgeführt werden, für die in den Plan eingetragene OP-Aufträge vorliegen. „Inoffizielle" Operationen, die nicht in eMed registriert werden, gibt es nicht. Spätestens die Pflegekräfte im Saal melden eine Operation an und sorgen so dafür, dass zumindest laufende und abgeschlossene Operationen in der Software angezeigt werden.[21] Andererseits ziehen sie nach Abschluss der Operationen einige der fehlenden oder falschen Daten im OP-Auftrag nach.

[21] Solche „Notfallanmeldungen" sind bei den Pflegekräften extrem unbeliebt, weil sie viel Zeit für Nacharbeit verschlingen. Die Pflegekräfte sehen die Erstellung eines vollständigen OP-Auftrags als Aufgabe der Chirurg*innen. In der Speziellen Chirurgie, in der Notfallanmeldungen regelmäßig vorkommen, ist die Nachlässigkeit der Chirurg*innen bei der Dateneingabe ein Anlass zu dauerhaftem Frust bei den Pflegekräften.

4.4.2.1 Widerstände

Die Abteilungsleiter*innen setzen die formalen Regeln zum Einsatz von eMed trotz des Widerwillens durch, den die meisten Mitarbeiter*innen der Software entgegenbringen. Allerdings befolgen die Mitarbeiter*innen die Regeln nicht in allen Details. Die Software wird zwar genutzt, um die Aufgaben zu erfüllen, die mit ihr erfüllt werden sollen. Die Daten, die eingegeben werden, sind jedoch regelmäßig unvollständig oder schlicht falsch.

Unvollständige oder falsche Planungsdaten werden nicht nur in der Allgemeinen Chirurgie sanktioniert, sondern auch in anderen chirurgischen Abteilungen (Beispiele aus einer zweiten Abteilung, der Speziellen Chirurgie, werden im nächsten Abschnitt vorgestellt). Auch in Abteilungen, in denen die OP-Säle nur gering ausgelastet sind, werden die OP-Aufträge für ausgefallene Operationen nicht immer unmittelbar storniert. Die Planer*innen verhindern so im täglichen Ablauf, dass „ihre" Säle an andere Abteilungen weitergereicht werden.

Bei der OP-begleitenden Dokumentation mit eMed werden die Regeln etwas genauer befolgt als bei der OP-Planung. Pflegekräfte und Chirurg*innen können in eMed eintragen, welche Arbeitsschritte vor und während der Operation durchgeführt worden sind, dabei den konkreten Zeitpunkt festhalten und die Arbeitsschritte inhaltlich genau beschreiben. Über die Software können bei der Operation verwendete Geräte, Medikamente und Materialien gemeinsam gespeichert und verschiedene medizinische Parameter in der Krankenakte der Patient*in dokumentiert werden. Zu all diesen Zwecken dient die Verknüpfung der unterschiedlichen Daten zum *Fall* in eMed, die in den Abschnitten 4.2.2 und 4.3 detailliert beschrieben worden ist. Ein großer Teil der medizinischen Daten ist juristisch relevant[22] oder hat Auswirkungen auf den weiteren Behandlungsverlauf der Patient*in (z. B. bei Allergien). Solche Daten werden (größtenteils) vollständig dokumentiert. Das gleiche gilt für Verbrauchsmaterialien. In diesem Fall begründen Interviewpartner*innen in der Sanusklinik die Sorgfalt bei der Dokumentation mit einem Verweis auf das Budget. Die stellvertretende OP-Leitung der Speziellen Chirurgie erzählt:

[22] Ein Beispiel dafür ist er Umgang mit bestimmten Ausrüstungsgegenständen für Operationen, die in der Sanusklinik *Zählobjekte* genannt werden: Die Pflegekräfte müssen dokumentieren können, dass bei der Operation keine Ausrüstungsgegenstände im Körper verblieben sind. Daher werden Gegenstände, bei denen dies theoretisch möglich ist, vor und nach der Operation abgezählt und die Anzahl verglichen (daher die Bezeichnung Zählobjekte). Belegbare Sorgfalt in dieser Frage hat für die Sanusklinik Priorität vor anderem: Die Programmierung einer Erweiterung von eMed, mit der die Zählung solcher Objekte vor und nach der Operation dokumentiert werden kann, hat sich im Customizing gegenüber anderen Funktionalitäten durchgesetzt, die z. B. die Planung besser an die Anforderungen der Nutzenden angepasst hätten.

„Der Materialverbrauch muss drin sein, weil die ganze Abrechnung über SAP läuft. Also wenn wir Materialien nicht eintragen, bekommen wir die auch nicht vergütet. So wurde mir das gesagt."

Die budgetrelevanten Daten in eMed sind in der Regel korrekt, weil die Mitarbeiter*innen darauf achten, dass die Abteilung dafür korrekt vergütet wird. Anders sieht es bei den Daten aus, die genutzt werden könnten, um den Verlauf oder die Inhalte der Arbeitsprozesse aus der Dokumentation zu rekonstruieren. Diese sind oft ebenso falsch wie die Daten, die bei der Planung eingegeben werden.

Dr. Mosser (OP-Koordinator): Wir haben im Augenblick die Situation, dass zwar alle operativen Tätigkeiten in SAP dokumentiert werden, also die Anmeldung, die Erstellung des OP-Auftrags, die Planung als solche, und auch die Zeiten der jeweiligen operativen Bereiche, also OP-Pflege und Operateure – mit gewissen Einschränkungen, das darf man auch nicht verschweigen. Also die einzigen Zahlen, die ich wirklich glaube, sind Naht und Schnitt. Alles andere ist gelogen. Sag ich jetzt mal rundheraus. […] Da stimmt nichts von.

Wie bei der Planung in der allgemeinen Chirurgie beeinträchtigen falsche Angaben den Alltag der Mitglieder des OP-Bereichs nicht groß: die meisten von Ihnen müssen Daten für die Dokumentation nur eingeben, aber nicht damit weiterarbeiten. Die Arbeit des OP-Koordinators Dr. Mosser wird hingegen massiv gestört: Er weiß, dass die Daten aus eMed nicht verlässlich sind, und muss sich die Informationen, die er zur Steuerung braucht, durch freundschaftliche Beziehungen zum Personal und regelmäßige Rundgänge durch die einzelne Saalcluster erarbeiten. Auf diese Weise bekommt er genug Übersicht über die Abläufe im OP-Bereich, um täglich Kleinigkeiten zu verändern, gegen die Abteilungsleiter*innen nicht protestieren. Seine übergeordnete Position in der Hierarchie kann er jedoch nicht einsetzen, da er davon ausgeht, *„dass diese Abteilungsleiter natürlich dann versuchen würden, meine Entscheidung zu konterkarieren"*. Stattdessen setzt er darauf, dass sich *„über irgendwelche Argumente, oder auch durch, sagen wir mal, zartes Schaffen von Fakten"* langfristig die Regeln zum Umgang mit eMed vollständig durchsetzen werden. Die hier beschriebenen Innovationsspiele werden über den Verlauf der Zeit, in der ich den Umgang mit eMed in der Sanusklinik untersucht habe, nicht zu Routinespielen. Die Frage, wie die Ressourcenverteilung gesteuert werden soll, bleibt umkämpft.

4.4.2.2 Verteilung der Kontrolle in den Spielen mit eMed

In den Spielen um die Steuerung des OP-Bereichs sind Strategien von Vorstand und Abteilungsleiter*innen erkennbar, die auf die Modelle des Organisierens

von eMed und die dahinterliegenden Kontrollversuche treffen. Das Krankenhausinformationssystem eMed basiert auf der Annahme, dass das Geschehen in der Organisation mit Hilfe der Daten abgebildet werden kann, die während der Arbeitsprozesse von Nutzer*innen in die Software eingegeben oder in der Software automatisch generiert werden. Nutzer*innen, die Einsicht in die Daten haben, die in der zentralen Datenbank von eMed gespeichert sind, sollen dadurch in die Lage versetzt werden, Unsicherheiten im Krankenhaus zu kontrollieren. Durch Entscheidungen im Customizing werden über den Umgang mit eMed vor allem die Aspekte der Arbeitsprozesse im eben genannten Sinn transparent gemacht, die in den Modellen des Organisierens aus dem ERP-Kern und in denen von IS-H enthalten sind.

Alle Daten, die für die Berechnung der Fallpauschale nach gesetzlichen Vorgaben notwendig sind,[23] müssen beim Umgang mit eMed standardmäßig im frühestmöglichen Prozessschritt verpflichtend eingegeben werden, in dem sie den Nutzer*innen bekannt sind. Wenn Nutzer*innen die Pflichteingaben vermeiden, wird der OP-Auftrag vom System nicht angenommen; die Operation kann nicht in den OP-Plan eingetragen und damit auch nicht dokumentiert werden. Anders verhält es sich bei der Dauer der Operation, die bei der OP-Planung angegeben werden soll. Diese Angabe wird nicht in die Dokumentation übernommen. Wenn die Operation abgeschlossen ist, erscheint sie in keinem der Berichte, mit denen das Geschehen in der Sanusklinik für das Management abgebildet werden soll. Die einst eingeplante OP-Zeit bleibt zwar irgendwo als Eintrag in der zentralen Datenbank von eMed erhalten, wird jedoch beim Umgang mit der Software nicht erneut bedeutsam.

Auch bei der OP-begleitenden und der nachgelagerten Dokumentation ist über eMed vorgegeben, dass alle Daten, die für die Abrechnung einer Behandlung mit der Krankenkasse notwendig sind, in einem festgelegten Format eingegeben werden müssen, damit das Dokumentierte überhaupt im System gespeichert werden kann. Die Datenstrukturen und Prozesse, die die Sammlung abrechnungsrelevanter Daten sicherstellen, sind in IS-H implementiert. Hier wird festgelegt, dass der *Fall* das sogenannte „führende Objekt“ ist, also das Datenelement, an das alle anderen geknüpft werden. Die Modellierung der Prozesse um den Fall ist dort jedoch nicht neu angelegt, sondern orientiert sich an den Produktionsprozessen im ERP-Kern. Der zentrale Geschäftsprozess im ERP-Kern dient der Herstellung

[23] Diagnose und zu erbringende Leistungen werden aus im System hinterlegten Katalogen ausgewählt. Die Kataloge enthalten die von den Gremien des Gesundheitssystems erstellten ICD (Diagnosen) und OPS-(Prozeduren)-Codelisten und Zusatzinformationen (z. B. zur Beatmungszeit), die Grundlage der Berechnung der Fallgruppe bilden.

eines Produkts durch die Kombination von Materialien. Durch die Veränderungen, die bei der Entwicklung von IS-H mit diesem Geschäftsprozess und den Elementen der Software, die in ihm kombiniert wird, vorgenommen werden, dient er in IS-H der Bearbeitung eines Falls durch die Kombination von Leistungen. Die Richtlinien des deutschen DRG-Systems geben die Fallorientierung vor und legen auch fest, welche Daten notwendig sind, damit ein Fall eindeutig einer Diagnosis Related Group, also einer diagnoseabhängigen Fallgruppe, zugeordnet werden kann. Diese Zuordnung zu einer Fallgruppe ist Voraussetzung für die Abrechnung. Nur wenn die Abrechnung möglich ist, kann ein Fall nach den Kriterien in eMed erfolgreich bearbeitet werden. Die Richtlinien des deutschen DRG-Systems und die daraus abgeleiteten Anforderungen an die Dateneingabe sind so in die Arbeitsprozesse integriert, dass unvollständige oder fehlerhafte Eingaben, die sich auf die Abrechnung auswirken, spätestens in dem Arbeitsschritt korrigiert werden, in dem die Operation nach ihrer Beendigung dokumentiert wird. Die Eingabe von Daten, die nach diesen Richtlinien nicht relevant für die Abrechnung sind, ist nicht verpflichtend. Verschiedene Arten von Daten werden in der Software unterschiedlich behandelt: Einige, wie zum Beispiel die Leistungsdaten, sind einfach aufzufinden und werden in Berichten aufbereitet. Andere, wie zum Beispiel die geplanten OP-Zeiten, verschwinden nach kurzer Zeit in der Datenbank und verlieren bald nach der Eingabe ihre Bedeutung für die Prozesse in der Sanusklinik.

Die Verfolgung von Verbrauchsmaterialien vom Einkauf bis zur Rechnungsstellung durch spezielle Berichte wird in der Standardversion des ERP-Kerns unterstützt. Zusammen mit den abrechnungsrelevanten Daten bilden die Daten zu diesen Materialien die Grundlage der Verhandlungen zwischen Vorstand, OP-Koordinator und Abteilungsleiter*innen über die mittel- und langfristige Steuerung des OP-Bereichs. Die Entscheidung, Daten aus dem Krankenhausinformationssystem als Grundlage für diese Verhandlungen zu nutzen, lässt sich als eine Strategie des Vorstands in den Innovationsspielen des OP-Bereichs betrachten. Die Abteilungsleiter*innen sollen zur Preisgabe von Informationen über die Arbeitsprozesse bewegt werden, indem ihr Abteilungsbudget von den Erlösen der Abteilung und den Kosten abhängig gemacht wird, die der Abteilung eindeutig zugerechnet werden können. Die Daten aus eMed bilden die einzig legitime Grundlage der Budgetverhandlungen: Jede Abteilungsleiter*in, die die Richtigkeit dieser Daten in Zweifel zöge, würde sich damit dem Verdacht aussetzen, sie sorge in ihrer eigenen Abteilung nicht ausreichend dafür, dass eMed so genutzt wird, wie es die Organisationsregeln vorgeben.

Daten aus eMed können nicht nur Aufschluss über finanzielle Aspekte der Arbeit wie Erlöse und Kosten geben, sondern auch über Arbeitsabläufe und –inhalte. Wenn Mitarbeiter*innen aller Abteilungen vollständige, nicht gefälschte und auf die gleiche Weise interpretierbare Daten zu Planung und Dokumentation in eMed eingeben würden, ließen sich die zwölf chirurgischen Abteilungen in allen Aspekten vergleichen, die von diesen Daten abgedeckt werden. Die professionelle Deutungshoheit über medizinische Aspekte der Krankenbehandlung, die Abteilungsleiter*innen gegenüber den Vorstandsmitgliedern besitzen, bliebe zwar prinzipiell bestehen. Ihre Kontrolle über das Geschehen in ihren Abteilungen wäre aber dahin: Die Daten aus eMed würden es den Vorstandsmitgliedern (und anderen) ermöglichen, sich über dieses Geschehen zu informieren, ohne dass die Abteilungsleiter*innen dies beeinflussen könnten. Informationen über das Geschehen in den Abteilungen wären für Außenstehende nicht mehr unsicher, die Abteilungsleiter*innen könnten also diese relevante Ungewissheitszone in den Spielen um die Steuerung des OP-Bereichs nicht mehr kontrollieren. Wenn sich im Vergleich der Arbeitsprozesse einzelner Abteilungen auf Basis der Daten aus eMed Muster zeigen würden (was wahrscheinlich ist, da viele Eingriffe und Arbeitsschritte sich wiederholen), wäre es schwer für die Abteilungsleiter*innen, sich einheitlichen Regeln für die Arbeitsprozesse im gesamten OP-Bereich zu widersetzen. Ein Teil der Kontrolle über die Arbeitsprozesse innerhalb der Abteilung würde den Abteilungsleiter*innen entzogen und zum Gegenstand der Aushandlung in anderen Handlungssystemen werden, in denen sie weniger starke Machtpositionen einnehmen.

Die Strategie des Vorstands, über das Budget den Einsatz von eMed zu motivieren und damit verlässliche Informationen für die OP-Koordinator*in zu generieren, geht aber nicht auf, da es den Abteilungsleiter*innen gelingt, die Widersprüche zwischen den Modellen des Organisierens von IS-H und i.s.h.med auszunutzen. Diese Widersprüche entstehen vor allem dadurch, dass die Unsicherheiten der Krankenbehandlung in den Modellen von IS-H eher für die Zwecke des Rechnungswesens kontrolliert werden, in den Modellen von i.s.h.med aber vor allem die Koordination der Akteur*innen zum Zweck der medizinisch adäquaten Behandlung im Vordergrund stehen. Dass die Abteilungsleiter*innen diese Widersprüche nutzen, zeigt sich daran, welche formalen Regeln zum Umgang mit eMed von ihnen gestützt und welche ignoriert werden.

Die prinzipielle Annahme, auf der eMed basiert, bringt mit sich, dass DRG-relevante Daten für alle Nutzer*innen transparent werden, die Zugang zur Datenbasis der Software haben. Technisch wäre es möglich, Operationen auch ohne Angabe der Verbrauchsmaterialien zu dokumentieren: Im Gegensatz zu anderen Elementen der Dokumentation wie zum Beispiel den Leistungen handelt

es sich bei den Verbrauchsmaterialien nicht um Pflichtangaben, deren Eingabe beim Umgang mit der Software automatisch überprüft wird. Der Verzicht auf die Eingabe der Verbrauchsmaterialien hätte für die Abteilung aber finanzielle Nachteile: Die Abteilungen als für die „Produktion" verantwortliche Organisationseinheiten bestellen Verbrauchsmaterialien intern über eMed. Damit können Materialkosten automatisch Abteilungen zugerechnet werden – was für die Budgetberechnung auch genutzt wird. Abteilungsleiter*innen sorgen mit ihrem Einfluss dafür, dass ihre Mitarbeiter*innen jene Daten vollständig und korrekt eingeben, die für die Berechnung des Abteilungsbudgets relevant sind, und arbeiten in dieser Hinsicht dem Vorstand in die Hände. Diese Kooperationsbereitschaft mit den Zielen des Vorstands beschränkt sich aber auf Daten, die aus der eMed-Datenbank einfach zu extrahieren sind, weil Standardberichte dafür existieren. Solche Berichte gibt es nur für Daten, die entweder den für einen industriellen Produktionsprozess relevanten entsprechen oder relevant für die Zuordnung eines Falls zu einer Fallgruppe sind und daher bei der Entwicklung von IS-H neu für die Organisationen im Gesundheitswesen modelliert wurden.

Vorgeschrieben ist auch die Dokumentation der Prozesszeiten, also die Angabe, welche Mitarbeiter*innen wann welche Teilaufgaben erledigt haben. Diese Daten sind eigentlich für die Kostenrechnung relevant und müssten damit theoretisch auch eine Rolle für die Bestimmung von Abteilungsbudgets spielen, in welchen die in der Abteilung verursachten Kosten berücksichtigt werden. Sie können in eMed auch eingegeben werden. Mit eMed ist es aber schwierig, Prozesszeiten systematisch auszuwerten.

> *Dr. Mosser (OP-Koordinator): Hab' ich ja auch schon gesagt, dass das manchmal schwierig ist hier, fallbezogene Kostenrechnung […] und natürlich auch die zeitliche Bindung des Personals, das ist ja ein ganz wesentlicher Faktor. Wir sind hier aber nicht so weit. […] Und natürlich ist auch die Zuordnung von Prozesszeiten nicht möglich […] So dass [die Sanusklinik] sich quasi 'ne Extra-Firma leistet, die die Daten aus SAP aufarbeitet, um sie dann ins [Drittsystem] zu stellen. Das ist so 'n mehr oder weniger webbasiertes Tool, was sämtliche Zahlen, egal ob jetzt Verwaltung, Finanzen, Prozesse, in so ein WebTool packt, das man hier unterschiedlich […] benutzt, um zu sehen, welche Zahlen was machen.*

Die Daten aus eMed werden von Mitarbeiter*innen der Controllingabteilung erhoben und unter anderem als Entscheidungsgrundlage für den Vorstand aufbereitet. Auch diese Mitarbeiter*innen sind (wie der OP-Koordinator Dr. Mosser) dem langfristigen Ziel verpflichtet, die Arbeitsprozesse im OP-Bereich effizienter steuerbar zu machen. Die Daten der Standardberichte aus eMed bilden diese Arbeitsprozesse zwar nur unvollständig ab, doch das IT-Projekt, das passendere

Berichte (die z. B. Prozesszeiten beinhalten) und damit eine genauere Übersicht über die Kosten erlauben würde, wird erstmal auf die lange Bank geschoben.

> *Dr. Mosser (OP-Koordinator): Das Controlling will's auch nicht. Die wollen zwar vordergründig Zahlen wissen, wissen aber, dass das mit so viel Arbeit verbunden ist für die. Die sind auch nicht so ausgestattet, dass die das bei ihnen erledigen können, und sind da deswegen eher so: Ja, hm, hm, mal kucken.*

Fehler oder Lücken in der OP-Dokumentation werden in den Spielen um die Ressourcenverteilung damit nur zum Problem, wenn es sich um Daten handelt, die von eMed verpflichtend abgefragt und in Standardberichten zusammengefasst und ausgegeben werden. Die relevante Ungewissheitszone „Informationen über das Geschehen in den Abteilungen" wird dadurch entscheidend verändert: Erlöse und Kosten nach der Definition von eMed werden für alle sichtbar, die Zugang zu den Berichten von eMed haben. Sie sind damit nicht mehr ungewiss in den Spielen um die Steuerung des OP-Bereichs. Über den Rest weiß nur Bescheid, wer vor Ort ist und das Geschehen in der Abteilung unmittelbar beobachten kann – zum Beispiel Dr. Mosser, der OP-Koordinator. Das so erworbene Wissen bleibt aber ein anekdotisches, das jederzeit bestritten werden kann. Dies tun die Abteilungsleiter*innen auch, wenn der OP-Koordinator sie mit Regelverletzungen beim Umgang mit eMed, mit offensichtlichen Planungsfehlern oder anderen Praktiken konfrontiert, mit denen sie verhindern, dass im OP-Bereich sorgsamer mit den knappen Ressourcen umgegangen wird:

> *Dr. Mosser (OP-Koordinator): So was kann man nur aufbrechen, indem man möglichst exakt dokumentiert und auch wirklich klar zeigen kann: Letzten Dienstag, letzten Donnerstag und die drei Wochen vorher hast du bei 15 oder bei 25 oder bei 35 Fällen immer das und das und das und das gemacht. […] Wenn ich solche Dinge anspreche, […] dann heißt es: Das stimmt überhaupt nicht! Das war unvorhersehbar. […] Also es gibt tausend Gründe, warum das im Zweifelsfall nicht stimmt.*

Der OP-Koordinator kann sein Wissen um das Geschehen im OP-Bereich anders als geplant in den Spielen um die Ressourcenverteilung nicht einsetzen, weil eMed ihm nicht die Daten liefert, mit denen er seine Beobachtungen belegen könnte.

Vorstand, Abteilungsleiter*innen und Mitarbeiter*innen im Controlling erreichen mit der beobachteten Nutzung (korrekte Dokumentation von Operationen in den leicht aus eMed extrahierbaren Datenfeldern), die von Abteilungsleiter*innen bei ihren Untergebenen durchgesetzt wird, einen temporären Kompromiss: Die Abteilungen erhalten ein möglichst hohes Budget und behalten einen großen Teil

der Kontrolle über die abteilungsinternen Abläufe. Der Vorstand erhält eine vollständigere Dokumentation der Operationen und kann dadurch höhere Erlöse bei den Kostenträger*innen erzielen. Eines von zwei Zielen der Einführung ist damit vorerst erreicht. Die OP-Koordinator*in kann sich mit dem Wunsch nach besseren Prozessdaten zunächst nicht durchsetzen, gibt aber das Ziel nicht auf, langfristig doch noch zu den Mitarbeiter*innen durchzudringen und sie davon zu überzeugen, dass sie durch einen regelkonformen Umgang mit eMed (also insbesondere die Eingabe korrekter Daten bei OP-Planung und OP-Dokumentation) auch ihre eigenen Ziele (z. B. berechenbarere Arbeitszeiten) erreichen können.

Die Analyse der Spiele um die Steuerung der Ressourcenverteilung im OP-Bereich hilft dabei, auch die Planungspraktiken in der allgemeinen Chirurgie besser zu verstehen. Anders sieht es bei den Planungspraktiken in der Speziellen Chirurgie aus. Auch deren Planung ist für Außenstehende unvorhersehbar, doch dies kann nicht allein dadurch erklärt werden, dass die Akteur*innen in ihren Strategien für die Planungsspiele ihre Strategien für die verketteten Spiele im OP-Bereich berücksichtigen. Stattdessen wird der Umgang mit eMed bei der Planung in der Speziellen Chirurgie vor allem durch abteilungsinterne Spiele beeinflusst.

4.4.3 Planungsspiele in der Speziellen Chirurgie

Die Spezielle Chirurgie ist die größte chirurgische Abteilung der Sanusklinik. Etwa fünfzig Chirurg*innen mehrerer verwandter Fachgebiete sowie dreißig OP-Pflegekräfte arbeiten in sechs Operationssälen, die räumlich abgeschlossen einen eigenen Bereich bilden. Wie die Allgemeine Chirurgie hat auch die Spezielle Chirurgie eine sehr hohe Notfallquote. Während sich aber Mitarbeiter*innen aus der Allgemeinen Chirurgie bereitwillig am Customizing von eMed beteiligt und ihren Einfluss genutzt haben, um die Prozessmodelle in der Software so weit wie möglich an die in ihrer Abteilung etablierten Planungsprozesse anzupassen, wird der Umgang mit eMed in der speziellen Chirurgie vor allem als Verwaltungstätigkeit betrachtet (vgl. den Einstieg von 4.4). Die Mitarbeiter*innen der Speziellen Chirurgie erreichen bei der OP-Planung mit eMed nicht annähernd die gelassene Routine, die in der Allgemeinen Chirurgie vorherrscht. Nicht nur für den OP-Koordinator ist der OP-Plan der Speziellen Chirurgie unverständlich. Auch innerhalb der Abteilung können die Mitarbeiter*innen den Plan kaum zur Übersicht nutzen. Dabei scheint gerade die Planungspraxis dieser Abteilung auf den ersten Blick darauf ausgelegt, allen Mitarbeiter*innen möglichst viel Einblick zu ermöglichen.

In der speziellen Chirurgie erstellen Chirurg*innen die OP-Aufträge oder beauftragen eine Sekretär*in damit, sie anzulegen und die Pflichtfelder vorerst auf Basis der Daten aus der Patient*innenakte auszufüllen. Aus den anstehenden Aufträgen stellen die Dienst habenden Chirurg*innen den OP-Plan des Folgetages gemeinsam in der morgendlichen Konferenz zusammen. Nach der Konferenz wird dieser wieder einer Sekretär*in übergeben, welche ihn (und alle Änderungen und Ergänzungen der OP-Aufträge, die im Sekretariat erstellt worden sind) in eMed überträgt. Durch dieses Vorgehen wissen alle bereits am Vortag, welche Operationen wann und wo anstehen.

Anders als in der allgemeinen Chirurgie wird die relevante Ungewissheitszone „alle anstehenden Operationen" nicht von einer zentralen Planer*in kontrolliert; die Planung wird kollegial ausgehandelt. Auch für die Entscheidung, wer bei welchem Eingriff als Operateur*in eingeteilt wird, gibt es keine klar definierten Verantwortlichkeiten. In eMed werden die Operateur*innen direkt benannt, nicht nur die Teams, in denen sie arbeiten. Da „Operateur" ein Pflichtfeld ist, muss bei der Auftragserstellung eine konkrete Person zugeordnet sein. Ob diese Zuordnung bestehen bleibt, ist Teil der Entscheidungen über den OP-Plan, die in der Morgenbesprechung getroffen werden.

Die leitende Oberärzt*in ist zwar auch in der Speziellen Chirurgie formal für die Planung, die Absprachen mit den Leitungen von Pflege und Anästhesie und die Koordination der Notfälle verantwortlich. Anders als der Planer der Allgemeinen Chirurgie widmet sie sich dieser Aufgabe jedoch nicht systematisch. Bei Notfällen organisiert sie zwar Anästhesist*innen für ungeplante Operationen, doch mit den Pflegekräften wird nur sporadisch über solche Veränderungen im Plan gesprochen. Mit dem Einfügen des OP-Auftrags in den Tagesplan ist die Pflege ausreichend informiert, so ihre Haltung. Die stellvertretende OP-Leitung der zugeordneten pflegerischen Abteilung ist anderer Meinung.

> *Frau Petschel (stellv. OP-Leitung der Speziellen Chirurgie): Der organisierende Anästhesist informiert mich. Oder eine Kollegin aus dem Saal ruft mich an: Ob ich schon gesehen hätte, dass dieser Patient weg ist, dafür ein neuer. Und selten ruft der leitende Oberarzt an, so wie es sein soll. […] Die Anrufe kommen leider nicht zuverlässig und zeitnah. […] Dann hab' ich nur Stress und muss umorganisieren und dafür sorgen, dass das Personal auch diese OP kann.*

In Fällen, in denen Patient*innen ohne OP-Auftrag im OP-Bereich eintreffen und spontan operiert werden sollen, erfüllen die Pflegekräfte ihre Verpflichtung, alle Operationen in eMed zu dokumentieren, indem sie sie über *Notfallanmeldungen* in der Software eintragen. Diese Funktion erlaubt es, Operationen ohne vorherigen OP-Auftrag in den Tagesplan einzufügen und sofort zu dokumentieren. Sie

ist für die seltenen Fälle eingerichtet, in denen Patient*innen direkt nach der Einlieferung ins Krankenhaus operiert werden müssen und verlangt umfangreiche Nacharbeiten, bei denen die fehlenden administrativen und medizinischen Daten an den richtigen Stellen in der Software eingefügt werden müssen. In der Speziellen Chirurgie müssen die Pflegekräfte täglich mehrere Eingriffe als Notfälle in eMed anmelden, was entsprechend viel Arbeitsaufwand nach sich zieht und auch zu Fehlern in der Dokumentation führt: Die Pflegekräfte, die zu Beginn der Operation erst Anmeldungen erstellen müssen, vergessen häufig Einzelheiten, die sie dokumentieren müssten. Das regelmäßige Nacharbeiten bindet überdies Personal an den Bildschirm, das eigentlich im Operationssaal gebraucht würde.

4.4.3.1 Widerstände

Einerseits gilt der OP-Plan in eMed sowohl in der Pflege als auch in der Chirurgie als die gemeinsame Basis für den Informationsaustausch zwischen den Mitarbeiter*innen der beiden Berufsgruppen. Andererseits verhindern gerade die Chirurg*innen, dass der OP-Plan für einen erfolgreichen Informationsaustausch genutzt werden kann, weil sie systematisch fehlerhafte Eingaben machen. Unvollständige Leistungen und irreführende Angaben zu Operateur*innen sind selbst in den Aufträgen für regulär geplante Operationen alltäglich.

Die Abteilungsleiter*in setzt beispielsweise kurz vor geplantem Operationsstart noch diagnostische Untersuchungen an, entscheidet, dass Chirurg*innen andere als die geplanten Eingriffe durchführen sollen oder setzt kurzfristig eigene Operationen auf den Plan.

Ihre Mitarbeiter*innen tun es ihr gleich: oft entscheiden sie laut eigener Aussage erst kurz vor OP-Beginn, wie umfangreich der Eingriff werden wird. Die OP-Aufträge sind da schon lange erstellt und in den Plan eingetragen – unvollständig. Viele Chirurg*innen der Abteilung setzen auf die Koordination auf Zuruf:

> *Frau Petschel (stellv. OP-Leitung der Speziellen Chirurgie): Bei ganz vielen Patienten erfahre ich dann von meinen Kollegen, oh. Meistens, wenn die die Chirurgen sehen, besprechen die schon: Bei der nächsten OP, gibt's was Besonderes? Und dann heißt es: Ja, da müssen wir noch dies und das machen. […] Wir machen auch Team Time-Out.*[24] *Und oft wird dann erst festgestellt, bevor Schnitt gemacht wird, was noch genau geplant ist bei dem Patienten. Im Team Time-Out. Und das ist zu spät.*

[24] Im Team Time-Out teilen alle Mitglieder des OP-Teams einander mit, was sie über die geplante Operation wissen. Dabei werden Informationen über die Patient*in, ihren Zustand, Allergien und Vorerkrankungen etc. abgeglichen. Diese Maßnahme soll verhindern, dass der falsche Eingriff durchgeführt, die falsche Seite geöffnet wird oder Medikamente verabreicht werden, die die Patient*in bekanntermaßen nicht verträgt.

Nicht nur die zu Eingriffen angegebenen Informationen sind oft unvollständig, auch die personelle Besetzung lässt sich aus dem Plan nicht zuverlässig ablesen.

> *Frau Petschel (stellv. OP-Leitung der Speziellen Chirurgie): Und oft gibt es Probleme: Die Chirurgen sind festgelegt, welcher Operateur, welche Assistenten. Dass die nicht abrufbar sind. Nicht ans Handy gehen, oder keine Zeit haben, dass es Veränderungen gibt. […] Wenn der Patient reinfährt, werden die Chirurgen informiert. Und oft dann passiert [es] erst, dass kein Operateur zur Verfügung steht.*

Die Suche nach der für die Operation (oder den nächsten Teil der Operation) verantwortlichen Person ist häufig ein Grund für *Wartezeiten:* ein fast vollständiges OP-Team steht um eine (je nach Phase bereits narkotisierte und teiloperierte) Patient*in und wartet auf die für den nächsten Schritt verantwortliche Operateur*in. Ob und inwiefern Patient*innen dadurch (abgesehen von dem Zeitverlust) Nachteile erleiden, ist nicht bekannt und wird auch in der Sanusklinik nicht thematisiert. Sicher ist, dass Wartezeiten aus organisatorischer Perspektive problematisch sind. Die Spezielle Chirurgie, in der es häufig nicht genug Säle für anstehende Operationen gibt, verliert dadurch nicht nur wertvolle Saalkapazitäten, auch die Motivation der Mitarbeiter*innen leidet darunter:

> *Dr. Mosser (OP-Koordinator): Weil das natürlich, wenn das über Jahre so geht, bei den Mitarbeitern auch engramiert ist. Ja? Hier wird im Prinzip mit unserer Arbeitskraft Schindluder getrieben, also wir werden nicht als wichtig wahrgenommen und es spielt überhaupt keine Rolle. Hauptsache, es zentriert sich um die Sonne.*

Pflegekräfte, Anästhesist*innen und auch Chirurg*innen empfinden die schlechte Planung als Zeichen der Missachtung. Die „Sonne“, die der OP-Koordinator anspricht, ist die Abteilungsleiter*in der Speziellen Chirurgie. Ihre Eingriffe in den Plan führen regelmäßig zu Wartezeiten. Da sie selbst die OP-Pläne nicht respektiert und auch bei ihren Mitarbeiter*innen kein anderes Verhalten durchsetzt, wird sie für die schlechte Planung verantwortlich gemacht. Pflegekräfte haben wenig Einfluss auf die Planungsprozesse, aber üben sporadisch passiven Widerstand: Es kommt vor, dass die einzigen Pflegekräfte, die eine lange und komplexe Operation unterstützen könnten, zum geplanten Startzeitpunkt gerade zu ihrer Mittagspause aufgebrochen sind, so dass die Operation auf den nächsten Tag verschoben werden muss. Als Begründung dafür, dass die OP-Leitung solche Pausen erlaubt, kann dann die generelle Unzuverlässigkeit des Plans herhalten.

Die Koordination in der speziellen Chirurgie funktioniert also mehr schlecht als recht: Es wird zwar operiert, aber von einer effizienten Steuerung der Arbeitsprozesse, durch die Patient*innen eine adäquate Behandlung erhalten und bei der gleichzeitig sorgsam mit den Ressourcen des Krankenhauses (inklusive

der Motivation der Mitarbeiter*innen) umgegangen wird, ist die Abteilung weit entfernt. Den Mitarbeiter*innen aller Berufsgruppen gelingt es nicht, die für die Lösung des Handlungsproblems notwendigen Informationen rechtzeitig unter allen Personen zu verbreiten, die sie brauchen. Dies stört nicht nur den OP-Koordinator, sondern – anders als in der Allgemeinen Chirurgie – vor allem die Mitarbeiter*innen. Nicht nur für Pflegekräfte und Anästhesist*innen ist die Situation von Nachteil. Auch die Chirurg*innen trifft die Folgen der unzuverlässigen Planung regelmäßig, nämlich dann, wenn Operationen, die sie selbst durchführen wollen, kurzfristig abgesetzt werden, und sich dadurch die Behandlung ihrer eigenen Patient*innen verzögert.

4.4.3.2 Verteilung der Kontrolle im Spiel mit eMed

Das Planungsspiel mit eMed in der speziellen Chirurgie unterscheidet sich stark von dem in der allgemeinen Chirurgie. Über die Software werden fast alle Aspekte der Unsicherheit bei der OP-Planung in beiden Abteilungen auf identische Weise reduziert. Unterschiede gibt es nur bei den abteilungsspezifischen Vorgaben zu möglichen Operateur*innen im Plan: Teams sind es in der allgemeinen, Individuen in der speziellen Chirurgie. Die Modelle des Organisierens und damit auch Kontrollversuche, die über eMed vermittelt werden, sind gleich. Was sich unterscheidet, ist die Art und Weise, wie die Kontrolle in den beiden Handlungssystemen verteilt ist.

Die Abteilungsleiter*in hat durch ihre Stellung in der Hierarchie der Sanusklinik und mehr noch durch ihren professionellen Status als renommierte Chirurg*in Befehlsgewalt über die Chirurg*innen in der Speziellen Chirurgie. Für die Software interessiert sie sich ebenso wenig wie für die Planung, in die sie immer wieder unvorhersehbar eingreift. Mitarbeiter*innen arrangieren sich (im Zeitraum, der für diese Untersuchung rekonstruiert worden ist) mit diesen Eingriffen; solange die Abteilungsleiter*in nicht interveniert, sind sie frei zu tun, was sie für richtig halten. Die Abteilungsleiter*in spielt nicht direkt im Planungsspiel mit; sie kann sich darauf verlassen, dass Untergebene das Handlungsproblem der OP-Planung irgendwie lösen. Gegen die ständige Unzuverlässigkeit des Plans unternimmt sie nichts – zumindest nichts, was in der Abteilung als Versuch der Problemlösung wahrgenommen wird. Im Spiel um die Steuerung des OP-Bereichs ist der unzuverlässige OP-Plan sogar vorteilhaft: Verlässliche Informationen über das Geschehen in der speziellen Chirurgie sind nicht nur nicht aus eMed zu bekommen, sondern allgemein rar.

Obwohl die Chirurg*innen ebenfalls unter der schlechten Koordination leiden, geben sie weiterhin selbst systematisch falsche Angaben in den OP-Plan ein. Auch regelmäßige Appelle, die die OP-Leitung an die leitende Oberärzt*in

richtet, bleiben folgenlos. Die kollegiale Planung führt dazu, dass die leitende Oberärzt*in keine der Ungewissheitszonen im Spiel kontrolliert. Bei der OP-Planung fehlen systematisch Angaben über die personelle Besetzung der Operationen (weil falsche Operateur*innen angegeben werden) und das Operationsaufkommen insgesamt (weil OP-Aufträge ganz fehlen), was die Lösung des Handlungsproblems immer wieder zum Scheitern bringt.

Im Planungsspiel der Speziellen Chirurgie allein findet sich keine Erklärung für die ständigen Fehleingaben der Chirurg*innen. Doch sind diese, anders als die Pflegekräfte, bei der Planung in ein weiteres Spiel verwickelt, das höhere Gewinne verspricht als eine reibungslos verlaufende Planung. Informationen über eine einzelne Operation, die im Planungsspiel der Abteilung nur in der Gesamtschau wertvoll sind, sind Ressourcen der Chirurg*innen in den Spielen um die chirurgischen Karrieren.

4.4.4 Spiele um die chirurgischen Karrieren

Bei der OP-Planung werden nicht nur Patient*innen disponiert, sondern auch Chirurg*innen. Die Frage der personellen Besetzung – wer operiert was – stellt ein eigenes Handlungsproblem dar, dem weit über eine reine Dienstplanung hinaus Bedeutung zukommt. Dieses Handlungsproblem ist typisch für die Chirurgie und überall mit dem Handlungsproblem der OP-Planung verbunden – zumindest dort, wo Medizin und Krankenbehandlung ähnlich wie in modernen westlichen Industriegesellschaften organisiert sind (Vogd 2004b; Egger und Wagner 1993).[25]

[25] Im Rahmen der Untersuchung ist es nicht gelungen, Aussagen von Chirurg*innen in der speziellen Chirurgie der Sanusklinik zum Karrierespiel zu bekommen. Die Spielbeschreibung stützt sich daher auf Daten zum gleichen Spiel in anderen Abteilungen der Sanusklinik und umfangreiche (qualitative) Antworten der schriftlichen Befragung aller formal Planungsberechtigten der Klinik, die nicht einer Abteilung zugeordnet werden konnten. Darüber hinaus habe ich mich vor allem an den Erkenntnissen orientiert, die die Analyse des gleichen Phänomens bei Vogd (2004b, S. 216–218) ergibt. In dieser Analyse wird gezeigt, dass es sich bei den von den Mitgliedern der Sanusklinik geschilderten Spieldynamiken um allgemein chirurgische Dynamiken und nicht um abteilungsspezifische Idiosynkrasien handelt. Dennoch ist die Rekonstruktion des Karrierespiels der speziellen Chirurgie aufgrund der Datenlage stärker eine Konstruktion, die in der Gesamtschau der verketteten Spiele den Umgang mit der Software nachvollziehbar macht, als dies bei den anderen drei Typen von Spielen der Fall ist. Die Rekonstruktion der Planungsspiele in der Allgemeinen und der Speziellen Chirurgie sowie die der Spiele um die Steuerung des OP-Bereichs basierte zu großen Teilen auf Interviews und Beobachtungen zu genau diesen Spielen.

Das Operieren bildet den Kern der beruflichen Tätigkeit in der Chirurgie. Um aufzusteigen, müssen Chirurg*innen lernen, sowohl häufig zu erwartende Eingriffe als auch komplizierte und seltene Operationen durchführen zu können. Angehende Chirurg*innen müssen eine umfangreiche Menge an Operationen in vorgeschriebenen Bereichen absolvieren, um zur Fachärzt*innenprüfung zugelassen zu werden. Da Operieren zu großen Teilen eine handwerkliche Fähigkeit ist, die habitualisiert werden muss, kann auch nach der Fachärzt*innenprüfung nur Karriere machen, wer die Gelegenheit bekommt, möglichst viele und auch möglichst komplizierte Operationen durchzuführen. Solche komplizierten Operationen sind selten und bei allen begehrt, die Interesse daran haben, sich beruflich weiterzuentwickeln. Die Sanusklinik ist eine große und renommierte Universitätsklinik, in der deutlich mehr von diesen komplizierten Operationen anfallen als in einem gewöhnlichen Krankenhaus. Passend dazu finden sich hier auch unter den Mitarbeiter*innen mehr Chirurg*innen, die sich von dieser herausfordernden Arbeitsumgebung angezogen fühlen und großes Interesse daran haben, für solche komplizierten Operationen eingeteilt zu werden. So, wie die Verfügbarkeit von chirurgischem Personal eine Ungewissheitszone im Planungsspiel darstellt, ist umgekehrt der Zugang zu den Operationen eine relevante Ungewissheitszone im Karrierespiel. In den meisten Abteilungen wird diese Zone von der OP-Planer*in kontrolliert, die damit eine Machtposition in der Abteilung einnimmt.

> *Dr. Graf (Oberarzt der Unfallchirurgie): Allgemein versuchen die Leute natürlich auch schon, den OP-Planer zu beeinflussen. Das ist schon klar. Aber ob der Einfluss Erfolg hat, weiß ich nicht. […] Und der OP-Planer weiß auch um seine Macht. Weil er für die Assistenzärzte die mächtigste Person ist. Und das heißt natürlich, mit dieser Person legt man sich nicht an. Weil das unter Umständen für sie negativ ist. Das heißt, es kann sein, sie werden nicht so berücksichtigt, wie sie vielleicht es gerne hätten. Also es ist auch Machtinstrument. Und man kann auch dann […] die Leute bestrafen indem man sagt: Ok, für die nächsten Tage wirst du jetzt nicht in die OP eingeteilt, weil du das und das sozusagen ausgefressen hast. Und zur Bestrafung kommst du nicht in den OP.*

Fragt man die Chirurg*innen der Sanusklinik,[26] wie sie selbst diese Chancen zur beruflichen Weiterentwicklung auf die Assistenzärzt*innen verteilen, wenn sie planen, behaupten sie, vor allem auf medizinische und organisatorische Anforderungen wie die nötige Expertise oder die Geschwindigkeit der Operateur*in

[26] Die Aussagen über die Selbst- und Fremdeinschätzung der Kriterien, die bei der OP-Planung angelegt werden, sind das Ergebnis einer schriftlichen Befragung aller eMed-Nutzer*innen in zwei Krankenhäusern, die über die Berechtigung verfügen, OP-Pläne in der Software zu erstellen. Eines der Krankenhäuser aus der Befragung war die Sanusklinik. Mehr zu dieser Untersuchung findet sich in Engelmann et al. (2017).

zu achten. Sie geben an, dass sie auch die Ausbildungsbedürfnisse der Kandidat*innen berücksichtigen und sich um Fairness bemühen. Fragt man jedoch Pflegekräfte und Sekretär*innen mit Planungsrechten, also Akteur*innen, die die Planung kennen, aber nicht am Karrierespiel beteiligt sind, sagen sie aus, dass Status und gute Beziehungen eine weit größere Rolle spielen, als Chirurg*innen eingestehen. Dass die personelle Besetzung bestimmter Operationen Gegenstand machtvoller Auseinandersetzungen ist, ist jedoch ein offenes Geheimnis:

> *Dr. Graf (Oberarzt der Unfallchirurgie): Es gibt natürlich auch immer dann so gewisse Gruppenbildungen. Und da findet dann schon auch, was so ein bisschen negativ ist, Günstlingswirtschaft statt. […] Es bilden sich schon da so Grüppchen, die da … Also die Gruppe unterstützt sich dahingehend, dass sie da sozusagen an die Töpfe ran wollen.*

Beim Versuch, für eine der komplizierten Operationen eingeteilt zu werden gibt es verschiedene Gewinnstrategien: Wer gute Argumente und ein gutes Verhältnis zu leitenden Oberärzt*innen oder der Abteilungsleiter*in hat, kann deren Entscheidung einfacher zu den eigenen Gunsten beeinflussen und auf diese Weise an solch einer begehrten Operation beteiligt werden. Das Wort der Abteilungsleiter*in ist in allen Abteilungen Gesetz, auch wenn nicht alle Abteilungsleiter*innen dies so offen zeigen wie jene der Speziellen Chirurgie. Oft gibt es allerdings mehrere Interessengruppen mit ähnlich starkem Einfluss in der Abteilung, die alle versuchen, die Planer*in auf ihre Seite zu ziehen. Dr. Schneider, Oberarzt in einer anderen chirurgischen Abteilung der Sanusklinik, in der die Planungsverantwortung rotiert, berichtet: Jedes Mal, wenn eine der begehrten Operationen auf dem Plan auftaucht, verbringe er viel Zeit mit Telefonaten mit verschiedenen Kolleg*innen. Diese rufen ihn an, um ihm darzulegen, warum ihre favorisierte Kandidat*in unbedingt für den Eingriff eingeteilt werden muss.

Die Frage, wer in der Speziellen Chirurgie welche konkrete Machtposition im Karrierespiel einnimmt, ist im Rahmen der Untersuchung nicht geklärt worden. Für den Umgang mit eMed sind die Details dieser Machtverteilung unter den Chirurg*innen jedoch weniger bedeutsam als die Rolle, die die Modelle des Organisierens in der Software für die Kontrolle relevanter Ungewissheitszonen spielen. Die für die Frage dieser Arbeit bedeutsame relevante Ungewissheitszone bilden die Informationen über das Operationsaufkommen. In der Speziellen Chirurgie gibt es keine Akteur*in, die diese Ungewissheitszone alleine kontrolliert. Da Informationen über Patient*innen, die einen der seltenen Eingriffe benötigen könnten, auf verschiedenen Wegen (z. B. über die verschiedenen Ambulanzen, die Spezialsprechstunden einzelner Bereiche der Abteilung oder

Konsile für nicht-chirurgische Abteilungen der Klinik) eingehen können, pflegen ehrgeizige Chirurg*innen die Beziehungen, die über die Abteilungsgrenzen hinausreichen, um frühzeitig von solchen Patient*innen zu erfahren. Sobald aber ein Eingriff auf den OP-Plan gesetzt wird, ist er für alle Akteur*innen im Spiel sichtbar; die Unsicherheit löst sich auf und die Inhaber*innen der Machtpositionen können beginnen, die Besetzung untereinander auszuhandeln. Um solche Einflussnahme zu verringern oder eigene Besetzungswünsche ohne Intervention hierarchisch höherstehender Chirurg*innen der Abteilung durchzusetzen, sind verdeckende Strategien vor allem für die Chirurg*innen interessant, die keine besondere Machtposition in der Speziellen Chirurgie besetzen. Solche Strategien basieren darauf, begehrte Operationen so lange wie möglich vor machtvolleren Akteur*innen zu verbergen.

4.4.4.1 Widerstände

Die fehlerhaften OP-Aufträge in der Speziellen Chirurgie und ihre Folgen für den Erfolg des Planungsspiels sind in 4.4.3.1 beschrieben. Vor dem Hintergrund des Karrierespiels erscheint das Verhalten, das aus Sicht der Pflegekräfte und Anästhesist*innen wie eine systematische Nachlässigkeit der Chirurg*innen bei der Erstellung von OP-Aufträgen aussah, als rationale Strategie zur Nutzung von eMed: Wer die Einflussnahme machtvoller Akteur*innen auf die Entscheidung über die personelle Besetzung bestimmter Operationen verhindern will, muss die Details dieser Operationen so lange wie möglich geheim halten. Dazu muss ein Zeitraum für die Operation eingeplant werden, ohne dass offensichtlich wird, dass es sich um einen der begehrten Eingriffe handelt. Ein anderer Grund, die Details eines solchen Eingriffs so lange wie möglich zu verbergen ist, dass derjenige, der über die Besetzung entschieden hat, eine Chirurg*in als Operateur*in eingeteilt hat, die einflussreiche Gegner*innen in der Abteilung hat. So, wie junge Chirurg*innen hoffen können, von einem guten Verhältnis zur Abteilungsleiter*in und den leitenden Oberärzt*innen zu profitieren, so müssen sie auch fürchten, dass eine dieser einflussreichen Personen ihnen im Fall eines Konflikts Steine in den Weg legen und versuchen wird zu verhindern, dass sie in Zukunft für eine der begehrten Operationen eingeteilt werden. In eMed ist es nicht möglich, Operationen einzuplanen, ohne dass der OP-Auftrag Details zum Eingriff und zur personellen Besetzung enthält. Was möglich ist, sind unvollständige Informationen über die Operation. Solche Informationen, also zum Beispiel die Angabe nur eines Teils aller Prozeduren, die eigentlich bei dem Eingriff durchgeführt werden sollen, sind typisch für die OP-Aufträge in der Speziellen Chirurgie. Sie ermöglichen es, dass die Akteur*innen im Karrierespiel verdeckende Strategien wie die benutzen, die im Folgenden beispielhaft beschrieben werden. Beide Beispiele sind

mir aus anderen chirurgischen Abteilungen berichtet worden, in denen ebenfalls kollegial geplant wird. Die erste Strategie ist die schon angedeutete Erweiterung des Umfangs der Operation: Die begehrte Operation wird mit fehlenden Informationen zu einem Zeitpunkt eingeplant, zu dem die Operation einer Konkurrent*in in einem anderen Saal laufen soll. Erst kurz vor OP-Beginn werden die Daten ergänzt, die zeigen, dass die bisher gewöhnlich aussehende Operation für andere interessant sein könnte. Eine Umbesetzung ist dann nicht mehr oder nur mit hohem Aufwand möglich. Vertreter*innen der im Spiel stärkeren Fraktion sind nicht im Dienst, an einen parallelen Operationssaal gebunden oder haben nicht mehr die Zeit, um ihren Einfluss spielen zu lassen. Die zweite Strategie ist die Angabe einer falschen Operateur*in: Wer unpopulären Kandidat*innen zu Operationen verhelfen will, trägt deren Namen erst im Nachhinein in den OP-Bericht ein und vermerkt im Plan Operateur*innen, die auf den ersten Blick nicht auffallen. Diese Strategie schafft Probleme wie die in Abschnitt 4.4.3 beschriebenen Wartezeiten auf die Operateur*in, wenn die im OP-Plan vermerkte Operateur*in nicht über das Manöver informiert ist oder es vergessen hat und nicht bereit ist, um die Ersatzkandidat*in in den Operationssaal zu schicken, wenn sie (als im Plan vermerkte Verantwortliche) von der zuständigen Pflegekraft über den baldigen Beginn der Operation informiert wird.

4.4.4.2 Verteilung der Kontrolle im Spiel mit eMed

Die spezielle Chirurgie wird von ihrer Abteilungsleiter*in dominiert. Die Besetzung von Operationen ist Teil des kollegialen Planungsprozesses. Kollegiales Entscheiden bedeutet in der Praxis aber weder, dass alle formal Beteiligten gleichberechtigt mitsprechen, noch, dass das Ergebnis allen gefällt. Die Machtverteilung und die Spielregeln sind nur weniger sichtbar als zum Beispiel in den hierarchischen Entscheidungsprozessen, die für die OP-Planung in der Allgemeinen Chirurgie charakteristisch sind. Dies gilt vor allem für Außenstehende (vgl. Vogd 2004b, S. 101–103).[27] Alle Mitglieder der Abteilung, die professionellen Status besitzen oder höhere Positionen in der formalen Hierarchie der Abteilung besetzen, können mitspielen. Die einzige Ungewissheitszone, um deren Kontrolle sich alle in den Spielen um die chirurgische Karriere bemühen können, sind die Informationen über das Operationsaufkommen. Wer sich die Chance sichern will, selbst einen besonderen Eingriff durchzuführen oder ihn anderen zuzuschanzen, muss möglichst lange geheim halten, dass so ein Eingriff ansteht.

[27] Vogd bezieht sich bei seinen Aussagen über die Machtverteilung in medizinischen Abteilungen auf Bourdieus Feldtheorie. Diese ist deutlich spezifischer in ihren theoretischen Vorannahmen als die vorliegende Arbeit, ist aber mit der hier genutzten Herangehensweise vereinbar.

Durch eMed ist Geheimhaltung schwierig geworden: Durch die größere Transparenz der Prozesse, die durch den Umgang mit der Software entsteht, verringert sich die Ungewissheit über das Operationsaufkommen bereits zu Beginn des Planungsprozesses. Dadurch können die Mitarbeiter*innen der Abteilung, deren Macht auf anderen Quellen als den Informationen über anstehende Operationen basiert (z. B. auf einer besonderen Expertise als Operateur*in, die der Abteilung Ansehen bringt) stärker als zuvor sehen und beeinflussen, was bei der OP-Planung entschieden wird. In den Karrierespielen kämpfen die Chirurg*innen gegeneinander um die Kontrolle über die relevante Ungewissheitszone „Informationen über das Operationsaufkommen", aber keine einzelne und keine Gruppe kann sie sich dauerhaft sichern. In den Spielen um die Karriere in der Speziellen Chirurgie ist und bleibt diese Kontrolle verteilt.

Die Koordination mit den Pflegekräften fällt diesen Strategien auch deswegen zum Opfer, weil eMed Planung und Dokumentation eng aneinanderknüpft. Den Pflegekräften fällt zwar auf, welche Angaben regelmäßig falsch sind, aber sie haben keinen alternativen Informationskanal, der zuverlässiger wäre als eMed: In den Karrierespielen gibt es keine zentrale Ansprechpartner*in, da jede für sich spielt und diese Konkurrenz auf die verketteten Planungsspiele durchschlägt.

4.5 Modelle des Organisierens in Organisationsspielen

Zusammenfassend lässt sich feststellen, dass sich die Prozesse rund um die OP-Planung vier Jahre nach der Einführung des Krankenhausinformationssystems im OP-Bereich der Sanusklinik teilweise so verändert haben, wie dies vom Vorstand gewünscht worden ist. Es wird aber auch sichtbar, dass die Akteur*innen in den untersuchten chirurgischen Abteilungen sich den Kontrollversuchen, die über die Software auf ihre Arbeitsprozesse ausgeübt werden, ebenso oft entziehen, wie sie ihnen nachgeben. Welche Rolle spielen nun die Modelle des Organisierens in eMed für den Umgang von Chirurg*innen mit der Software in der allgemeinen und speziellen Chirurgie der Sanusklinik?

Die Standardsoftware bringt mehr Einheitlichkeit in den OP-Bereich, vor allem bei der Dokumentation. Die Arbeitsprozesse sind stärker voneinander abhängig, es gibt mehr Daten über jeden einzelnen Arbeitsprozess und eine zentrale Darstellung von dem, was in der Organisation vor sich geht, die für einige Mitglieder der Organisation (insbesondere aus dem Vorstand) zugänglich ist. Betrachtet man die Situation im OP-Bereich der Sanusklinik nach der Einführung des Krankenhausinformationssystems genauer, zeigen sich Lücken und

Widersprüche. Der Umgang mit eMed unterscheidet sich von Abteilung zu Abteilung und auch die Software selbst ist nicht so einheitlich, wie es auf den ersten Blick scheint.

Das KIS der Sanusklinik setzt sich aus mehreren technischen Ebenen zusammen, in denen die Organisation unterschiedlich modelliert ist. In ihnen kommen unterschiedliche Vorstellungen davon zum Ausdruck, wie Arbeitsprozesse in einer Organisation am besten kontrolliert werden können. In diesen Vorstellungen finden sich Hinweise auf die sozialen Systeme, in denen die technischen Ebenen entwickelt worden sind, und auf die Tatsache, dass die Modelle aufeinander aufbauen: Der ERP-Kern ist dazu geschaffen, alle Arbeitsprozesse eines produzierenden Unternehmens über Daten abzubilden, sie miteinander zu integrieren und unterschiedliche Perspektiven darauf bereitzustellen, damit die Organisation mit Hilfe dieser Datenbasis von der Organisationsleitung zentral gesteuert werden kann. Das Modell des Organisierens im ERP-Kern ist eines von informationstechnisch vollständig integrierter Industrieproduktion. Die Organisation, die in IS-H abgebildet ist, unterschiedet sich von der des ERP-Kerns nur durch die Art ihrer Produkte: Sie kombiniert Diagnosen und Behandlungselemente zu Fällen, die zu Pauschalpreisen an Kostenträger*innen verkauft werden. Das Krankenhaus wird als „Fallfabrik“ modelliert, die jenseits der Fallproduktion keine systematischen Unterschiede zu anderen Fabriken aufweist. Eine medizinische Perspektive auf das Geschehen ist in einzelnen Elementen angelegt, aber nicht durchgängig umgesetzt. Die integrierten Arbeitsprozesse orientieren sich vor allem an der Perspektive des Rechnungswesens. Diese entspricht in Krankenhäusern jedoch nicht einfach der in produzierenden Unternehmen.

Mit Hilfe der Funktionen des ERP-Kerns ist es möglich, einen Großteil der Kosten in produzierenden Unternehmen durch die Daten zu erfassen, die im Rahmen der Arbeitsprozesse generiert werden. In IS-H werden ebenfalls viele Daten über die Arbeitsprozesse erhoben, aber die Auswahl und Struktur dieser Daten ist an den Vorgaben des DRG-Systems orientiert. In diesem wird davon ausgegangen, dass Fälle der gleichen Kategorie gleiche Ressourcen verbrauchen; die tatsächlichen Arbeitsprozesse sind irrelevant. In i.s.h.med werden die Modelle von IS-H um zusätzliche Modelle und Anpassungen an bestehenden erweitert, in denen eine medizinische Perspektive auf die Arbeitsprozesse abgebildet wird. Aus dieser medizinischen Perspektive gehören zur Repräsentation eines Krankenhauses in Software Modelle, in denen Mitarbeiter*innen mit unterschiedlichen Qualifikationen, medizinische Abteilungen mit fachspezifischen Anforderungen und andere Aspekte der Krankenbehandlung dargestellt werden, die für die Koordination der Arbeitsprozesse im Krankenhaus relevant sind, wenn die adäquate Behandlung von Patient*innen im Mittelpunkt steht. Mit i.s.h.med wird versucht,

ein Modell des Organisierens in das KIS einzuführen, in dem Arbeitsprozesse, die nach unterschiedlichen Logiken funktionieren, durch Datenaustausch problemlos miteinander koordiniert werden können. Dies gelingt jedoch nicht, da i.s.h.med zu sehr von den Strukturen von IS-H abhängig ist. Über i.s.h.med werden den Nutzer*innen vor allem zusätzliche Optionen angeboten, die den Umgang mit dem KIS komplizierter machen. Die systematische Nutzung dieser zusätzlichen Optionen in allen Arbeitsprozessen des Krankenhauses ist in der Praxis schwer durchzusetzen, weil die Modelle in i.s.h.med, in denen die medizinische Perspektive zum Ausdruck kommt, nur Einzelteile der Software betreffen. Die Möglichkeit, das Krankenhaus durchgängig aus der medizinischen Perspektive darstellen zu lassen, bietet i.s.h.med nicht. Die vollständige Integration der Organisationsdaten ist nur aus der Perspektive des Rechnungswesens möglich.

Beim Customizing in der Sanusklinik fallen die Spannungen zwischen den verschiedenen Schichten der Software nicht auf. Die Einführung des komplexen KIS ist aufwändig, der Kernkonflikt des Krankenhauses – ob medizinische Effektivität im Sinne einer adäquaten Krankenbehandlung oder ökonomische Effizienz im Sinne eines sparsamen Umgangs mit verfügbaren Ressourcen wichtiger sind – scheint schon in der Standardversion gelöst. Anpassungen und Erweiterungen der Software werden im Customizingprozess auf ein Minimum beschränkt. Den mächtigen Chefärzt*innen der chirurgischen Abteilungen wird durch abteilungsspezifische Anpassungen gezeigt, dass ihre Autonomie nicht vollständig verschwinden soll. Der Vorstand spielt ein langfristiges Spiel, das zum Teil (vorerst) scheitert, weil aus der buchhalterischen Perspektive im Krankenhaus nicht das erfasst werden kann, was in der Fabrik aus der gleichen Perspektive sichtbar würde. Das Krankenhausinformationssystem, das in der Sanusklinik nach dem Customizing zum Einsatz kommt, und die Logiken der Modelle des Organisierens, die in den einzelnen technischen Ebenen eingebettet sind, sind in Abbildung 4.8 zusammenfassend dargestellt.

Durch Beobachtung der Organisationsspiele wird deutlich, dass die Beschäftigten der Sanusklinik die Software im Arbeitsalltag systematisch anders nutzen, als es vorgesehen ist. Das Modell der informationstechnisch gesteuerten Industrieproduktion des ERP-Kerns koppelt die Praktiken der OP-Planung und -Dokumentation eng aneinander. Durch die Erweiterungen IS-H und i.s.h.med werden die Arbeitsprozesse im OP-Bereich aber nur in den Aspekten vollständig miteinander integriert, die für die standardisierte Abrechnung notwendig sind. Daten, die dafür nicht notwendig sind, können eingegeben, aber schwer überprüft und weiterverarbeitet werden. Formale Regeln der Organisation werden nur dort effektiv durchgesetzt, wo es um diese abrechnungsrelevanten Daten geht.

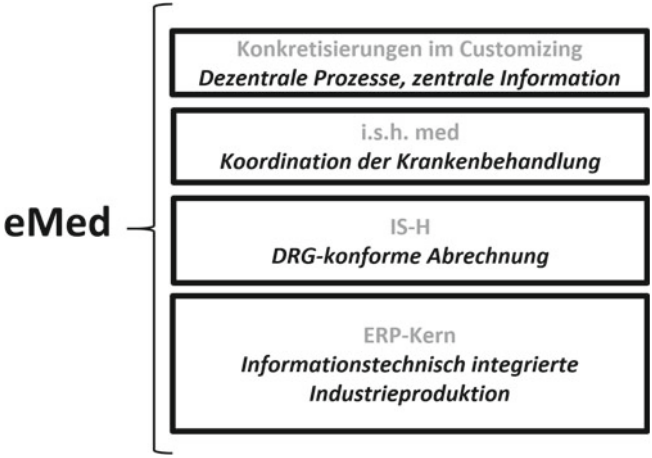

Abbildung 4.8 Modelle des Organisierens im Krankenhausinformationssystem der Sanusklinik. (Eigene Darstellung)

Die Akteur*innen in den Organisationsspielen machen sich dies zu Nutze, um die Kontrolle über relevante Ungewissheitszonen zu behalten, die durch den exklusiven Zugang zu Informationen über Patient*innen oder Arbeitsprozesse entstehen. In der Allgemeinen Chirurgie tauschen die Leiter*innen der chirurgischen, pflegerischen und anästhesistischen Organisationseinheiten, die in dieser Abteilung zusammenarbeiten, solche Informationen telefonisch aus, um zu verhindern, dass Kolleg*innen innerhalb der Abteilung ihre Arbeit bewerten oder dass sich Außenstehende in die Arbeitsorganisation in der Allgemeinen Chirurgie einmischen. In der Speziellen Chirurgie behalten die Chirurg*innen so viele Details wie möglich so lange wie möglich für sich.

Die Übersetzung der Anforderungen des Gesundheitswesens in IS-H, denen entsprechend das Krankenhaus primär aus buchhalterischer Perspektive modelliert wird, schafft die Illusion, Praktiken im OP-Bereich könnten auf Basis der abrechnungsrelevanten Daten beobachtet und gesteuert werden. Durch die Art und Weise, wie diese Perspektive in den Modellen im Krankenhausinformationssystem umgesetzt worden ist, können alle Akteur*innen formal wie vorgesehen mit dem KIS umgehen, ohne dass sie dabei dazu beitragen müssten, dass die im Customizing-Projekt vorgegebenen Ziele erreicht werden. Beim Umgang mit eMed, der sich im OP-Bereich der Sanusklinik etabliert, können Akteur*innen diese Ziele größtenteils ignorieren und in den Spielen um OP-Planung, Karriere

und Ressourcenverteilung weiterhin (wie vor der Einführung von eMed) vor allem ihre eigenen Ziele verfolgen.

Die Praktiken der OP-Planung mit dem KIS werden durch die Software ebenso geformt wie durch die Spiele. Erst die gemeinsame Betrachtung von Software und Spielen ermöglicht es, die innere Rationalität des Handelns der Akteur*innen nachzuvollziehen, also zu verstehen, warum das, was sie tun, auf Basis ihres Wissens, ihrer Ziele und der für sie verfügbaren Handlungsmöglichkeiten als rational betrachtet werden muss.

In der Gesamtbetrachtung zeigt sich an der Fallstudie, dass die Software sowohl in den einzelnen Modellierungsprozessen als auch bei der Anwendung performativ wird. Beim Umgang mit eMed versuchen die Akteur*innen, Kontrolle über die Unsicherheiten der Prozesse im Krankenhaus zu gewinnen oder die Kontrolle, die sie vor Einführung des Krankenhausinformationssystems hatten, zu verteidigen. Durch die Auseinandersetzungen mit der und um die Software wird dabei Kontrolle im Geflecht der verketten Spiele verteilt, in denen Akteur*innen mit dem KIS umgehen. Die Akteur*innen nehmen dabei die Unsicherheitsreduktionen, die in den Modellen enthalten sind, immer auch in Teilen an. Damit vertrauen sie faktisch darauf, dass die Entscheidungen, die bei der Softwareentwicklung über die Modellierung der Krankenbehandlung getroffen worden sind, (auch und zumindest teilweise) den Zwecken dienen, die die Akteur*innen selbst beim Umgang mit der Software verfolgen. Die Nutzer*innen vertrauen damit auch auf die sozialen Systeme, in denen die Modelle in eMed entstanden sind. Durch dieses Vertrauen treten sie einen Teil der Kontrolle über die Prozesse, in denen sie mit eMed umgehen, an diese sozialen Systeme ab. In der Fallstudie wird auch ein Licht auf die Grenzen der Handlungsspielräume der Akteur*innen geworfen, die Erweiterungen des ERP-Kerns entwickeln, und auf die Grenzen der Kontrolle, die sie in ihren Modellierungsprozessen über die Unsicherheiten ausüben, die bei der Nutzung des KIS relevant werden:

(1) Durch die Entscheidungen darüber, wie Unsicherheit in den Modellen der Software reduziert werden soll, die bei der Entwicklung des ERP-Kerns getroffen werden, werden alle weiteren Entwicklungsschritte bis hin zur Anwendung vorstrukturiert. Die meisten Aspekte der Unsicherheit, die die Modellierung eines Krankenhauses mit sich bringt, werden bei der Entwicklung von IS-H nicht bearbeitet. Die Entwickler*innen von IS-H vertrauen auf die Unsicherheitsreduktionen, die in den Modellen des ERP-Kerns vorgenommen worden sind. Damit geben sie Kontrolle über ihr eigenes Arbeitsprodukt (die Software IS-H) an die Entwickler*innen des ERP-Kerns ab. Ob und inwieweit diese Auslagerung von Kontrolle im Entwicklungsprozess von IS-H thematisiert worden ist, ist in der Fallstudie nicht untersucht worden. Aus der Online-Softwaredokumentation und

aus dem Umgang der Nutzer*innen mit eMed lässt sich jedoch rekonstruieren, dass die Entwickler*innen von IS-H bei der Modellierung der Arbeitsprozesse eines Krankenhauses nur einen sehr geringen Handlungsspielraum ausgenutzt haben – aus welchen Gründen auch immer.

(2) Die Entwickler*innen von i.s.h.med hingegen nutzen die Handlungsspielräume ausgiebig, die sie beim Umgang mit den Modellen aus dem ERP-Kern und aus IS-H haben, also den Modellen aus den technischen Ebenen, auf die sie ihre Erweiterung aufsetzen. Sie modellieren den Großteil der Arbeitsprozesse im Krankenhaus grundlegend neu und differenzieren ihr Modell des Organisierens damit deutlich von dem des ERP-Kerns. Sie definieren damit in vielen Einzelheiten, wie Aspekte der Unsicherheit im Krankenhausalltag reduziert werden können. Über diese Definitionen werden jedoch nur Details der Nutzung kontrolliert; für die Nutzung der Software ist es nicht notwendig, die neuen Modelle aus i.s.h.med zu übernehmen. Kontrolle über die zentralen Modellelemente, die die Abrechnung und das Berichtswesen strukturieren, verbleibt auch bei der Entwicklung von i.s.h.med in dem sozialen System, das den ERP-Kern gestaltet. Beim Umgang mit dem KIS im Krankenhaus geben die Akteur*innen über ihr Vertrauen in die Modelle zwar Kontrolle über Unsicherheiten in ihren Nutzungsprozessen an die sozialen Systeme ab, in denen die Software produziert worden ist. Welche der Unsicherheitsreduktionen, die in den Modellen enthalten sind, in jedem konkreten Fall angenommen wird, lässt sich aber nicht bei der Softwareentwicklung festlegen. Die Software kann nur Mittel von Kontroll*versuchen* sein, die Akteur*innen, die mit ihr umgehen, annehmen oder auch ablehnen können. Welche der Kontrollversuche in welchem Ausmaß angenommen werden, ist abhängig von den konkreten sozialen Bedingungen, unter denen die Software eingesetzt wird. Beim Umgang mit eMed im OP-Bereich der Sanusklinik wird der Erfolg dieser Kontrollversuche vor allem durch die professionelle Ordnung, die Organisation des Gesundheitswesens und die konkreten Machtverhältnisse in den einzelnen Abteilungen beeinflusst. Zusammenfassend lässt sich sagen, dass sich durch den Einsatz der Software Kontrolle über die Unsicherheiten in der Sanusklinik innerhalb der Organisation und auch über ihre Grenzen hinaus verschiebt, aber nicht in irgendeinem sozialen System konzentriert wird. Auch wenn die Akteur*innen eine standardisierte Organisationssoftware wie eMed nutzen, die (auch) zu dem Zweck geschaffen worden ist, die Arbeitsprozesse im Krankenhaus zentral steuerbar zu machen, bleibt Kontrolle über die relevanten Ungewissheitszonen der Krankenbehandlung verteilt.

Zur Soziologie der Software

5

Zu Beginn dieser Arbeit habe ich die Frage gestellt, wie der Umgang mit digitaler Technik unsere Gesellschaften verändert. Ich habe konstatiert, dass die Soziologie sich mehr mit dem Verhältnis zwischen dem „Digitalen“ und dem Sozialen befassen muss, wenn sie Neues zur Beantwortung dieser Frage beitragen will. Ich habe mir vorgenommen, einen Beitrag zu dieser Aufgabe zu leisten, indem ich eine soziologische Perspektive skizziere, aus der heraus verständlich wird, wie Software in das Verhältnis von sozialem Handeln und sozialer Ordnung eingebettet ist. Macht und Kontrolle sollten dabei besonders berücksichtigt werden. Diese Perspektive sollte als Grundlage für eine Vorgehensweise dienen, mit der Soziolog*innen den Umgang mit Software soziologisch untersuchen können. Dabei sollte die instrumentelle Sichtweise auf Software, die Akteur*innen bei diesem Umgang oft einnehmen, in der konstruktivistischen Sichtweise auf das Soziale, die häufig von Soziolog*innen vertreten wird und die auch ich vertrete, ausreichend Berücksichtigung finden. Als Begründung für das ganze Unterfangen habe ich mich auf C.W. Mills berufen, der es als Aufgabe von Soziolog*innen gesehen hat, mit ihrer Forschung deutlich zu machen, wie die Schwierigkeiten der Individuen („private troubles“) und die Probleme der Allgemeinheit („public issues“) zusammenhängen (Mills 2000 [1959], S. 8), und den Individuen dadurch ihre Handlungsmöglichkeiten beim Umgang mit diesen Schwierigkeiten und Problemen aufzuzeigen (a.a.O., S. 20). Am Ende der Arbeit steht die Analyse einiger Spiele, in denen eine kleine Gruppe von Mitgliedern eines einzigen Krankenhauses eine spezifisch konfigurierte Software nutzt, um ihre Arbeit zu erledigen und um die Bedingungen zu beeinflussen, unter denen in diesem Krankenhaus zusammengearbeitet wird.

D. Ametowobla, *Zur Soziologie der Software*,
https://doi.org/10.1007/978-3-658-37256-9_5

Dass die Erkenntnisse über einen kleinen Ausschnitt der Gesellschaft nicht ausreichen, um die große Frage nach der Veränderung der Gesellschaft zu beantworten, ist offensichtlich; welche Rückschlüsse von der Beobachtung dieses kleinen Ausschnitts auf das generelle Verhältnis von Software und Gesellschaft gezogen werden können, will ich in diesem abschließenden Kapitel herausarbeiten, indem ich die Grundlinien der Arbeit rekapituliere. Mit diesem Rückblick zeige ich, wie die von mir skizzierte Soziologie der Software Schwierigkeiten von Individuen (wie die durch Software beeinflussten Arbeitsbedingungen in der Sanusklinik) und Probleme der Allgemeinheit (wie den Wandel der Gesellschaft durch Digitalisierung) zueinander in Beziehung setzt und wie Soziolog*innen, die den Umgang mit Software aus dieser Perspektive untersuchen, dadurch in die Lage versetzt werden, in ihren Ergebnissen Handlungsmöglichkeiten aufzuzeigen, mit denen Schwierigkeiten und Probleme angegangen werden könnten. Dadurch soll auch deutlich werden, welchen Beitrag Fallstudien wie die, die im vorangegangenen Kapitel vorgestellt worden ist, zu dem großen Unterfangen leisten, an das ich mit meiner Arbeit anknüpfen will: die Entwicklung einer Soziologie der Software, die Neues zur Beantwortung der Frage beiträgt, wie digitale Technik die Gesellschaft verändert.

Zur Rolle von Software im Sozialen

Die Rolle digitaler Technik für die Gesellschaft wird in soziologischen Diskursen mit Hilfe unterschiedlicher theoretischer Begriffe verhandelt, deren Beziehung zueinander ungeklärt ist. In dieser Arbeit steht *Software* im Mittelpunkt. Software bringt Daten und Algorithmen zusammen, setzt sie mit Hardware in Verbindung und steuert so die Funktionen von Computern, Informationssystemen und digitalen Infrastrukturen. Gesellschaftlich wird digitale Technik relevant, weil deren Funktionen beeinflussen, wie Menschen miteinander interagieren. In technik- und organisationssoziologischer Forschung wird gezeigt, dass die Funktionen, die Software auslöst, von sozialen Ordnungsmustern abhängen, die das Geschehen während der Entwicklung der Software strukturieren, und davon, wie die an der Entwicklung Beteiligten mit diesen Mustern sozialer Ordnung konkret umgehen. Aus dieser Forschung geht auch hervor, dass die Funktionen, die bei der Softwareentwicklung definiert worden sind, den Umgang mit Software und die sozialen Ordnungsmuster der Nutzung beeinflussen, ohne sie zu determinieren. Welche Prozesse als Entwicklungs- und welche als Nutzungsprozesse gelten sollen, lässt sich bei Software jedoch schwerer als bei anderen Formen der Technik bestimmen. Die Software Studies zeigen, dass beim Umgang mit Software eine Situation entsteht, in der lokale

soziale Ordnungsmuster und die Folgen lokaler Interaktionen auf neue Weise beeinflussen, was anderswo und zu anderen Zeitpunkten getan werden kann und wie sich die Bedingungen dieses Tuns verändern.

Welche sozialen Ordnungsmuster und Formen sozialen Handelns durch Software miteinander in Beziehung gesetzt werden, lässt sich nicht allgemein bestimmen. Die Beziehungen, die durch den Umgang mit Software entstehen, hängen von den spezifischen Bedingungen ab, die bei Entwicklung, Einführung und Nutzung herrschen. Die Rolle von Software im Sozialen entsteht und verändert sich durch die jeweiligen sozialen Kontexte, die sie im Rahmen ihres Lebenszyklus durchläuft. Wer die soziale Dimension der Funktionen von Software verstehen will, muss daher in der Biographie der Software nach den Aushandlungen suchen, in denen sie ihre Ursprünge hat.

Wie soziale Ordnungsmuster und Formen sozialen Handelns über Software miteinander in Beziehung stehen, lässt sich hingegen allgemein bestimmen. Software ist Technik, die gezielt geschaffen wird, um Zwecke zu erfüllen. Insofern ist sie immer ein Instrument, mit dem Kontrolle über soziale Prozesse ausgeübt werden soll. Sie ist an Hardware gebunden, um wirksam werden zu können. Insofern ist sie immer den Bedingungen unterworfen, die Hardware setzt. Diese Bedingungen sind im Abstrakten einfache Grundprinzipien mit im Konkreten komplexen Folgen. Die abstrakten Grundprinzipien sind: Freiheit von Unsicherheit, Diskretheit und Determinismus. Die konkreten Folgen sind: Kontextabhängigkeit der Funktionen, undurchschaubare Zerlegungen der Modelle des Sozialen und rekursive Abhängigkeiten durch Wiederverwendung.

Über Software wird die Funktionslogik der Hardware mit den Funktionswünschen vermittelt, die die Gesellschaft an digitale Technik richtet, und mit den Funktionsbedingungen, die sie für den Umgang mit digitaler Technik einrichtet. Die Vermittlung geschieht über Modelle, die so miteinander verknüpft werden, dass aus den sozialen Phänomenen voller Unsicherheit, die die Vorlagen für die Modellierung liefern, die unsicherheitsfreien mathematischen Formalismen entstehen, die Hardware verarbeiten kann. Dazu sind wiederholte Modellierungsprozesse erforderlich, in denen Unsicherheit immer weiter reduziert wird. Jeder dieser Prozesse ist sozial. Entscheidungen darüber, welche Aspekte der Unsicherheit für das Modell wie reduziert werden sollen, sind in jedem dieser Prozesse nicht nur Ergebnis von, sondern auch Mittel für Kontrollversuche. Wie diese Modellierungsprozesse verlaufen hängt davon ab, in welchem Verhältnis soziale Ordnung und soziales Handeln in diesen Prozessen stehen, ebenso wie von anderen Modellen, die bei der Modellierung als Grundlage genutzt werden und in denen Entscheidungen aus vorangegangenen Prozessen abgelagert sind. Solche fertigen Modelle werden performativ, wenn sie als Grundlage für die Erstellung neuer Modelle genutzt werden: Sie beeinflussen das,

was sie beschreiben, so, dass es der Beschreibung ähnlicher wird. Darin gleichen Modellierungsprozesse jenen anderen Prozessen, in denen Software „nur“ genutzt wird: Auch solche Nutzungsprozesse werden durch den Umgang mit Software in einzelnen Aspekten den Modellen ähnlicher, in denen dieser Umgang beschrieben ist.

In welchen Aspekten und in welchem Ausmaß die Prozesse, bei denen mit Software umgegangen wird, sich an ihre in der Software modellierten Vorlagen anpassen, lässt sich weder bei der Softwareentwicklung festlegen noch an der Software ablesen. Es hängt davon ab, wie Akteur*innen Modelle, auf denen die Software beruht, durch soziales Handeln in die sozialen Ordnungsmuster integrieren, durch die der Kontext der Nutzung strukturiert wird. Software wird im Sozialen durch performative Modelle relevant, in deren Unsicherheitsreduktionen sich die verschiedenen Kontrollversuche aus dem Lebenszyklus abgelagert haben.

Zur soziologischen Untersuchung von Software

Aus dieser Antwort auf den ersten Teil der Forschungsfrage folgen drei Punkte, die bei der soziologischen Untersuchung von Software beachtet werden müssen:

(1) Entwicklung und Nutzung von Software sind soziale und sozial geordnete Prozesse, die prinzipiell zusammengedacht werden müssen, weil sie sich immer gegenseitig beeinflussen: (Vorgestellte und tatsächliche) Nutzung dient der Entwicklung als Vorlage für Modelle, Entwicklung schafft in den Modellen Bedingungen für die Nutzung.
(2) Software soll Soziales immer auf irgendeine Weise kontrollieren und diese Kontrollversuche sind niemals notwendigerweise erfolgreich: Das Ziel, Kontrolle über den Umgang mit Software auszuüben, ist Anlass für all die Modellierungen, die Software hervorbringen. Der Gegenstand, auf den sich diese Kontrollversuche richten (also die bei der Entwicklung vorgestellten Nutzungsprozesse) und die Kontexte, in denen sie beim Umgang mit Software sozial bedeutsam werden (also die tatsächlichen Nutzungsprozesse), können sich aber grundlegend unterscheiden. Selbst wenn Software in genau dem Kontext eingesetzt wird, für den sie entwickelt worden ist, bleibt der Umgang mit Software sozial und damit (aus der in dieser Arbeit eingenommenen konstruktivistischen Perspektive) mit Unsicherheiten behaftet.
(3) Mit der Entscheidung, eine bestimmte Software zum Gegenstand einer soziologischen Untersuchung zu machen, ist noch nicht festgelegt, welche Modelle und welche Umgangsformen erforscht werden müssen, um die Rolle der Software im Sozialen zu verstehen. Auch die Frage, welche Prozesse als Ursprung der

> Performativität und welche als performativ beeinflusste betrachtet werden müssen, kann nicht allein durch die Wahl der Software entschieden werden. Diese Entscheidungen lassen sich nur im Hinblick auf konkrete Forschungsinteressen treffen: Jeder Software liegen eine Vielzahl an Modellen zu Grunde, zwischen denen komplexe Abhängigkeiten bestehen. Jedes dieser Modelle ist das Ergebnis von Aushandlungen in miteinander verketteten Prozessen, in denen entschieden wird, wie Unsicherheit durch das Modell reduziert werden soll. Jede dieser Aushandlungen ist von den Bedingungen in den sozialen Systemen beeinflusst, in denen sie stattfindet. Jeder dieser Faktoren kann bedeutsam werden, wenn Akteur*innen in sozialen Systemen mit Software umgehen und dabei einzelne Modelle performativ werden. Insgesamt gibt es zu viele mögliche Antworten auf die Frage, was eine Software ausmacht und welche Folgen der Umgang mit ihr hat, als dass Forscher*innen sie in einzelnen empirischen Studien erschöpfend beantworten könnten.

Auf Basis dieser Grundsätze sind bei der soziologischen Untersuchung von Software unterschiedliche Vorgehensweisen denkbar. Ich habe mich für eine entschieden, die mit dem Konzept der Dualität von Technik von Wanda Orlikowski (1992) vereinbar ist und an bestehende technik- und organisationssoziologische Untersuchungen zu Software anschließt. In der Vorgehensweise, die ich in der vorliegenden Arbeit vorschlage, wird Software als Expert*innensystem im Sinne von Anthony Giddens (2010 [1995]) betrachtet, das auf Modellen basiert, die ebenfalls Expert*innensysteme sind. Diese Modelle sind das Ergebnis von Aushandlungen in sozialen Systemen, in denen die Praktiken und strukturellen Eigenschaften dieser Systeme zum Ausdruck kommen. Für die empirische Arbeit bedeutet die Entscheidung, Software und Modelle als Expert*innensysteme zu beobachten, folgendes: Wenn Akteur*innen mit Software und ihren Modellen umgehen, setzen sie – vielleicht nicht bewusst, aber faktisch – Vertrauen in die sozialen Systeme, aus denen die Expert*innensysteme stammen. Damit geben sie den Kontrollversuchen, die in den Modellen, die beim Umgang mit Software performativ werden, zum Ausdruck kommen, insofern nach, als sie die Unsicherheitsreduktionen akzeptieren, die diese Modelle anbieten. Performative Modelle in Software lassen sich jedoch nicht direkt beobachten, denn die Akteur*innen gehen mit einzelnen Elementen der Software um. Informationen über die sozialen Systeme, die an der Entwicklung der Software beteiligt waren, können genutzt werden, um zu rekonstruieren, welche Modelle die Elemente der Software, die sich bei der Beobachtung als bedeutsam für das Forschungsinteresse erweisen, miteinander verbinden, und welche Kontrollversuche in diesen Modellen zum Ausdruck kommen. Der Umgang mit Software wird als kollektives Handeln in strukturierten und von Machtungleichgewichten

geprägten strategischen Spielen betrachtet und mit der strategischen Organisationsanalyse nach Michel Crozier und Erhardt Friedberg (1979) untersucht. Solche Spiele sind kollektive Handlungszusammenhänge, in denen Akteur*innen ein gemeinsames Handlungsproblem bearbeiten, wobei sie mehrfache, divergierende und flexible individuelle Ziele verfolgen. Die Kontrollversuche, die in den Modellen der Software zum Ausdruck kommen, sind insofern nur Teil eines ohnedies von Kontrollversuchen geprägten sozialen Kontextes, mit dem Akteur*innen strategisch umzugehen wissen. Für die empirische Arbeit bedeutet diese Entscheidung für die strategische Organisationsanalyse, dass der beobachtete Umgang mit Software als Ergebnis individueller, von den Ordnungen der Handlungsfelder geprägter Strategien betrachtet wird, in denen Akteur*innen versuchen, ihre Handlungsspielräume zu bewahren oder auszudehnen. Inwieweit sie darin erfolgreich sind, hängt von ihrer Machtposition im Feld und somit davon ab, wie gut es ihnen gelingt, für die Spiele relevante Ungewissheiten zu kontrollieren. Dabei haben sie nicht nur die Möglichkeit, sich den Kontrollversuchen, die in den Modellen der Software zum Ausdruck kommen, vollständig zu unterwerfen oder zu verweigern. Sie können sich vor allem auch partiell so auf sie einlassen, so dass die Software ihnen als Ressource zum Erreichen ihrer Ziele und zur Verbesserung ihrer Position im Handlungsfeld dient.

Zur Fallstudie

Die Fallstudie, mit der die Arbeit abschließt, dient auf der einen Seite dazu, die vorgeschlagene Herangehensweise zu verdeutlichen und zu zeigen, wie eine von den Grundsätzen der skizzierten Soziologie der Software geleitete Forschung aussehen und welche Ergebnisse sie produzieren kann. Auf der anderen Seite stellt sie, gemäß dem in der Einleitung formulierten Anspruch an Soziologie, die Verbindung in den Mittelpunkt, die zwischen den Schwierigkeiten der Individuen und den Problemen der Allgemeinheit besteht, und zeigt, welche Rolle Software bei dieser Verbindung spielt: Die Veränderungen im deutschen Gesundheitswesen, vor allem der Wechsel von bedarfswirtschaftlicher zu fallbasierter Vergütung der Krankenbehandlung, und die Verbreitung standardisierter Organisationssoftware, die den Verantwortlichen im Krankenhaus die buchhalterische Perspektive auf Organisationen als primäre Steuerungsperspektive anbietet, beeinflusst konkret den Arbeitsalltag von Einzelnen im Krankenhaus. Im untersuchten Fall setzt die Klinikleitung sich mit ihren Versuchen, mit Hilfe der Software Kontrolle über den OP-Bereich auszuüben, teilweise durch. Teilweise scheitert sie aber auch, und zwar vor allem an etablierten Strukturen des Krankenhauses, die durch die Logik der medizinischen Profession beeinflusst sind, und den mehr oder weniger problematischen Folgen der Strategien, in denen Akteur*innen die Software einsetzen, um Ziele zu erreichen, deren Bedeutung für

die Nutzer*innen bei der Entwicklung der relevanten Modelle offensichtlich nicht hinreichend berücksichtigt worden ist.

Die in der Fallstudie untersuchte Software ist auch über das Anwendungsfeld Gesundheitswesen hinaus von gesellschaftlicher Relevanz: Standardisierte Organisationssoftware ist eine der Voraussetzungen für die Selbstverständlichkeiten des Lebens in unseren reifen Industriegesellschaften. Verteilte Produktion und komplexe Logistik wären ohne sie nicht vorstellbar. Im Alltag begegnen uns die Auswirkungen dieser Phänomene überall: Im Supermarkt, wo wir täglich frisches Obst aus allen Ecken der Welt vorfinden, im Berufsalltag, wo Produktivitätsanalysen in Konzernzentralen dazu führen, dass der eigene Arbeitsablauf komplett umgestaltet wird, oder bei der Urlaubsplanung, wo die Informationen der Fluglinien über die zum Reisezeitpunkt noch verfügbaren Plätze darüber entscheiden, was es uns kostet, unseren Urlaubsort zu erreichen. Auch der Trend, organisiertes Handeln aller Art nach Kennzahlen zu bewerten und zu steuern, der von Michel Power in „The Audit Society“ (Power 2002) als charakteristisch für unsere Gesellschaften dargestellt wurde, wäre ohne die technischen Voraussetzungen zur massenhaften Sammlung und Analyse von Daten über die Arbeitsprozesse in Organisationen unmöglich.

Die Fallstudie in dieser Arbeit baut auf dem umfangreichen soziologischen Forschungsstand zu SAP auf. Die Standardsoftware ist vermutlich die soziologisch am besten untersuchte Software überhaupt; es gibt eine Vielzahl empirischer Studien, die Entstehung, Weiterentwicklung, Einführung und Nutzung in verschiedensten Organisationen aus unterschiedlichen Perspektiven erforschen. Die in der Einleitung kritisierte Trennlinie zwischen Entwicklungs- und Nutzungsforschung durchzieht jedoch auch diesen Bereich. Zu der gesellschaftlich relevanten Frage, inwieweit die (gut erforschten) Prozesse bei der Entwicklung und Weiterentwicklung, die in wenigen, stark ungleich strukturierten Kontexten stattfinden, sich auf die (gut erforschten) Prozesse des Umgangs mit SAP in den unterschiedlichen, oft völlig anders strukturierten Kontexten auswirken, in denen SAP zum Einsatz kommt, ist bisher wenig bekannt. Mit der Fallstudie demonstriere ich insofern keine beliebige Anwendungsmöglichkeit für die Soziologie der Software, sondern eine, die relevant und dank der vielen Vorarbeiten unmittelbar umsetzbar ist.

Zu den Grenzen dieser Arbeit

Die Fallstudie dient in der vorliegenden Arbeit vor allem, um zu zeigen, wie sich die in der Theorie entwickelte soziologische Perspektive auf den Umgang mit Software für empirische Forschung operationalisieren und anwenden lässt. Methodische Überlegungen sind bei der Durchführung der Studie oft pragmatischen Erwägungen untergeordnet worden. Nicht alle Akteur*innen, die ich befragen wollte, um die im Forschungsprozess vorläufig gebildeten Hypothesen zu prüfen,

waren bereit, Interviews zu geben; nicht alle Dokumente, die für eine fundierte Analyse der Software wünschenswert gewesen wären, waren für mich verfügbar. Der fehlende Zugang zu Softwareentwickler*innen und die Tatsache, dass das Thema „persönliche Machtkalküle in der Krankenbehandlung" unter Chirurg*innen als problematisch wahrgenommen und von den meisten nur indirekt angesprochen wird, haben die Datenerhebung erschwert. Bei der Auswertung ist mehr auf die methodologische Grundhaltung geachtet worden, auf der die Grounded Theory basiert, als auf die strenge Befolgung eines der verschiedenen Codierverfahren, die von verschiedenen Anhänger*innen der Grounded Theory entwickelt worden sind. In der Studie werden daher nur die Konturen der Spiele in der Sanusklinik dargestellt. Details über den Umgang mit der Software, die Machtverteilung in den Abteilungen und die Strategien der Akteur*innen fehlen überall dort, wo auf Basis der verfügbaren Daten keine belastbaren Aussagen über diese Details getroffen werden konnten. Mit der für eine Soziologie der Software zweifellos bedeutsame Frage, mit welchen Methoden Soziolog*innen allgemein Erkenntnisse über Modelle in Software gewinnen können, wird in dieser Arbeit nur am Rande behandelt. Für den Gegenstand, den ich in meiner Fallstudie untersuche, sind ausführliche Vorarbeiten vorhanden, die ich so gut wie möglich mit Hilfe der Daten ergänzt habe, die ich selbst erheben konnte.

Die Tatsache, dass die Fallstudie stark auf solchen Vorarbeiten zur untersuchten Software aufbaut, zeigt auch, wo die Grenzen der hier vorgeschlagenen Vorgehensweise liegen: Auch wenn sie sich auf allgemeine Prinzipien des Zusammenhangs zwischen Software und Sozialem stützt, liefert sie keine Vorlage, wie Forschende einfach zu allgemeinen Aussagen darüber kommen können, welche Rolle eine beliebige Software für den Umgang mit ihr in einem beliebigen sozialen Kontext spielt. An der Skizze einer Soziologie der Software, die ich in dieser Arbeit entwickelt habe, zeigt sich, mit welcher Komplexität Forschende konfrontiert sind, wenn sie nicht nur beobachten wollen, welche Rolle Software im Sozialen spielt, sondern auch verstehen wollen, wie es dazu kommt, dass Software diese Rolle spielt, und wer wann welche Handlungsspielräume hat, um diese Rolle mitzugestalten.

Ich habe für die Rekonstruktion dieser Komplexität viele Forschungsstränge zusammengeführt, in denen Wissenschaftler*innen sich intensiv mit der Grundfrage der vorliegenden Arbeit auseinandersetzen. Einige habe ich allerdings bewusst ausgelassen. Aus der Soziologie ist dies vor allem die Arbeitssoziologie, die hier keine Erwähnung findet, und die Kultur- und Mediensoziologie, aus der nur punktuell Forschungsergebnisse aufgegriffen werden, z. B. Grenz (2014) Arbeit zu digitalen Plattformen. Der Grund dafür ist, dass die Komplexität, die ich im Zentrum der soziologischen Auseinandersetzung mit Software sehe, in beiden Bereichen eine noch geringere Rolle spielt, als in denen, die hier aufgenommen sind: In den Studien

der Arbeitssoziologie steht praktisch ausschließlich die Nutzung digitaler Technik im Mittelpunkt. Fragen nach der Gestaltung dieser Technik, die die gefürchtete Digitalisierung der Arbeit vorantreibt, werden von den meisten Arbeitssoziolog*innen ignoriert. In der Kultur- und Mediensoziologie hingegen konzentrieren sich die Forscher*innen so stark darauf, herauszufinden, was Algorithmen, Daten oder Netzwerke allgemein für Konsequenzen für das Soziale haben, dass die Frage, wie diese Konsequenzen konkret von Menschen unter sozialen Bedingungen hervorgebracht werden, meistens untergeht.

Die größeren Lücken, die ich bei der Aufarbeitung des Forschungsstands für die vorliegende Arbeit gelassen habe, liegen außerhalb meiner eigenen Disziplin: Die Rolle von Software im Sozialen wird auch in der Informatik verhandelt. Auch wenn das Verhältnis von sozialem Handeln und sozialer Ordnung für Informatiker*innen nicht generell als relevanter Forschungsgegenstand betrachtet wird, ähneln sich viele der Probleme, die durch eine soziologische Perspektive auf den Umgang mit Software verstanden werden können, und viele der Probleme, auf die Softwareentwickler*innen im Alltag und in der Forschung Lösungen suchen und finden. Ich habe für die vorliegende Arbeit vor allem auf Literatur der Informatik zurückgegriffen, die in der Ausbildung von Informatiker*innen eingesetzt wird. Im Ergebnis spiegelt meine Darstellung der Perspektive von Informatiker*innen auf die Softwareentwicklung vor allem wider, was in der Praxis der Softwareentwicklung geschieht. Inwieweit Forscher*innen in der Informatik sich der potentiell weitreichenden gesellschaftlichen Folgen der technischen Komplexität von Software bewusst sind, bleibt im Dunkeln. Der Eindruck, der bleibt, wenn man sich einen groben Überblick über die Themen der Forschung in Software Engineering, Systems Engineering und Wirtschaftsinformatik verschafft, ist der eines unbearbeiteten Feldes. Informatiker*innen nehmen den Soziolog*innen die Aufgabe nicht ab, zu erklären, welcher Zusammenhang zwischen den gesellschaftlichen Bedingungen und der Arbeit von Informatiker*innen besteht.

Zum Nutzen der Arbeit für weitergehende Forschung

Soziolog*innen, die sich ernsthaft mit Software auseinandersetzen wollen, müssen der Informatik mehr Aufmerksamkeit widmen. Mit der vorliegenden Arbeit will ich Leser*innen auch davon überzeugen, dass Soziolog*innen fundiertere Antworten auf Fragen geben können, in denen Software eine Rolle spielt, wenn sie zumindest in groben Zügen verstanden haben, was die Technik auszeichnet, die sie betrachten. Eine Soziologie der Software kann mehr zu einem Verständnis der Rolle von Software im Sozialen beitragen als bisher, wenn Soziolog*innen in der Lage sind zu identifizieren, welche Elemente der Software für ihr Funktionieren bedeutsamer und welche weniger bedeutsam sind und wenn sie eine Vorstellung von den Bedingungen

haben, unter denen diese Elemente entstanden und zusammengefügt worden sind. In dieser Arbeit stelle ich Grundprinzipien der Funktion und des Lebenszyklus von Software dar, um darauf hinzuweisen, dass diese Grundprinzipien für die Soziologie bedeutsam sind und dafür zu werben, dass es sinnvoll ist, die Komplexität von Software in soziologischer Forschung stärker als bisher zu berücksichtigen. Moderne Softwareentwicklung ist ein in der Soziologie kaum beachtetes Untersuchungsfeld. Dass Forschung über aktuelle Software sich fast ausschließlich auf Erkenntnisse zur Softwareentwicklung stützen muss, die durch Studien in den 1990er Jahren gewonnen worden sind, ist angesichts der Ausmaße, in denen sich die Praxis der Softwareentwicklung seit diesem Zeitpunkt verändert hat, mindestens problematisch. Ich hoffe, dass die vorliegende Arbeit dazu beiträgt, dass Soziolog*innen den Fokus ihrer Forschung wieder stärker auf Prozesse und Prinzipien der Softwareentwicklung richten. Ausgehend von den Darstellungen dieser Arbeit lassen sich zwei Gründe benennen, warum diese Ausweitung des Fokus für eine soziologischen Digitalisierungsforschung relevant ist.

Erstens ist Softwareentwicklung an sich ein relevanter Untersuchungsgegenstand, in dem arbeits-, organisations- und techniksoziologische Fragestellungen auf spezielle und bisher nur ansatzweise verstandene Weise zusammenspielen. Welche Auswirkungen hat die immer stärkere Verwendung von standardisierten „Softwarebausteinen“ auf die Handlungsspielräume bei der Softwareentwicklung? Inwieweit kommen in den Nutzungsdaten, die z. B. für die Weiterentwicklung digitaler Plattformen herangezogen werden, tatsächliche Anforderungen von Nutzenden zum Ausdruck? Inwieweit sind diese Nutzungsdaten Artefakte der Programmierung, technischer Trends oder vergangener und vergessener Vorgaben von Manager*innen, Geldgeber*innen oder Regulator*innen? Forschungsprojekte, in denen solche Fragen untersucht werden, lassen nicht nur soziologische Erkenntnisse erwarten, sondern könnten auch für Informatiker*innen relevant sein und diese in ihrer Praxis informieren.

Zweitens werden sich Fragen nach den Auswirkungen, die der Umgang mit konkreten, weit verbreiteten Arten von Software haben kann, nur beantworten lassen, wenn die Bedingungen, unter denen diese konkreten Artefakte entstehen und weiterentwickelt werden, besser bekannt sind. SAP ist nicht die einzige Software, die für einen großen gesellschaftlichen Bereich relevant ist. Mehr Aufmerksamkeit verdienen vor allem Standards, Protokolle und Designmodelle, die vielen Einzelanwendungen zugrunde liegen, aber für die Nutzenden nicht sichtbar sind. Studien, in denen Forschende sich mit den sozialen Bedingungen und den Beziehungen zwischen den Akteur*innen befassen, welche bei der Entwicklung solch weit verbreiteter Softwarebausteine bedeutsam werden, gibt es bisher kaum. Solche Studien können die Grundlage für andere Arbeiten bilden, in denen sich Soziolog*innen mit

der Rolle konkreter Anwendungen befassen, die aus solchen Bausteinen aufgebaut sind.

In der Soziologie fehlt es nicht an Studien zum Umgang mit Software oder am allgemeinen Austausch über die Ergebnisse; was fehlt, ist eine Grundlage dafür, dass Forschende mit unterschiedlichen Forschungsfragen zur Rolle von Software im Sozialen über Einzelstudien hinweg kooperieren, ihre Ergebnisse systematisch aufeinander beziehen und reflektieren, wo ihre jeweiligen blinden Flecken liegen. Mit einer Soziologie der Software können nicht alle blinden Flecken soziologischer Forschung erhellt werden, denn die Tatsache, dass für jede Untersuchung ein Fokus gewählt werden muss, macht blinde Flecken unvermeidlich. Aber eine Soziologie der Software kann es für Forscher*innen einfacher machen zu erkennen, wo die spezifischen blinden Flecken in einzelnen Forschungsarbeiten liegen, durch welche anderen Arbeiten diese Forschungslücken gefüllt werden können und was zu tun ist, wenn ganz bestimmte Fragen beantwortet werden sollen.

In der Softwareentwicklung wird von „technischen Schulden“ gesprochen, die angehäuft werden, wenn darauf verzichtet wird, komplexe Probleme unmittelbar zu beheben, um schneller und mit weniger unmittelbarem Aufwand zu vorerst funktionsfähigen Lösungen für aktuelle Aufgaben zu kommen (Tom et al. 2013). Der Jurist und Internetforscher Jonathan L. Zittrain beschreibt den aktuellen gesellschaftlichen Umgang mit einer bestimmten Form von Software, der künstlichen Intelligenz, als das Anhäufen von „intellektuellen Schulden“ („intellectual debt“): Obwohl wir wissen, dass die Prinzipien, nach denen sogenannte „intelligente“ Systeme operieren, nicht bekannt sind, werden Entscheidungen dennoch an solche Systeme ausgelagert, weil diese Systeme schneller und auf den ersten Blick zuverlässiger entscheiden als Menschen (Zittrain 2019). Eine Soziologie der Software muss Mechanismen wie die, die Zittrain für den Umgang mit künstlicher Intelligenz beschreibt, in den Blick nehmen und die Debatte auf die gesellschaftlichen Bedingungen und Folgen der Komplexität von Software lenken. Soziolog*innen, die die Aufgabe, die Mills ausformuliert hat, ernst nehmen, dürfen nicht selbst der Versuchung erliegen, heute weitreichende Aussagen zur Rolle digitaler Technik für die Gesellschaft zu treffen, denen die Grundlage fehlt, weil Forscher*innen die fundierte Auseinandersetzung mit der Technik auf morgen verschieben. Ich hoffe, dass meine Arbeit dazu beiträgt, dass Soziolog*innen erkennen, dass sie mit schwach fundierten Aussagen eigene „Schulden“ anhäufen, die fällig werden, wenn sich die geringe Verlässlichkeit solcher Aussagen zeigt.

Anhang

Für die Analyse der generischen Form des KIS wurde die öffentlich zugängliche Dokumentation der Module genutzt, die über die Website von SAP verfügbar ist. Die Dokumentation jedes der betrachteten Module wurde dabei als eigene Quelle ins Literaturverzeichnis aufgenommen. Dadurch soll sichtbar gemacht werden, welche im Text beschriebenen Eigenschaften der Software bei der Analyse auf der gleichen Ebene des KIS lokalisiert worden sind. Um darüber hinaus im Text direkt auf einzelne Unterbereiche der Dokumentation verweisen zu können, wurde ein eigenes Format entwickelt, welches hier erläutert wird.

Nummerierung der Unterbereiche der Online-Dokumentation
Die für die Analyse verwendeten Online-Dokumentationen des ERP-Kerns, von IS-H und i.s.h.med wurden als Mengen von nummerierten Dokumenten gesichert. Die Dokumentation jedes einzelnen Moduls umfasst dabei mehrere Dokumente. Die Nummerierung der Dokumente entspricht der Position der Unterpunkte im Navigationsbaum der Online-Dokumentation des jeweiligen Moduls. Die Referenzen im Text benennen

- die Dokumentation des Moduls in der Form „SAP AG<x>“
- die Nummerierung des Dokuments entsprechend der o.a. Logik in der Form „D<x>“
- die Seiten im jeweiligen Dokument in der Form „S.<x>“.

Auffinden der referenzierten Stellen in der Online-Dokumentation
Aus Gründen des Urheber*innenrechts werden die zur Analyse verwendeten Dokumente nicht mit der Dissertation veröffentlicht. Alle Inhalte sind jedoch

D. Ametowobla, *Zur Soziologie der Software*,
https://doi.org/10.1007/978-3-658-37256-9

online verfügbar. Die folgenden Tabellen ordnen den im Text referenzierten Stellen der Dokumentation die jeweiligen URLs zu, um eine Quellenprüfung zu ermöglichen. Darüber hinaus wird jeweils angegeben, wie ausgehend von der Startseite der Online-Dokumentation des entsprechenden Moduls zur angegebenen Seite navigiert werden kann.

ERP-Kern

SAP AG (2001): SAP-Bibliothek. Release 4.6 C, April 2001. Online verfügbar unter https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/e1/8e51341a06084de10000009b38f83b/frameset.htm, zuletzt geprüft am 10.01.2020.

Dauerhafte Version der Online-Dokumentation erstellt von der consolut GmbH. Online verfügbar unter https://www.consolut.com/s/sap-ides-zugriff/sap-online-help-pdfs.html, zuletzt geprüft am 9.1.2020

Dokument D8.5.2, Titel: „Allgemeine Berichtsauswahl“

Referenz im Text	Url	Navigation
SAP AG 2001, D8.5.2, S. 6–8	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/eb/13815143c411d1896f0000e8322d00/frameset.htm	Basis > Basis-Services / Kommunikationsschnittstelle (BC-SRV) > Allgemeine Berichtsauswahl

Dokument D8.16.1: „SAP Business Workflow (BC-BMT-WFM)“

Referenz im Text	Url	Navigation
SAP AG 2001, D8.16.1, S. 36f,	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/a5/172437130e0d09e10000009b38f839/frameset.htm	Basis > Business Management (BC-BMT) > SAP Business Workflow (BC-BMT-WFM)
SAP AG 2001, D8.16.1, S. 39f	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/c5/e4a957453d11d189430000e829fbbd/frameset.htm	Basis > Business Management (BC-BMT) > SAP Business Workflow (BC-BMT-WFM) > Technische Grundlagen des SAP Business Workflow
SAP AG 2001, D8.16.1, S. 91–94	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/c5/e4a964453d11d189430000e829fbbd/frameset.htm	Basis > Business Management (BC-BMT) > SAP Business Workflow (BC-BMT-WFM) > Allgemeines Vorgehensmodell

Dokument D8.16.2.1: „Organisationsmanagement (BC-BMT-OM)“

Referenz im Text	Url	Navigation
SAP AG 2001, D8.16.2.1, S. 8–32	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/3c/c1366ac68411d2b499006094b9c9b4/frameset.htm	Basis > Business Management (BC-BMT) > Organisationsmanagement (BC-BMT-OM) > Modus Aufbauorganisation > Organisationsmanagement (und Unterseiten)
SAP AG 2001, D8.16.2.1, S. 12–32	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/b2/145f4ac70711d2b49a006094b9c9b4/frameset.htm	Basis > Business Management (BC-BMT) > Organisationsmanagement (BC-BMT-OM) > Modus Aufbauorganisation > Organisationsmanagement > Aufbauorganisation (und Unterseiten)
SAP AG 2001, D8.16.2.1, S. 15f	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/24/1188b0cbc711d2b49e006094b9c9b4/frameset.htm	Basis > Business Management (BC-BMT) > Organisationsmanagement (BC-BMT-OM) > Modus Aufbauorganisation > Organisationsmanagement > Aufbauorganisation > Organisationseinheit

Dokument D8.16.2.2: „Integration zum SAP Business Warehouse“

Referenz im Text	Url	Navigation
SAP AG 2001, D8.16.2.2	https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/96/aef837aea75351e10000009b38f8cf/frameset.htm	Basis > Business Management (BC-BMT) > Organisationsmanagement > Integration zum SAP Business Workflow (und Unterseiten)

IS-H

SAP AG (o. J.d): SAP Patient Management. Version IS-H 600 SP 050, German. Online verfügbar unter https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/dbdbc9531a081f4be10000000a174cb4.html, zuletzt geprüft am 10.01.2020.

Dauerhafte Version der Online-Dokumentation erstellt von Florence Eyok und Leonie Mader.

Dokument D3, „Basisdaten“

Referenz im Text	Url	Navigation
SAP AG o. J.d, D3, S. 98–108	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/4ed25fe5f6a85718e10000000a42189d.html	Basisdaten > Leistungsstammdaten (und Unterseiten)

Dokument D5, „Medizinische Pflegerische Dokumentation“

Referenz im Text	Url	Navigation
SAP AG o. J.d, D5	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/4eaeeb6f830825b4e10000000a42189b.html	Medizinische/Pflegerische Dokumentation (und Unterseiten)

Dokument D7, „Diagnosis Related Groups (Deutschland)“

Referenz im Text	Url	Navigation
SAP AG o. J.d, D7, S. 2	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/7772983a6d3943858f4c5c42bf1edce1.html	Diagnosis Related Groups (Deutschland)

Dokument D8, „Patientenabrechnung“

Referenz im Text	Url	Navigation
SAP AG o. J.d, D8	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/4f0a4e7bc95365c5e10000000a421937.html	Patientenabrechnung (und Unterseiten)
SAP AG o. J.d, D8, S. 24f	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/4dff2319d43b536de10000000a15822b.html	Patientenabrechnung > Leistungserfassung

Dokument D10, „Integration Controlling“

Referenz im Text	Url	Navigation
SAP AG o. J.d, D10	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/4d37f06c041e00d3e10000000a42189b.html	Integration Controlling (und Unterseiten)

Dokument D11, „Integration Materialwirtschaft“

Referenz im Text	Url	Navigation
SAP AG o. J.d, D11	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/4d3e0982a9961b32e10000000a42189b.html	Integration Materialwirtschaft (und Unterseiten)

Dokument D12, „Informationssystem IS-H-Reports“

Referenz im Text	Url	Navigation
SAP AG o. J.d, D12	https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/4d35bc8530265dc6e10000000a42189c.html	Informationssystem/IS-H-Reports (und Unterseiten)

i.s.h.med

SAP AG (o. J.a): i.s.h.med Klinisches System. Version IS-H 600 SP 052, German. Online verfügbar unter https://help.sap.com/viewer/15f3e584a3ba40b1b941f8623325492b/600.52/de-DE/e7dbc9531a081f4be10000000a174cb4.html, zuletzt geprüft am 10.01.2020.

Dauerhafte Version der Online-Dokumentation erstellt von Florence Eyok und Leonie Mader.

Dokument D1, „Allgemeine und rechtliche Informationen über i.s.h.med“

Referenz im Text	Url	Navigation
SAP AG o. J.a, D1, S. 3f	https://help.sap.com/viewer/15f3e584a3ba40b1b941f8623325492b/600.52/de-DE/d08a171381174364968c047c514726bd.html	Allgemeine und rechtliche Informationen über i.s.h.med > Intended Use Statement

Dokument D3, „Basisdaten“

Referenz im Text	Url	Navigation
SAP AG o. J.a, D3	https://help.sap.com/viewer/15f3e584a3ba40b1b941f8623325492b/600.52/de-DE/4d957fddd1b83c46e10000000a42189e.html	Klinischer Auftrag (und Unterseiten)

Dokument D9, „OP-System“

Referenz im Text	Url	Navigation
SAP AG o. J.a, D9, S. 2	https://help.sap.com/viewer/15f3e584a3ba40b1b941f8623325492b/600.52/de-DE/4dd33e668ab66524e10000000a42189b.html	OP-System

Literaturverzeichnis

Abbott, Andrew Delano (1988): The system of professions. An essay on the division of expert labor. Chicago: University of Chicago Press.

Akrich, Madeleine (2000): The De-Scription of Technical Objects. Describing the Interactions between Technics and Humans. In: Wiebe E. Bijker und John Law (Hg.): Shaping technology/building society. Studies in sociotechnical change. 3. print. Cambridge, Mass.: MIT Press, S. 205–224.

Akrich, Madeleine (2006): Die De-Skription technischer Objekte. In: Andréa Belliger und David J. Krieger (Hg.): ANThology. Ein einführendes Handbuch zur Akteur-Netzwerk-Theorie. Bielefeld: Transcript-Verl., S. 407–428.

Arthur, W. Brian (2011): The nature of technology. What it is and how it evolves. 1. Free Press pbk. ed. New York: Free Press.

Atkinson, Paul (1995): Medical talk and medical work. The liturgy of the clinic. London [u. a.]: Sage.

Austin, John L. (2007): Zur Theorie der Sprechakte. Zweite Vorlesung. In: Uwe Wirth (Hg.): Performanz. Zwischen Sprachphilosophie und Kulturwissenschaften. Orig.-Ausg., [Nachdr.]. Frankfurt am Main: Suhrkamp, S. 63–71.

Baecker, Dirk (2017): Wie verändert die Digitalisierung unser Denken und unseren Umgang mit der Welt? In: Rainer Gläß und Bernd Leukert (Hg.): Handel 4.0: Die Digitalisierung des Handels – Strategien, Technologien, Transformation. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 3–24.

Balzert, Helmut (2011): Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb. Heidelberg: Spektrum Akademischer Verlag.

Barley, Stephen R. (1990): The alignment of technology and structure through roles and networks. In: *Administrative Science Quarterly* 35 (1), S. 61–103.

Barrett, Rowena (2004): Working at Webboyz. An Analysis of Control Over the Software Development Labour Process. In: *Sociology* 38 (4), S. 777–794.

Beck, Kent; Beedle, Mike; van Bennekum, Arie; Cockburn, Alistair; Cunningham, Ward; Fowler, Martin et al. (2001): Manifesto for Agile Software Development. Online verfügbar unter https://moodle2016-17.ua.es/moodle/pluginfile.php/80324/mod_resource/content/2/agile-manifesto.pdf, zuletzt geprüft am 17.06.2019.

D. Ametowobla, *Zur Soziologie der Software*,
https://doi.org/10.1007/978-3-658-37256-9

Benders, Jos; Batenburg, Ronald; van der Blonk, Heico (2006): Sticking to standards; technical and other isomorphic pressures in deploying ERP-systems. In: *Information & Management* 43 (2), S. 194–203.

Beniger, James R. (1986): The control revolution. Technological and economic origins of the information society. Cambridge, Mass.: Harvard University Press.

Bijker, Wiebe E. (1995): Of bicycles, bakelites, and bulbs. Toward a theory of sociotechnical change. Cambridge, Mass: MIT Press.

Bijker, Wiebe E.; Hughes, Thomas P.; Pinch, Trevor J. (Hg.) (1987): The Social Construction of Technological Systems. New Directions in the Sociology and History of Technology. Universiteit Twente. Cambridge, Mass.: MIT Press.

Bloomfield, Brian P. (1995): Power, Machines and Social Relations: Delegating to Information Technology in the National Health Service. In: *Organization* 2 (3–4), S. 489–518.

Bloomfield, Brian P.; Vurdubakis, Theo (1997): Visions of Organizing and Organizations of Vision: The Representational Practices of Information Systems Development. In: *Accounting, Organizations and Society* 22 (7), S. 639–668.

Bode, Ingo (2010): Der Zweck heil(ig)t die Mittel? Ökonomisierung und Organisationsdynamik im Krankenhaussektor. In: Martin Endress und Thomas Matys (Hg.): Die Ökonomie der Organisation – die Organisation der Ökonomie. Wiesbaden: VS Verlag für Sozialwissenschaften / GWV Fachverlage GmbH, Wiesbaden, S. 63–92.

Boeder, Jochen; Groene, Bernhard (2014): The Architecture of SAP ERP. Understand how successful software works. Hamburg: Tredition.

Boudreau, M. C.; Robey, D. (2005): Enacting integrated information technology: A human agency perspective. In: *Organization Science* 16 (1), S. 3–18.

Bowker, Geoffrey C. (Hg.) (1997): Social science, technical systems, and cooperative work. Beyond the great divide. Mahwah, NJ: Lawrence Erlbaum Associates.

Bowker, Geoffrey C.; Star, Susan Leigh (2000): Sorting things out. Classification and its consequences. First paperback edition. Cambridge, Massachusetts, London, England: The MIT Press.

Brandt-Pook, Hans; Kollmeier, Rainer (2015): Softwareentwicklung kompakt und verständlich. Wie Softwaresysteme entstehen. 2. Auflage. Wiesbaden: Springer Vieweg.

Button, Graham; Sharrock, Wes (1996): Project work: The organisation of collaborative design and development in software engineering. In: *Comput Supported Coop Work* 5 (4), S. 369–386.

Button, Graham; Sharrock, Wes (1998): The Organizational Accountability of Technological Work. In: *Social studies of science* 28 (1), S. 73–102.

Callon, Michel; Millo, Yuval; Muniesa, Fabian (Hg.) (2007): Market devices. Malden, Mass.: Blackwell.

Cassell, Joan (1987): On control, certitude, and the "paranoia" of surgeons. In: Culture, Medicine and Psychiatry 11 (2), S. 229–249.

Claus, Volker; Schwill, Andreas (Hg.) (2003): Duden Informatik. Ein Fachlexikon für Studium und Praxis. Originalausg., 3. Aufl., [Taschenbuchausg.]. Mannheim: Dudenverl.

Crozier, Michel; Friedberg, Erhard (1979): Macht und Organisation. Die Zwänge kollektiven Handelns. Königstein/Ts.: Athenäum.

D'Adderio, Luciana (2008): The performativity of routines: Theorising the influence of artefacts and distributed agencies on routines dynamics. In: *Research Policy* 37 (5), S. 769–789.

Dakanalis, Antonios; Clerici, M.; Caslini, M.; Favagrossa, L.; Prunas, A.; Volpato, C. et al. (2014): Internalization of sociocultural standards of beauty and disordered eating behaviours. The role of body surveillance, shame and social anxiety. In: *Journal of Psychopathology* 20, S. 33–37.

Davenport, Thomas H. (1998): Putting the enterprise into the enterprise system. In: *Harvard Business Review* 76 (4), S. 121–131.

David, Shay; Pinch, Trevor J. (2008): Six Degrees of Reputation: The Use and Abuse of Online Review and Recommendation Systems. In: Trevor J. Pinch und Richard Swedberg (Hg.): Living in a material world. Economic sociology meets science and technology studies. Cambridge, Mass.: MIT Press, S. 341–374.

Davis, Martin (1995): Mathematical Logic and the Origin of Modern Computers. In: Rolf Herken (Hg.): The universal Turing machine. A half-century survey. 2. ed. Wien: Springer, S. 135–158.

Derr, Erik; Bugiel, Sven; Fahl, Sascha; Acar, Yasemin; Backes, Michael (2017): Keep me Updated. An Empirical Study of Third-Party Library Updatability on Android. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. Dallas, Texas, USA: ACM, S. 2187–2200.

DeSanctis, G.; Poole, M. S. (1994): Capturing the complexity in advanced technology use: Adaptive structuration theory. In: *Organization Science* 5 (2), S. 121–147.

Dierkes, Meinolf; Hoffmann, Ute; Marz, Lutz (1992): Leitbild und Technik: Edition Sigma.

DIMDI (2019): Klassifikationen. Deutsches Institut für Medizinische Dokumentation und Information. Online verfügbar unter https://www.dimdi.de/dynamic/de/klassifikationen/, zuletzt geprüft am 14.08.2019.

Dolata, Ulrich (2011): Wandel durch Technik. Eine Theorie soziotechnischer Transformation. Frankfurt am Main: Campus-Verl.

Edwards, Paul N. (2008): Y2K. Millennial reflections on computers as infrastructure. In: *History and Technology* 15 (1–2), S. 7–29.

Egger, Edeltraud; Wagner, Ina (1993): Negotiating temporal orders. In: *Computer Supported Cooperative Work (CSCW)* 1 (4), S. 255–275.

Engelmann, Carsten; Ametowobla, Dzifa (2017): Advancing the integration of hospital IT. Pitfalls and perspectives when replacing specialized software for high-risk environments with enterprise system extensions. In: *Applied clinical informatics* 8 (2), S. 515–528.

Engelmann, Carsten; Grote, Gudela; Geyer, Siegfried; Ametowobla, Dzifa (2017): Operating lists are created by rational algorithms and use of power. What can a social scientific view offer surgeons? In: *Langenbeck's archives of surgery* 402 (1), S. 187–190.

Feißt, Martin; Moltzberger, Kaspar (2016): Die Praxis der Zahlen im Krankenhausmanagement. Fakt oder Fetisch? In: Ingo Bode und Werner Vogd (Hg.): Mutationen des Krankenhauses. Soziologische Diagnosen in organisations- und gesellschaftstheoretischer Perspektive. Wiesbaden: Springer VS, S. 119–142.

Floyd, Christiane (1989): Softwareentwicklung als Realitätskonstruktion. In: Wolfram-M Lippe (Hg.): Software-Entwicklung: Konzepte, Erfahrungen, Perspektiven Fachtagung, veranstaltet vom Fachausschuß 2.1 der GI Marburg, 21.–23. Juni 1989 Proceedings. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 1–20.

Freidson, Eliot (Hg.) (1963): The hospital in modern society. London: The Free Press of Glencoe.

Freidson, Eliot; Rohde, Johann Jürgen; Schoene, Wolfgang (1979): Der Ärztestand. Berufs- und wissenschaftssoziologische Durchleuchtung einer Profession. Stuttgart: Enke.

Friedberg, Erhard (1995): Ordnung und Macht. Dynamiken organisierten Handelns. Frankfurt/Main, New York: Campus-Verl.

Friedland, Roger; Alford, Robert R. (1991): Bringing society back in: Symbols, practices and institutional contradictions.

Fulk, Janet; Steinfield, Charles (Hg.) (1990): Organizations and communication technology. Newbury Park, Calif: Sage.

Funken, Christiane (2001): Modellierung der Welt. Wissenssoziologische Studien zur Software-Entwicklung. Opladen: Leske und Budrich.

Funken, Christiane; Schulz-Schaeffer, Ingo (Hg.) (2008): Digitalisierung der Arbeitswelt. Zur Neuordnung formaler und informeller Prozesse in Unternehmen. 1. Aufl. Wiesbaden: VS, Verl. für Sozialwiss.

Gerlinger, Thomas (2009): Der Wandel der Interessenvermittlung in der Gesundheitspolitik. In: Britta Rehder, Thomas von Winter und Ulrich Willems (Hg.): Interessenvermittlung in Politikfeldern: Vergleichende Befunde der Policy- und Verbändeforschung. Wiesbaden: VS Verlag für Sozialwissenschaften, S. 33–51.

Giddens, Anthony (1995): Agency, Institution, and Time-Space Analysis. In: Donald McQuarie und R. Serge Denisoff (Hg.): Readings in contemporary sociological theory. From modernity to post-modernity. Englewood Cliffs N.J.: Prentice Hall, S. 335–343.

Giddens, Anthony (2008 [1984]): The constitution of society. Outline of the theory of structuration. Cambridge: Polity Press.

Giddens, Anthony (2010 [1995]): Konsequenzen der Moderne. 1. Aufl., [Nachdr.]. Frankfurt am Main: Suhrkamp.

Giddens, Anthony; Fleck, Christian; Zilian, Hans G. (Hg.) (1999): Soziologie. 2., überarb. Aufl. Graz: Nausner & Nausner.

Glaser, Barney G.; Strauss, Anselm L. (2017): Theoretical Sampling. In: Norman K. Denzin (Hg.): Sociological Methods: Routledge, S. 105–114.

Gocke, Peter (2011): Übersicht: Klinisches Arbeitsplatzsystem. In: Jörg F. Debatin und Peter Gocke (Hg.): IT im Krankenhaus. Von der Theorie in die Umsetzung. 1. Auflage. s.l.: Medizinisch Wissenschaftliche Verlagsgesellschaft, S. 147–158.

Gosain, Sanjay (2004): Enterprise Information Systems as Objects and Carriers of Institutional Forces: The New Iron Cage? In: *Journal of the Association for Information Systems* 5 (4).

Greenwood, Royston; Hinings, C. R. (1993): Understanding Strategic Change: The Contribution of Archetypes. In: *The Academy of Management Journal* 36 (5), S. 1052–1081.

Grenz, Tilo (2014): Digitale Medien und ihre Macher. Mediatisierung als dynamischer Wechselwirkungsprozess. In: Tilo Grenz und Gerd Möll (Hg.): Unter Mediatisierungsdruck: Änderungen und Neuerungen in heterogenen Handlungsfeldern. Wiesbaden: Springer Fachmedien Wiesbaden, S. 19–50.

Grenz, Tilo; Möll, Gerd (Hg.) (2014): Unter Mediatisierungsdruck: Änderungen und Neuerungen in heterogenen Handlungsfeldern. Wiesbaden: Springer Fachmedien Wiesbaden.

Grint, Keith; Woolgar, Steve (1992): Computers, Guns, and Roses: What's Social about Being Shot? In: *Science, Technology & Human Values* 17 (3), S. 366–380.

Großegger, Beate (2017): Bodyshaming und Social Media. Fokusgruppen mit 15- bis 19-jährigen Mädchen/jungen Frauen aus Wien. Hg. v. Intsitut für Jugendkulturforschung. Wien. Online verfügbar unter https://www.wien.gv.at/gesundheit/beratung-vorsorge/frauen/frauengesundheit/pdf/bodyshaming-fokusgruppen.pdf, zuletzt geprüft am 03.10.2019.

Haas, Peter; Kuhn, Klaus (2011): Krankenhausinformationssysteme – Ziele, Nutzen, Topologie, Auswahl. In: Rüdiger Kramme (Hg.): Medizintechnik: Verfahren – Systeme – Informationsverarbeitung. Berlin, Heidelberg: Springer Berlin Heidelberg, S. 781–805.

Hanseth, Ole; Monteiro, Eric (1997): Inscribing behaviour in information infrastructure standards. In: *Accounting, Management and Information Technologies* 7 (4), S. 183–211.

Heimer, Carol A. (1999): Competing Institutions: Law, Medicine, and Family in Neonatal Intensive Care. In: *Law & Society Review* 33 (1), S. 17–66.

Hellige, Hans Dieter (1996): Technikleitbilder als Analyse-, Bewertungs- und Steuerungsinstrumente: Eine Bestandsaufnahme aus informatik- und computerhistorischer Sicht. In: Hans Dieter Hellige (Hg.): Technikleitbilder auf dem Prüfstand. Leitbild-Assessment aus Sicht der Informatik- und Computergeschichte. Berlin: Ed. Sigma, S. 15–35.

Herold, Helmut; Lurz, Bruno; Wohlrab, Jürgen (2011): Grundlagen der Informatik. Praktisch – technisch – theoretisch. [Nachdr.]. München: Pearson Studium.

Hirsch-Kreinsen, Hartmut (2015): Digitalisierung von Arbeit. Folgen, Grenzen und Perspektiven. Hg. v. Hartmut Hirsch-Kreinsen, Johannes. Weyer und Maximilane Wilkesmann. Technische Universität Dortmund. Dortmund (Soziologische Arbeitspapiere, 43). Online verfügbar unter http://www.wiwi.tu-dortmund.de/wiwi/de/forschung/gebiete/fp-hirschkreinsen/forschung/soz_arbeitspapiere/AP-SOZ-43.pdf, zuletzt geprüft am 07.06.2018.

Hoda, Rashina; Salleh, Norsaremah; Grundy, John (2018): The Rise and Evolution of Agile Software Development. In: *IEEE Softw.* 35 (5), S. 58–63.

Hoeksma, John (2009): Siemens buys i.s.h.med from T-Systems Austria. Online verfügbar unter https://www.digitalhealth.net/2009/03/siemens-buys-i-s-h-med-from-t-systems-austria/, zuletzt aktualisiert 17.03.2009, zuletzt geprüft am 15.08.2019.

Hohlmann, Brita (2007): Organisation SAP – Soziale Auswirkungen technischer Systeme. Dissertation. Technische Universität Darmstadt, Darmstadt. Fachbereich Gesellschafts- und Geschichtswissenschaften.

Holland, C. R.; Light, B. (1999): A critical success factors model for ERP implementation. In: *IEEE Software* 16 (3), S. 30–36.

Houben, Daniel; Prietl, Bianca (Hg.) (2018): Datengesellschaft. Einsichten in die Datafizierung des Sozialen: transcript Verlag.

Hutter, Michael; Knoblauch, Hubert; Rammert, Werner; Windeler, Arnold (2011): Innovationsgesellschaft heute: Die reflexive Herstellung des Neuen. TU Berlin. Berlin (TUTS Working Papers, 4–2011).

Hyysalo, Sampsa; Jensen, Torben Elgaard; Oudshoorn, Nelly (Hg.) (2016): The New Production of Users. Changing Innovation Collectives and Involvement Strategies. 1. Aufl. New York, Abingdon: Routledge.

Iseringhausen, Olaf; Staender, Johannes (2012): Das Krankenhaus als Organisation. In: Maja Apelt und Veronika Tacke (Hg.): Handbuch Organisationstypen. Wiesbaden: VS Verlag für Sozialwissenschaften, S. 185–203.

Jarke, Juliane (2018): Digitalisierung und Gesellschaft. In: *Soziologische Revue* 41 (1), S. 3–20.

Kallinikos, Jannis (2005): The order of technology: Complexity and control in a connected world. Technology as Organization/Disorganization. In: *Information and Organization* 15 (3), S. 185–202.

Kallinikos, Jannis (2009): On the Computational Rendition of Reality: Artefacts and Human Agency. In: *Organization* 16 (2), S. 183–202.

Kallinikos, Jannis; Aaltonen, Aleksi; Marton, Attila (2013): The Ambivalent Ontology of Digital Artifacts. In: *MIS Quarterly* 37 (2), S. 357–370.

Kallinikos, Jannis; Hasselbladh, Hans (2009): Work, control and computation: Rethinking the legacy of neo-institutionalism. In: Renate E. Meyer, Kerstin Sahlin, Marc J. Ventresca und Peter Walgenbach (Hg.): Institutions and Ideology, Bd. 27: Emerald Group Publishing Limited, S. 257–282.

Karasti, Helena; Pipek, Volkmar; Bowker, Geoffrey C. (2018): An Afterword to 'Infrastructuring and Collaborative Design'. In: *Computer Supported Cooperative Work (CSCW)* 27 (2), S. 267–289.

Kellogg, Katherine C. (2009): Operating Room: Relational Spaces and Microinstitutional Change in Surgery. In: *American Journal of Sociology* 115 (3), S. 657–711.

Kellogg, Katherine C. (2011): Challenging operations: Medical reform and resistance in surgery: University of Chicago Press.

Kirchner, Stefan; Beyer, Jürgen (2016): Die Plattformlogik als digitale Marktordnung. In: *Zeitschrift für Soziologie* 45 (5), S. 868.

Kirchner, Stefan; Schüßler, Elke (2019): The Organization of Digital Marketplaces. Unmasking the Role of Internet Platforms in the Sharing Economy. In: Nils Brunsson und Göran Ahrne (Hg.): Organization outside Organizations. Cambridge: Cambridge University Press, 131–154.

Kitchin, Rob; Dodge, Martin (2011): Code/Space. Software and Everyday Life. Cambridge, Mass.: MIT Press.

Klatetzki, Thomas (2010): Soziale personenbezogene Dienstleistungsorganisationen als Typus. In: Thomas Klatetzki (Hg.): Soziale personenbezogene Dienstleistungsorganisationen. Soziologische Perspektiven. Wiesbaden: VS Verlag für Sozialwissenschaften, S. 7–24.

Klaus, Helmut; Rosemann, Michael; Gable, Guy G. (2000): What is ERP? In: *ISF* 2 (2), S. 141–162.

Kneer, Georg (2009): Akteur-Netzwerk-Theorie. In: Georg Kneer und Markus Schroer (Hg.): Handbuch Soziologische Theorien. Wiesbaden: VS Verlag für Sozialwissenschaften.

Kropf, Jonathan; Laser, Stefan (Hg.) (2019): Digitale Bewertungspraktiken. Für eine Bewertungssoziologie des Digitalen. Wiesbaden: Springer Fachmedien Wiesbaden.

Kühne, Thomas (2006): Matters of (Meta-) Modeling. In: *Software & Systems Modeling* 5 (4), S. 369–385.

Lange, Johannes (2016): Alles nur Illusion? Problematische Steuerungsrationalitäten im zeitgenössischen Krankenhauswesen. In: Ingo Bode und Werner Vogd (Hg.): Mutationen des Krankenhauses. Soziologische Diagnosen in organisations- und gesellschaftstheoretischer Perspektive. Wiesbaden: Springer VS, S. 47–65.

Latour, Bruno (2001): Eine Soziologie ohne Objekt? Anmerkungen zur Interobjektivität. In: *Berliner Journal für Soziologie* 11 (2), S. 237–252.

Law, John (1992): Notes on the theory of the actor-network: Ordering, strategy, and heterogeneity. In: *Systems Practice* 5 (4), S. 379–393.

Lee, Charlotte P.; Schmidt, Kjeld (2018): A Bridge Too Far? Critical Remarks on the Concept of "Infrastructure" in Computer-Supported Cooperative Work and Information Systems. In: Volker Wulf, Volkmar Pipek, David A. Randall, Kjeld Schmidt, Gunnar Stevens und Markus Rohde (Hg.): Socio-informatics. A practice-based perspective on the design and use of IT artifacts. Oxford: Oxford University Press, S. 177–217.

Leonardi, Paul M. (2009): Crossing the Implementation Line: The Mutual Constitution of Technology and Organizing Across Development and Use Activities. In: *Communication Theory* 19 (3), S. 278–310.

Leonardi, Paul M. (2013): When does technology use enable network change in organizations? A comparative study of feature use and shared affordances. In: *MIS Quarterly* 37 (3), S. 749–775.

Leonardi, Paul M.; Barley, Stephen R. (2008): Materiality and change: Challenges to building better theory about technology and organizing. In: *Information and Organization* 18 (3), S. 159–176.

Leonardi, Paul M.; Barley, Stephen R. (2010): What's Under Construction Here? Social Action, Materiality, and Power in Constructivist Studies of Technology and Organizing. In: *The Academy of Management Annals* 4 (1), S. 1–51.

Leonardi, Paul M.; Nardi, Bonnie A.; Kallinikos, Jannis (Hg.) (2012): Materiality and organizing. Social interaction in a technological world. 1. Aufl. Oxford: Oxford University Press.

Light, Ben (2001): The maintenance implications of the customization of ERP software. In: *J. Softw. Maint. Evol.: Res. Pract* 13 (6), S. 415–429.

Linares-Vásquez, Mario; Bavota, Gabriele; Bernal-Cárdenas, Carlos; Di Penta, Massimiliano; Oliveto, Rocco; Poshyvanyk, Denys (2013): API change and fault proneness. A threat to the success of Android apps. In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. Saint Petersburg, Russia: ACM, S. 477–487.

Locke, Joanne; Lowe, Alan (2007): A Biography: Fabrications in the Life of an ERP Package. In: *Organization* 14 (6), S. 793–814.

Ludewig, Jochen (2003): Models in software engineering—an introduction. In: *Software and Systems Modeling* 2 (1), S. 5–14.

Ludewig, Jochen; Lichter, Horst (2013): Software Engineering. Grundlagen, Menschen, Prozesse, Techniken. 3., korrigierte Aufl. Heidelberg: dpunkt.verl.

Luhmann, Niklas (2000): Organisation und Entscheidung. Opladen: Westdeutscher Verlag.

Luhmann, Niklas (2009): Zur Komplexität von Entscheidungssituationen. In: *Soziale Systeme* 15 (1).

Luhmann, Niklas (2018): Soziale Systeme. Grundriß einer allgemeinen Theorie. 17. Auflage. Frankfurt am Main: Suhrkamp.

Lundin, Rolf A.; Söderholm, Anders (1995): A theory of the temporary organization. In: *Scandinavian Journal of Management* 11 (4), S. 437–455.

Mackenzie, Adrian (2006): Cutting code. Software and sociality. New York: Peter Lang.

MacKenzie, Donald (2006): An engine, not a camera. How financial models shape markets. Cambridge Mass.: MIT Press.

MacKenzie, Donald; Muniesa, Fabian; Siu, Lucia (Hg.) (2007): Do economists make markets? On the performativity of economics. Princeton N.J.: Princeton Univ. Press.

Mahoney, Michael S. (2002): Software as Science—Science as Software. In: Ulf Hashagen, Reinhard Keil-Slawik und ArthurL Norberg (Hg.): History of Computing: Software Issues: Springer Berlin Heidelberg, S. 25–48.

Meissner, Gerd (1997): SAP – die heimliche Software-Macht. Wie ein mittelständisches Unternehmen den Weltmarkt eroberte. 1. Aufl. Hamburg: Hoffmann und Campe.

Merriam-Webster (2019): solicit. In: The Merriam-Webster.com Dictionary. Online verfügbar unter https://www.merriam-webster.com/dictionary/solicit, zuletzt geprüft am 06.12.2019.

Mills, C. Wright (2000 [1959]): The Sociological Imagination: Oxford University Press.

Mintzberg, Henry (1991): Mintzberg über Management. Führung und Organisation, Mythos und Realität. Wiesbaden: Gabler.

Monteiro, Eric; Hanseth, Ole (1996): Social Shaping of Information Infrastructure: On Being Specific about the Technology. In: Wanda J. Orlikowski (Hg.): Information technology and changes in organizational work. Proceedings of the IFIP WG8.2 Working Conference on Information Technology and Changes in Organizational Work, December 1995. London: Chapman & Hall on behalf of the International Federation for Information Processing (IFIP), S. 325–343.

Monteiro, Eric; Pollock, Neil; Hanseth, Ole; Williams, Robin (2013): From Artefacts to Infrastructures. In: *Computer Supported Cooperative Work (CSCW)* 22 (4–6), S. 575–607.

Monteiro, Eric; Pollock, Neil; Williams, Robin (2014): Innovation in Information Infrastructures. Introduction to the Special Issue. In: *Journal of the Association for Information Systems* 15 (4), S. i–x.

Mormann, Hannah (2010): Zur informationstheoretischen und organisationstheoretischen Formalisierung von Organisation. In: Jan-Hendrik Passoth und Josef Wehner (Hg.): Web 3.0. Zur Vermessung des Internets. 1., Auflage. Wiesbaden: VS Verlag für Sozialwissenschaften, S. 69–86.

Mormann, Hannah (2016): Das Projekt SAP. Zur Organisationssoziologie betriebswirtschaftlicher Standardsoftware. Dissertation. Bielefeld: transcript Verlag.

Myers, Glenford J.; Badgett, Tom; Sandler, Corey (2012): The art of software testing. 3. ed. Hoboken, NJ: Wiley.

Nielsen, Jakob (1993): Usability Engineering. Burlington: Elsevier Science.

Opdyke, William F. (1992): Refactoring Object-Oriented Frameworks. PhD Thesis. University of Illinois, Urbana-Champaign. Computer Science.

Orlikowski, Wanda J. (2000): Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations. In: *Organization Science*, S. 404–428.

Orlikowski, Wanda J. (1992): The Duality of Technology: Rethinking the Concept of Technology in Organizations. In: *Organization Science* 3 (3), S. 398–427.

Orlikowski, Wanda J. (2005): Material Works: Exploring the Situated Entanglement of Technological Performativity and Human Agency. In: *Scandinavian Journal of Information Systems* 17 (1), Artikel 6.

Orlikowski, Wanda J.; Iacono, C. Suzanne (2001): Research Commentary: Desperately Seeking the "IT" in IT Research—A Call to Theorizing the IT Artifact. In: *Information Systems Research* 12 (2), S. 121–134.

Orlikowski, Wanda J.; Robey, Daniel (1991): Information technology and the structuring of organizations. In: *Information Systems Research* 2 (2), S. 143–169.

Ortmann, Günther; Sydow, Jörg; Windeler, Arnold (2000): Organisation als reflexive Strukturation. In: Günther Ortmann, Jörg Sydow und Klaus Türk (Hg.): Theorien der Organisation. Die Rückkehr der Gesellschaft. 2., durchges. Aufl. Wiesbaden: Westdt. Verl., S. 315–354.

Ortmann, Günther; Windeler, Arnold; Becker, Albrecht; Schulz, Hans-Joachim (1990): Computer und Macht in Organisationen. Mikropolitische Analysen. Opladen: Westdt. Verl.

Oudshoorn, Nelly; Rommes, Els; Stienstra, Marcelle (2004): Configuring the User as Everybody: Gender and Design Cultures in Information and Communication Technologies. In: *Science Technology Human Values* 29 (1), S. 30–63.

Patterson, David A.; Hennessy, John L. (2016): Rechnerorganisation und Rechnerentwurf. Die Hardware/Software-Schnittstelle. 5. Auflage. Berlin, Boston: De Gruyter Oldenbourg.

Perrow, Charles (1986): Complex organizations. A critical essay. New York: Random House.

Petersen, Olaf (2004): Unternehmensgestaltung zwischen Hierarchie und Egalität. Dynamik, Konflikte und Commitment in IT-Startups. Dissertation. Freie Universität, Berlin. Fachbereich Erziehungswissenschaften und Psychologie.

Pfleeger, Shari Lawrence; Atlee, Joanne M. (2006): Software engineering. Theory and practice. 3rd ed. Upper Saddle River, N.J: Pearson/Prentice Hall.

Pinch, Trevor J.; Bijker, Wiebe E. (1987): The Social Construction of Facts and Artifacts: Or How the Sociology of Science and the Sociology of Technology Might Benefit Each Other. In: Wiebe E. Bijker, Thomas P. Hughes und Trevor J. Pinch (Hg.): The Social Construction of Technological Systems. New Directions in the Sociology and History of Technology. Cambridge, Mass.: MIT Press, S. 17–50.

Pipek, Volkmar; Karasti, Helena; Bowker, Geoffrey C. (2017): A Preface to 'Infrastructuring and Collaborative Design'. In: *Computer Supported Cooperative Work (CSCW)* 26 (1), S. 1–5.

Pipek, Volkmar; Wulf, Volker (2009): Infrastructuring: Towards an Integrated Perspective on the Design and Use of Information Technology. In: *Journal of the Association for Information Systems* 10 (5), S. 306–332.

Pollock, Neil; Cornford, James (2004): ERP systems and the university as a "unique" organisation. In: *Information Technology & People* 17, S. 31–52.

Pollock, Neil; Williams, Robin (2009): Software and organisations. The biography of the enterprise-wide system or how SAP conquered the world. London, New York: Routledge.

Pollock, Neil; Williams, Robin; D'Adderio, Luciana (2007): Global Software and its Provenance. In: *Social studies of science* 37 (2), S. 254–280.

Pressman, Roger S. (1997): Software engineering. A practitioner's approach. 4. Aufl. New York: McGraw-Hill.

Rammert, Werner (1998): Die Form der Technik und die Differenz der Medien. Auf dem Weg zu einer pragmatischen Technikheorie. In: Werner Rammert (Hg.): Technik und Sozialtheorie. Frankfurt/Main: Campus, S. 293–326.

Rammert, Werner (2000): Technik aus soziologischer Perspektive. Wiesbaden: Westdt. Verl.

Rammert, Werner (2006): Technik in Aktion und Interaktion: verteilte Aktivitäten in hybriden Konstellationen. In: Werner Rammert und Cornelius Schubert (Hg.): Technografie. Zur Mikrosoziologie der Technik. Frankfurt am Main: Campus-Verl., S. 163–195.

Rammert, Werner; Schlese, Michael; Wagner, Gerald; Wehner, Josef; Weingarten, Rüdiger (1998): Wissensmaschinen. Soziale Konstruktion eines technischen Mediums: das Beispiel Expertensysteme. Frankfurt, New York: Campus.

Reckwitz, Andreas (2003): Grundelemente einer Theorie sozialer Praktiken. In: *Zeitschrift für Soziologie* 32 (4), S. 282.

Robbes, Romain; Lungu, Mircea; Röthlisberger, David (2012): How do developers react to API deprecation? In: Will Tracz, Martin Robillard und Tevfik Bultan (Hg.): Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering – FSE '12. the ACM SIGSOFT 20th International Symposium. Cary, North Carolina, 11.11.2012–16.11.2012. New York, New York, USA: ACM Press, S. 1.

Robey, Daniel; Ross, Jeanne W.; Boudreau, Marie-Claude (2002): Learning to Implement Enterprise Systems: An Exploratory Study of the Dialectics of Change. In: *Journal of Management Information Systems* 19 (1), S. 17–46.

Rohde, Johann Jürgen (1974): Soziologie des Krankenhauses. Zur Einführung in die Soziologie der Medizin. 2., überarb. Aufl. Stuttgart: Enke.

Rolf, Arno; Möller, Andreas; Wolff, Bernd (1998): Grundlagen der Organisations- und Wirtschaftsinformatik. Berlin, New York: Springer.

Rönkkö, Kari; Dittrich, Yvonne; Randall, Dave (2005): When Plans do not Work Out: How Plans are Used in Software Development Projects. In: *Comput Supported Coop Work* 14 (5), S. 433–468.

Rooksby, John; Rouncefield, Mark; Sommerville, Ian (2009): Testing in the Wild. The Social and Organisational Dimensions of Real World Practice. In: *Computer Supported Cooperative Work (CSCW)* 18 (5), S. 559.

Rose, Jeremy; Jones, Matthew; Truex, Duane (2005): Socio-Theoretic Accounts of IS: The Problem of Agency. In: *Scandinavian Journal of Information Systems* 17 (1), Artikel 8.

SAP AG (o.J.a): i.s.h.med Klinisches System. Version IS-H 600 SP 052, German. Online verfügbar unter https://help.sap.com/viewer/15f3e584a3ba40b1b941f8623325492b/600.52/de-DE/e7dbc9531a081f4be10000000a174cb4.html, zuletzt geprüft am 15.08.2019.

SAP AG (o.J.b): SAP Dokumentation. Glossar. Online verfügbar unter https://help.sap.com/doc/saphelp_glossary/latest/de-DE/35/2cd77bd7705394e10000009b387c12/frameset.htm, zuletzt geprüft am 14.08.2019.

SAP AG (o.J.c): SAP Patient Management. IS-H 600 SP52. Hg. v. SAP AG. Online verfügbar unter https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.52/en-US/dbdbc9531a081f4be10000000a174cb4.html, zuletzt geprüft am 20.10.2019.

SAP AG (o.J.d): SAP Patient Management. Version IS-H 600 SP 050, German. Online verfügbar unter https://help.sap.com/viewer/0dd0cfcad4be430385f6839d286008b5/600.50/de-DE/dbdbc9531a081f4be10000000a174cb4.html, zuletzt geprüft am 14.08.2019.

SAP AG (2001): SAP-Bibliothek. Release 4.6C, April 2001. Online verfügbar unter https://help.sap.com/doc/saphelp_46c/4.6C/de-DE/e1/8e51341a06084de10000009b38f83b/frameset.htm, zuletzt aktualisiert April 2001, zuletzt geprüft am 05.07.2019.

Sawyer, Steve (2001): A market-based perspective on information systems development. In: *Commun. ACM* 44 (11), S. 97–102.

Schmidt, Kjeld (2018): Practice and Technology. On the Conceptual Foundations of Practice-Centered Computing. In: Volker Wulf, Volkmar Pipek, David A. Randall, Kjeld Schmidt, Gunnar Stevens und Markus Rohde (Hg.): Socio-informatics. A practice-based perspective on the design and use of IT artifacts. Oxford: Oxford University Press, S. 47–104.

Schmidt, Kjeld; Bannon, Liam (2013): Constructing CSCW. The first quarter century. In: *Comput Supported Coop Work* 22 (4–6), S. 345–372.

Schmidt, R. (2008): Praktiken des Programmierens. Zur Morphologie von Wissensarbeit in der Software-Entwicklung. In: *Zeitschrift für Soziologie* 37 (4), S. 282–300.

Schmidt, Robert (2012): Code Decay. Organizational Performance and Destructivity. In: *Critical Studies* 36 (1), S. 195–208.

Schulz-Schaeffer, Ingo (1999): Technik und die Dualität von Ressourcen und Routinen. In: *Zeitschrift für Soziologie* 28 (6), S. 409–428.

Schulz-Schaeffer, Ingo; Bottel, Matthias (2017): Regulierung durch Technik. Arbeitsverteilung und Arbeitsorganisation in Projekten transnational verteilter Softwareentwicklung. In: Stefan Lessenich (Hg.): Geschlossene Gesellschaften. Verhandlungen des 38. Kongresses der Deutschen Gesellschaft für Soziologie in Bamberg 2016. 38. Kongress der Deutschen Gesellschaft für Soziologie. Bamberg, 2016. Deutsche Gesellschaft für Soziologie. Online verfügbar unter http://publikationen.soziologie.de/index.php/kongressband_2016/article/view/429, zuletzt geprüft am 26.09.2019.

Schulz-Schaeffer, Ingo; Bottel, Matthias (2018): Die Herstellung transnational mobiler Arbeitstätigkeiten in der Softwareentwicklung. In: Sigrid Quack, Ingo Schulz-Schaeffer, Karen Shire und Anja Weiß (Hg.): Transnationalisierung der Arbeit. Wiesbaden: Springer Fachmedien Wiesbaden, S. 99–127.

Scott, Judy E.; Kaindl, Lisa (2000): Enhancing functionality in an enterprise software package. In: *Information & Management* 37 (3), S. 111–122.

Scott, W. Richard (2014): Institutions and organizations. Ideas, interests, and identities. Fourth edition.

Seyfert, Robert; Roberge, Jonathan (Hg.) (2017): Algorithmuskulturen. Über die rechnerische Konstruktion der Wirklichkeit. Algorithmic Cultures Conference; Universität Konstanz; Transcript GbR; „Algorithmic Cultures“ Conference. Bielefeld: Transcript.

Siemens AG (2009): Sieben gute Gründe für i.s.h.med – das ganzheitliche Krankenhausinformationssystem. Hg. v. Siemens AG. Erlangen.

Simon, Herbert Alexander (1997 [1945]): Administrative behavior. A study of decision-making processes in administrative organizations; [now updated with extensive new commentaries by the author]. 4. ed. New York, NY: Free Press.

Sommerville, Ian (2012): Software engineering. 9., aktualisierte Auflage. München, Harlow, Amsterdam: Pearson.

Stachowiak, Herbert (1973): Allgemeine Modelltheorie. Wien: Springer.

Star, Susan Leigh; Ruhleder, Karen (1996): Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces. In: *Information Systems Research* 7, S. 111–134.

Strübing, Jörg (2019): Grounded Theory und Theoretical Sampling. In: Nina Baur und Jörg Blasius (Hg.): Handbuch Methoden der empirischen Sozialforschung. Wiesbaden: Springer Fachmedien Wiesbaden, S. 525–544.

Suchman, Lucy A. (1987): Plans and situated actions. The problem of human-machine communication: Cambridge University Press.

Sydow, Jörg; Windeler, Arnold; Schubert, Cornelius; Möllering, Guido (2012): Organizing R&D consortia for path creation and extension. The case of semiconductor manufacturing technologies. In: *Organization Studies* 33 (7), S. 907–936.

Tilson, David; Lyytinen, Kalle; Sørensen, Carsten (2010): Research Commentary—Digital Infrastructures. The Missing IS Research Agenda. In: *Information Systems Research* 21 (4), S. 748–759.

Vogd, Werner (2004a): Ärztliche Entscheidungsfindung im Krankenhaus. Komplexe Fallproblematiken im Spannungsfeld von Patienteninteressen und administrativ-organisatorischen Bedingungen. In: *Zeitschrift für Soziologie* 33 (1), S. 26–47.

Vogd, Werner (2004b): Ärztliche Entscheidungsprozesse des Krankenhauses im Spannungsfeld von System- und Zweckrationalität. Eine qualitativ rekonstruktive Studie unter dem besonderen Blickwinkel von Rahmen („frames“) und Rahmungsprozessen. 1. Aufl. Berlin: VWF Verlag für Wissenschaft und Forschung.

Vogd, Werner (2016): Das Missverstehen des Ökonomischen. Oder vom Sündenfall falsch verstandener Rationalitäten im Krankenhaus. In: Ingo Bode und Werner Vogd (Hg.): Mutationen des Krankenhauses. Soziologische Diagnosen in organisations- und gesellschaftstheoretischer Perspektive. Wiesbaden: Springer VS, S. 281–308.

Volkoff, Olga; Strong, Diane M.; Elmes, Michael B. (2007): Technological Embeddedness and Organizational Change. In: *Organization Science* 18 (5), S. 832–848.

Wagner, Erica L.; Scott, Susan V.; Galliers, Robert D. (2006): The creation of ‘best practice’ software: Myth, reality and ethics. In: *Information and Organization* 16 (3), S. 251–275.

Walsham, Geoffrey (1993): Interpreting information systems in organizations. Chichester, West Sussex, England, New York: Wiley.

Weber, Max (2005 [1922]): Wirtschaft und Gesellschaft. Grundriss der verstehenden Soziologie; zwei Teile in einem Band. Frankfurt am Main, Affoltern a.A.: Zweitausendeins; Buch 2000.

Weltz, Friedrich; Ortmann, Rolf G. (1992): Das Softwareprojekt. Projektmanagement in der Praxis. Frankfurt u. a: Campus-Verl.

Wieandt-Ledebur, Michaela (2014): Regulierung internationaler IT-Beratungsprojekte. Eine Fallstudie aus Sicht der Strategischen Organisationsanalyse. Augsburg: Rainer Hampp Verlag.

Williams, Robin; Pollock, Neil (2012): Moving Beyond the Single Site Implementation Study: How (and Why) We Should Study the Biography of Packaged Enterprise Solutions. In: *Information Systems Research* 23 (1), S. 1–22.

Winner, Langdon (1980): Do artifacts have politics? In: *Daedalus* 109 (1), S. 121–136.

Woolgar, Steve (1991): Configuring the user: the case of usability trials. In: John Law (Hg.): A sociology of monsters: essays on power, technology and domination: Routledge London, S. 57–102.

Wulf, Volker; Pipek, Volkmar; Randall, David A.; Schmidt, Kjeld; Stevens, Gunnar; Rohde, Markus (Hg.) (2018): Socio-informatics. A practice-based perspective on the design and use of IT artifacts. Oxford: Oxford University Press.

Wurster, Glenn; van Oorschot, P. C. (2008): The Developer is the Enemy. In: Matt Bishop, Christian W. Probst, Angelos Keromytis und Anil Somayaji (Hg.): Proceedings of the 2008 workshop on New security paradigms – NSPW ‘08. the 2008 workshop. Lake Tahoe, California, USA, 22.09.2008–25.09.2008. New York, New York, USA: ACM Press, S. 89–97.

Yau, Stephen S.; Collofello, James S.; MacGregor, T. (Hg.) (1978): Ripple effect analysis of software maintenance. The IEEE Computer Society’s Second International Computer Software and Applications Conference, 1978. COMPSAC ‘78. The IEEE Computer

Society's Second International Computer Software and Applications Conference, 1978. COMPSAC '78.

Zibran, M. F.; Eishita, F. Z.; Roy, C. K. (Hg.) (2011): Useful, But Usable? Factors Affecting the Usability of APIs. 2011 18th Working Conference on Reverse Engineering. 2011 18th Working Conference on Reverse Engineering.

Zittrain, Jonathan (2019): Intellectual Debt: With Great Power Comes Great Ignorance. Hg. v. Medium. Online verfügbar unter https://medium.com/berkman-klein-center/from-technical-debt-to-intellectual-debt-in-ai-e05ac56a502c?, zuletzt geprüft am 15.12.2019.

Zuboff, Shoshana (1988): In the age of the smart machine. The future of work and power. New York: Basic Book