

Sonnenholzner, Johannes

Article

Keine Digitalisierung ohne Kommunikation: Warum Softwaregestaltung nicht nur ein technisches Problem ist

WZB-Mitteilungen: Quartalsheft für Sozialforschung

Provided in Cooperation with:

WZB Berlin Social Science Center

Suggested Citation: Sonnenholzner, Johannes (2023) : Keine Digitalisierung ohne Kommunikation: Warum Softwaregestaltung nicht nur ein technisches Problem ist, WZB-Mitteilungen: Quartalsheft für Sozialforschung, ISSN 2943-6613, Wissenschaftszentrum Berlin für Sozialforschung (WZB), Berlin, Iss. 180 (2/23), Online-Supplement, <https://bibliothek.wzb.eu/artikel/2023/f-25641.pdf>

This Version is available at:

<https://hdl.handle.net/10419/327842>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.



<https://creativecommons.org/licenses/by/4.0/>

Keine Digitalisierung ohne Kommunikation

Warum Softwaregestaltung nicht nur ein technisches Problem ist

Von [Johannes Sonnenholzner](#)

Organisationen in allen Branchen stehen vor der Aufgabe, branchenspezifische Software zu programmieren oder programmieren zu lassen. Dafür ist Wissen über den Anwendungsbereich der zukünftigen Software notwendig. Die Entwickler*innen müssen wissen, was sie zu programmieren haben. Der Teil der Softwareentwicklung, in dem darüber verhandelt wird, was eine Software können soll, wird im Folgenden Softwaregestaltung genannt. Er wurde in der Forschung bislang vernachlässigt. Johannes Sonnenholzner hat sich des Themas angenommen.

„Erst hat man in der Verwaltung versucht, IT-Experten Steuerrecht beizubringen und dann Steuerrechtler zu Programmierern umzuschulen“, sagt der Chef eines Finanzdienstleisters für das Bundeszentralamt für Steuern. Laut einem Artikel des Tagesspiegels war keiner der beiden Wege zufriedenstellend, um eine funktionierende Software zu entwickeln.

Nicht nur Verwaltungen, sondern Organisationen aller Branchen stehen vor der Herausforderung, branchenspezifische Software zu programmieren oder programmieren zu lassen. Dafür ist Wissen über den Anwendungsbereich der zukünftigen Software notwendig. Die Entwickler*innen müssen wissen, was sie zu programmieren haben. Der Teil der Softwareentwicklung, in dem darüber verhandelt wird, was eine Software können soll, wird im Folgenden Softwaregestaltung genannt. Diese wurde in der Forschung bislang vernachlässigt. Erforscht wurde meist die Anwendung von Software, ohne zu berücksichtigen, dass die eigentliche Technologie dahinter die programmierbedürftige Universalmaschine Computer ist.

Es ist ein neuer Blick auf Organisationen notwendig, wenn Software zentral für die Arbeitsgestaltung wird. In vielen Organisationen reden die Mitglieder noch so, als ob sie nicht eigentlich schon längst ein softwarebasierter und damit soziotechnischer Betrieb wären.

Für eine entsprechende neue Organisationstheorie wird als erster Schritt die Softwaregestaltung analysiert. Das ist jener Prozess, der dafür sorgt, dass die Anwender*innen die Software bekommen, die sie brauchen. Nach einigen allgemeineren Ausführungen und Begriffen soll diese an drei Fallstudien verdeutlicht und mit dem Begriff der soziotechnischen Netzwerkarbeit verknüpft werden.

Softwareentwicklung und -gestaltung ist kein Nischenthema: In Deutschland gibt es 304.005 Programmierer*innen (2022), 96.486 ERP-Berater*innen (2012) und 22.734 IT-Projektleiter*innen (2012). IT-Projektleiter*innen und ERP-Berater*innen

implementieren oder erweitern Standard-Softwarepakete. Größter Anbieter (und nach Börsenwert das wertvollste Unternehmen in Deutschland) ist SAP, dessen Softwarelösungen 457 Partnerfirmen in Deutschland und 2.796 weltweit implementieren, anpassen oder erweitern.

IT-Projektleiter*innen und ERP-Berater*innen sind nicht die einzigen Wissensarbeiter*innen, die zwischen Anwendung und Entwicklung für die Softwaregestaltung agieren. Anforderungsmanager*in, [Product Owner](#), [Scrum Master](#), Applikationsbetreuer*in, Digitalisierungsmanager*in, [Key User](#) oder Prozessmanager*in weisen nur auf eine Auswahl jener Rollen hin. Daneben kennt die Softwaregestaltung eigene softwarebasierte Werkzeuge (z.B. [Ticketsysteme](#)) und eigene Prozesse (z.B. [Scrum](#), IT-Projekte).

Diese Vielzahl an Rollen und Aufgaben verweist auf eine von drei Ursachen, warum Softwaregestaltung kein rein technisches Problem ist.

1. Softwaregestaltung ist ein eigenständiger Arbeitsprozess, der wie andere auch, organisiert werden muss. Wissensarbeiter*innen handeln in soziotechnischen Strukturen mit dem Ziel, die passende Software herzustellen. Dabei sind die wesentlichen Arbeitsmittel Wissen und Kommunikation.
2. Der Arbeitsprozess muss in verschiedenen Konstellationen etabliert werden, wobei unterschiedliche Interessen berücksichtigt werden müssen. Wie andere Arbeitsprozesse auch, ist die Softwaregestaltung wirtschaftlichen Indikatoren untergeordnet. Softwaregestaltung zu etablieren, wird koordinationsaufwendiger, je mehr Anwenderperspektiven berücksichtigt werden müssen und je mehr marktförmige und hierarchische Beziehungen zwischen Anwendung und Entwicklung existieren.
3. Es gibt zwei Kernprobleme, die im Arbeitsprozess der Softwaregestaltung gelöst werden müssen, das der softwaretechnischen Interdisziplinarität und das der softwaretechnischen Bezugsebene. Was ist darunter zu verstehen? In Arbeitskontexten mit umfangreichem und sich stetig änderndem Domänenwissen (wie z. B. Steuerrecht), muss interdisziplinär gearbeitet werden. Es ist selten und oft unmöglich, dass eine Person sämtliches Wissen auf sich vereint und allein eine Software schreiben kann. Denn hierfür ist die Spezialisierung innerhalb der IT und die der jeweiligen Branche, ihre Anwendungskontexte und organisationspezifischen Arbeitsabläufe, meist zu weit fortgeschritten. Das zweite Problem der softwaretechnischen Bezugsebene besteht darin, dass bei der Gestaltung einer Software für ein bestimmtes Anwendungsfeld Entscheidungen über den Zuschnitt der Software und die Ausrichtung der Organisation getroffen werden müssen. Soll etwas Individuelles programmiert oder ein Standard geschaffen werden, den auch andere nutzen können? Sollte die Organisation an einer bestehenden (Standard-)Software oder auf die Entwicklung von Software

ausgerichtet werden? Beide Fragen gehören zum Problem der softwaretechnischen

Bezugsebene. Die softwaretechnische Bezugsebene ist in Zeiten des CloudComputing allgegenwärtig, weil sie den Organisationen erlaubt, virtuell auf unzählige Applikationen und IT-Experten zuzugreifen und sie für den eigenen Arbeitskontext zu nutzen. Das ist für Fragen von Effizienz oder gar Überleben im Wettbewerb zentral. Digitale Start-ups und große Konzerne wie Amazon sind deshalb erfolgreich, weil sie von Anfang an ihre eigene Software entwickeln, die sie für ihre jeweiligen branchenspezifischen Bedarfe nutzen und zugleich sogar Konkurrenten aus der Branche anbieten. Gleichzeitig sind viele Industrieunternehmen und Verwaltungen auf Anbieter von Standard-Software angewiesen, weil sie selbst nicht die notwendigen Kompetenzen haben, über die nötigen Ressourcen verfügen oder keinen Mehrwert darin sehen, selbst zu programmieren.

Vor allem in der Energiewirtschaft lässt sich aufgrund der großen Vielfalt an Organisationen (z. B. Konzerne, große und kleine Stadtwerke), vieler regulatorischer Neuerungen und der Energiewende zeigen, wie mit den Möglichkeiten der Softwaregestaltung auf Änderungsbedarf reagiert wird. Dazu werden aus meiner am WZB entstehenden Doktorarbeit drei von sieben exemplarischen Fallstudien vorgestellt, hier als KOOP, INTERN und STARTUP bezeichnet. Sie alle erfordern eine kooperative Zusammenarbeit, um eine industriespezifische Software zu gestalten. Doch werden die Probleme der Interdisziplinarität und der softwaretechnischen Bezugsebene gleichermaßen gelöst?

In KOOP organisiert ein IT-Dienstleister (mit Branchen- und IT-Expertise) eine kooperative Softwaregestaltung mehrerer Energieversorgungsunternehmen (EVUs) (mit mehr Branchen- als IT-Expertise). Dafür finden regelmäßige Treffen statt. Dort wird für einzelne Anforderungen verhandelt, ob sie Teil des industriespezifischen Standards werden, den dann alle beteiligten EVUs nutzen. Die Anwender in den EVUs ordnen sich der so entstehenden Standardsoftware unter. Es war und ist eine Herausforderung, die Interessen der verschiedenen EVUs unter einen Hut zu bringen. Es gibt einen professionellen Mediator, der bei den Treffen vermittelt.

„Ich glaube, eines der größten Learnings ist, dass die größten Herausforderungen, einer der großen Kostenpunkte, die Abstimmung überhaupt und der Austausch über den Kunden ist. Also, dieses ganze Anforderungsmanagement und alles, was nichts mit Technik zu tun hat, [...] sondern wirklich miteinander reden und abstimmen und irgendwie sich auf einen gemeinsamen Nenner zu einigen. [...] Dafür haben wir eine Lösung gefunden, auch sogar irgendwann extern moderiert und auch immer noch.“ (Befragte*r KOOP)

Ein Kompromiss ist, dass es individuelle Abweichungen vom Standard geben kann. Dies muss von den EVUs entsprechend bezahlt werden. Da es sich um eine Marktbeziehung handelt, prüfen die EVUs ständig, ob es nicht einen günstigeren oder innovativeren Anbieter gibt. Gleichzeitig wissen sie um die Vorteile langfristiger Beziehungen (v. a. die gemeinsame Wissensbasis).

Bei INTERN entwickelt ein EVU selbst eine industriespezifische Anwendung. Da mehrere Fachbereiche die gleiche Software nutzen, wurde eine Anforderungsrunde eingerichtet, in der sich alle treffen und abstimmen. Weil hunderte Anwender*innen betroffen sind, gibt es in den Fachbereichen selbst noch Mitarbeiter*innen, die Anforderungen aufnehmen.

Sie können die Software aber nicht frei gestalten, weil die Hierarchien eine übergreifende Gestaltung nicht zulassen:

„D. h. Prozesse die eigentlich waagrecht laufen, die werden dauernd durch irgendwelche Hierarchien getrennt und der Leiter sagt halt: ‚Nicht mein Bier. Ich kümmere mich nicht um die Schnittstelle, weil mir tut es ja nicht weh.‘“ (Befragte*r INTERN)

Nur ein kleiner Teil der Anwender*innen ist in die Softwaregestaltung eingebunden. Zudem verändert sich ihre Stellung in der Organisation des EVUs durch den Arbeitsprozess der Softwaregestaltung. Ein Manager gibt als Ziel an, die Hierarchien abzubauen und eine soziotechnische Prozessorganisation zu etablieren, der die Entwickler dann direkt zugeordnet sind. So könnte effizienter interdisziplinär zusammengearbeitet und die Möglichkeiten der Softwaregestaltung ausgereizt werden.

Bei STARTUP basiert die Organisation von Anfang an darauf, dass sie ihren industriespezifischen Zweck (Abwicklung des Emissionshandels für E-Mobilität) durch Softwareentwicklung erreicht. Es gilt das Primat der Softwareentwicklung. Die gesamte Organisation ist darauf ausgerichtet, sich interdisziplinär auszutauschen und iterativ die

Software weiterzuentwickeln – im Netzwerk: ohne Hierarchien, Organisations- und Abteilungsgrenzen oder Konkurrenzgedanken. Anforderungen werden in regelmäßigen und spontanen Treffen geschrieben. Die Beziehungen sind offen und kooperativ:

„Da ziehen halt alle am gleichen Strang, ohne viele persönliche Befindlichkeiten, die dahinterstecken, ohne irgendwelche Ego-Geschichten.“ (Befragte*r STARTUP)

Den Anwender*innen verbleibt die Arbeit, die durch Softwareentwicklung nicht automatisiert werden kann.

Überblick über die Fallstudien:

Fälle	Grund- koordination	Netzwerke der Softwaregestaltung	Softwaretechnische Bezugsebene
KOOP	Markt	IT-Dienstleister: organisiert Anforderungsrunde mit EVUs	verhandelte Standardsoftware
INTERN	Hierarchie	IT-Abteilung: organisiert Anforderungsrunde mit Fachbereichen	individuelle Software
STARTUP	Netzwerk	Teamtreffen und Chatgruppen	Primat der Softwareentwicklung

Bei der Softwaregestaltung handelt es sich um eine spezifische Form von Arbeit und Organisation, die sich als soziotechnische Netzwerkarbeit konzeptionieren lässt. Diese zeichnet sich dadurch aus, dass die Wissensarbeiter*innen rollenbasiert und situativ

handeln. Typische Erwartungen sind: kooperativ und selbstorganisiert zu sein, sich auf Software einzulassen, mit Nicht-Wissen umzugehen und sich auf wechselnde zwischenmenschliche Konstellationen einzustellen. Darüber hinaus sind Beziehungen zentral: partnerschaftlich und auf Augenhöhe, für eine interpersonelle Abstimmung auf operativer und strategischer Ebene. Außerdem müssen Erwartungen regelmäßig abgeglichen und Konflikte professionell gelöst werden (u. U. mit Mediator*innen). Zuletzt gehört ein feedbackorientierter, iterativer Arbeitsablauf dazu, der auf ein Netzwerk aus Wissensarbeiter*innen (verteilt auf Organisationen, Abteilungen, Teams) für die interdisziplinäre Anforderungsaufnahme zurückgreift. So ist sichergestellt, dass die entwickelte Software die Erwartungen des Anwendungsbereichs erfüllt und den technologischen Möglichkeiten gerecht wird.

Fazit: Es geht nicht darum, dass Steuerfachleute programmieren lernen oder Programmierer zu Steuerexperten werden. Wichtiger als das Wissen der Einzelnen ist, dass die Übersetzung gelingt. Es gilt, einen Arbeitsprozess in Form einer soziotechnischen Netzwerkarbeit zu etablieren, in dem branchenspezifisches Wissen iterativ aufbereitet und den Entwickler*innen in umsetzbarer Form übergeben wird. Dies ist kein „nice to have“, sondern entscheidend, um mit den stetigen Veränderungen zurechtzukommen, Softwareentwicklung für eine effiziente Organisation nutzen zu können und sich nicht in eine Abhängigkeit zu begeben. Wie eine Abhängigkeit vermieden werden kann, zeigt eine der Fallstudien: Statt eine große Softwarelösung auszuschreiben und dem Anbieter ausgeliefert zu sein, übernimmt bei INTERN das EVU die Softwaregestaltung selbst und lässt die Programmierung auf der eigenen Entwicklungsumgebung von externen Freelancern erledigen.

Literatur

Bundesagentur für Arbeit/Statistik. Online: <https://statistik.arbeitsagentur.de/> (Status February 2023).

Job futuromat. Online: <https://job-futuromat.iab.de/> (Status February 2023).

SAP: Partnerfirmen der Kategorie „Solution Building“ und „Consulting Services“. Online: <https://www.sap.com/partners/find.html> (Status 26.04.2023).

Stiens, Teresa: Verwaltungschaos – Warum Deutschland bei der Digitalisierung nicht vorankommt. Tagesspiegel, 29. January 2023.

Image Description: Software surrounds and permeates working life in software-based organizations. For this image, which was generated by means of artificial intelligence, Gesine Born used the Midjourney software. She entered the following keywords: „coding technology on computer screen, in the style of dense composition, tiny people walking in it, creative commons attribution, fujifilm pro 400h, uhd image, anglocore, text-heavy, sopheap pich --ar 281:187 --v 5.1 --s 50”.

--

20. Juni 2023

Das Werk ist eine Open-Access-Publikation und steht unter der Creative-Commons-Lizenz Namensnennung 4.0 International (CC BY 4.0: <https://creativecommons.org/licenses/by/4.0/>).

