4. Softwaregestaltung als Teil der Digitalisierung

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2. Die zwei Kernprobleme der Softwaregestaltung

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

4.2.1. Softwaretechnische Interdisziplinarität

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

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

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

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

4.2.2. Softwaretechnische Gestaltungsmöglichkeiten

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Exkurs: Entwicklungsplattform von SAP

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

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

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

Tabelle 2: Anzahl kooperierende Firmen SAP allgemein und Versorgungswirtschaft

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

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