Sonnenholzner, Johannes

Article

Digitalization needs communication: Why software shaping is not just a technical issue

WZB-Mitteilungen: Quartalsheft für Sozialforschung

Provided in Cooperation with:
WZB Berlin Social Science Center

Suggested Citation: Sonnenholzner, Johannes (2023) : Digitalization needs communication: Why software shaping is not just a technical issue, 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-25642.pdf

This Version is available at:
https://hdl.handle.net/10419/327843

Originally published July 20th, 2023 in
https://www.wzb.eu/en/article/digitalizationneeds-communication

Digitalization needs communication

Why software shaping is not just a technical issue

By Johannes Sonnenholzner

Organizations in all industries are faced with the task of programmming industryspecific software or having it programmed. This requires knowledge about the application area of the future software. Developers need to know what they have to program. The part of software development in which the desired software for an application area is negotiated is called software shaping. It hasn't received much attention in previous research. Johannes Sonnenholzer has taken on the topic.

"First, people in the administration tried to teach tax law to IT experts; then, they tried to turn tax law experts into programmers," says the head of a financial services provider for the Federal Central Tax Office. According to a recent TAGESSPIEGEL article, neither approach was satisfactory in terms of developing functioning software.

Programming industry-specific software, or having it programmed, is a major challenge faced not only by administrations but by organizations in all industries. This requires knowledge about the application area of the future software. Developers need to know what they have to program. In the following, I will refer to the part of software development in which the desired software for an application area is negotiated as software shaping. It hasn't received much attention in previous research. Most studies focused on the application of software without acknowledging that the actual technology behind that software is the computer — a universal machine in need of programming.

Once software becomes key to how people work, organizations need to be reconsidered. In many organizations, members still talk as if they were not actually already a softwarebased and hence a sociotechnical enterprise. And for any software-based organization, this involves a division of labor between software application and software development. In terms of organizational theory, this means zooming in on those processes that ensure that users get the software (landscape) that works for them: software shaping. I will begin with a few general remarks and key terms, illustrate these with the help of three case studies, and then connect them to the notion of sociotechnical network-labor.

Software development and shaping is not a niche topic: in Germany, there are 304,005 programmers (2022), 96,486 ERP consultants (2012), and 22,734 IT project managers (2012). IT project managers and ERP consultants implement or enhance standard software packages. The largest provider (and the most valuable company in Germany in terms of

stock market value) is SAP: its software solutions are implemented, adapted, or expanded by 457 partner companies in Germany and 2,796 worldwide.

IT project managers and ERP consultants are not the only knowledge workers involved in software shaping between application and development. Other roles include requirements managers, product owners, scrum masters, application managers, digitalization managers, key users, or process managers, to name just a few. In addition, software shaping has its own software-based tools (e.g., issue tracking systems) and its own processes (e.g., scrum, IT projects).

This multiplicity of roles and responsibilities points to one of three reasons why software shaping is not merely a technical issue.

1. Software shaping is a distinct work process that, just like any other work process, must be organized. Knowledge workers act in sociotechnical structures with the goal of developing concepts or generally finding out what the software should be able to do. In this process, the essential tools are knowledge and communication.

2. The work process must be established in different constellations and must accommodate different interests. Like other work processes, software shaping is subject to economic indicators. The more user perspectives are to be considered and the more market-like and hierarchical relationships exist between application and development, the more coordination is required to establish software shaping.

3. There are two core problems that must be solved in the software shaping process:
   the problem of interdisciplinarity and the software technical reference level problem. What does this mean? In work settings featuring extensive and constantly changing domain knowledge (e.g., tax law), work must be interdisciplinary. It is rare and often impossible for one person to know everything and write software alone, given the high degree of specialization both within IT and within the industry in question, its contexts of application,

and its organization-specific workflows. The second problem exists, because if software is designed for a specific application, decisions have to be made about the softwaretechnical cut and the software-technical orientation of the organisation. Do they want something individual, or do they want to create a standard that others might use as well? Should the organization be built on existing (standard) software, or should it develop its own software? Both questions belong to the software technical reference-level problem. The latter is ubiquitous in the age of cloud computing, which allows organizations to virtually access and use countless applications and IT experts for their own work context. This is essential to issues of efficiency or even survival in a competitive environment. Digital start-ups and large corporations such as Amazon are successful because they develop their own software from the outset, which they use for their respective industry-specific needs, while at the same time even offering the software to competitors from within the industry. Then again, many companies and public administrations rely on providers of standard software because they neither have the necessary skills nor the resources to do any programming of their own, or do not see any added value in doing so.

The energy industry, with its vast diversity of organizations (e.g., corporations, large and small municipal utilities), many regulatory changes, and the energy transition, provides a great example for illustrating how the possibilities of software shaping are used to respond to the need for change. To that end, I present three of seven exemplary case studies I have been studying in my doctoral dissertation at the WZB. I refer to them as KOOP, INTERN, and STARTUP. They all require cooperative collaboration to shape industryspecific software. But do they manage to solve both problems: that of interdisciplinarity and that of software technical reference level?

In KOOP, an IT service provider (with both industry and IT expertise) organizes a cooperative software shaping process involving multiple energy companies (with more industry than IT expertise). At regular meetings, participants discuss individual requirements, negotiating whether these should become part of the industry-specific standard, which is then used by all participating companies. The company users subordinate themselves to the standard software created in this way. It has been a challenge to reconcile the interests of the various companies. The meetings are moderated by a professional mediator.

“I think one of the biggest learnings is that the biggest challenge, one of the big cost factors, is general coordination and discussions with the customer [energy companies]. Requirements management and everything that has nothing to do with technology, [...] but really talking to each other and coordinating and somehow agreeing on a common denominator. [...] We found a solution for that, even hired an outside moderator at some point, whom we still use.” (KOOP respondent)

As a compromise, there may be individual deviations from the standard, for which companies must pay accordingly. Given that this is a market relationship, the energy companies constantly check if there is a cheaper or more innovative software provider. At the same time, they appreciate the benefits of long-term relationships (especially the shared knowledge base).

INTERN involves an energy company developing its own industry-specific application. Since multiple departments use the same software, a requirements group was set up for meetings and coordination. Given that hundreds of users are affected, there are staff in the departments writing additional requirements. However, they cannot create the software as they see fit because hierarchies stand in the way of a comprehensive design:

"This means that processes that actually run horizontally are constantly cut off by some hierarchies, and the manager just says, 'That's none of my business. I don't care about the interface; it doesn't interfere with my work." (INTERN respondent)

Only a fraction of users is involved in the software shaping. In addition, their position in the organization of the energy company changes through the software shaping work process. According to one manager, the goal is to reduce hierarchies and establish a sociotechnical process organization to which the developers are then directly assigned. In this way, interdisciplinary collaboration could be made more efficient, and the potential of software shaping could be fully exploited.

In STARTUP, the organization was built from the outset to accomplish its industry-specific purpose (emissions trading management for e-mobility) through software development. The primacy of software development applies. The entire organization is designed to enable interdisciplinary exchange and iterative development of the software — in a network: no hierarchies, no organizational and departmental boundaries, no competitive mindset. Requirements are written in regular and spontaneous meetings. Relationships are open and cooperative:

"Everyone's just pulling in the same direction, without strong personal sensitivities behind it, without any ego stories." (STARTUP respondent)

Users are left with the work that cannot be automated through software development.

Overview of the case studies:

Cases	Basic coordination	Networks of software shaping	Software technical reference level
KOOP	Market	IT service provider: organizes requirements meetings with energy	Negotiated standard software
INTERN	Hierarchy	IT department: organizes requirements meetings with departments	Individual software
STARTUP	Network	Team meetings and chat groups	Primacy of software development

Software shaping is a specific type of work and organization that may be conceptualized as sociotechnical network-labor, characterized by knowledge workers acting in a rolebased and situational manner. Typical expectations include: to be cooperative and selforganized, to engage with software, to deal with non-knowledge, and to adapt to changing interpersonal constellations. In addition, relationships are key: to work as equal partners, creating interpersonal alignment at the operational and strategic levels. In addition, expectations must be aligned in regular intervals, and conflicts must be resolved in a professional way (using mediators, if necessary). Finally, sociotechnical network-labor requires a feedback-driven, iterative workflow drawing on a network of knowledge workers (distributed across organizations, departments, teams) collecting requirements in an interdisciplinary manner. This helps ensure that the developed software meets the expectations of the application domain and makes the most of the technological possibilities.

Conclusion: Developing functional software is not about tax experts becoming programmers or programmers becoming tax experts. The key is not individual knowledge but a successful translation. It is about creating a work process designed as sociotechnical network-labor in which industry-specific knowledge is iteratively processed and shared with developers in a form they can implement. This is not a niceto-have but an essential task for keeping up with constant change, using software development for an efficient organization, and steering clear of becoming dependent. One of the case studies above shows how dependency may be avoided: Instead of putting a large software solution out to tender and being at the mercy of the provider, the energy company in INTERN takes software shaping in its own hands and has external freelancers do the programming on its own development platform.

Literature

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”.

--

20/6/2023 MSt