6-17-2021

Generic Enterprise Software Implementation as Context for User-Oriented Design: Three Conditions and their Implications for Vendors

Magnus Li
University of Oslo, magl@ifi.uio.no

GENERIC ENTERPRISE SOFTWARE IMPLEMENTATION AS CONTEXT FOR USER-ORIENTED DESIGN: THREE CONDITIONS AND THEIR IMPLICATIONS FOR VENDORS.

Research paper

Magnus Li

Department of Informatics, University of Oslo

magl@ifi.uio.no

Abstract

User-oriented approaches to designing IT are consistently promoted by academic and practitioner literature. These orients the design process around the specific practices and needs of end-users to build usable and relevant systems. However, an increasingly relevant but little explored context for the design of IT is that of implementing generic enterprise software solutions. In this paper, we explore conditions for user-oriented design during the implementation of generic enterprise software. Our empirical data is based on an ongoing engaged research project, where we work with the vendor of a global generic software solution and a set of implementation specialist groups (ISGs). Together, we explore how user-oriented design during implementation of the software solution can be supported and promoted. The paper contributes to the body of knowledge on the design and implementation of generic enterprise software by identifying several challenges and three conditions for user-oriented design in this context. The conditions are: the project configuration, the implementation practices of the ISGs, and the features and adaption capabilities of the generic software solution. We further contribute by discussing their implications for vendors who want to support and promote user-oriented design during implementation of their software solutions.

Keywords: generic enterprise software implementation, user-oriented design, conditions, implementation-level design.

1 Introduction

User-oriented approaches to design and innovation, such as User-Centered Design and Participatory Design emphasize basing the design of IT on the practices and needs of specific end-users. These approaches are consistently promoted by research (Baxter & Sommerville, 2011; Ellingsen & Hertzum, 2019; Gulliksen et al., 2003; Mumford, 2006) and practitioner guidelines and literature (digitalprinciples.org, 2019; gov.uk, 2019; D. Norman, 2013). Meanwhile, a significant portion of the IT systems implemented in organizations are not built 'bottom up' based on the specific practices and needs of singular organizations. They are rather designed and developed as comprehensive generic software solutions that aim to serve a diverse audience (Berente et al., 2019; Pollock et al., 2007; Sykes & Venkatesh, 2017). Examples are Enterprise Resource Planning Systems (ERPs), and Electronic Health Record Systems (EHRs). Two increasingly relevant contexts for designing IT are thus that of building generic solutions (Pollock et al., 2007), and that of implementing these solutions into specific organizations by configuring them to local needs (Bansler, 2021; Dittrich, 2014; Ellingsen &

Hertzum, 2019; Martin et al., 2007). In this paper, we refer to the latter context as implementation-level design.

Traditionally, generic enterprise software solutions have been described as inflexible for local adaption, and the process of implementation-level design as one of adapting the organization according to the software, rather than building software according to their specific needs (Kallinikos, 2004). As a result, Information Systems (IS) literature argues, generic solutions often fail to meet the expectation of organizations and that the consequences for end-users and the organization as a whole may be adverse (Berente et al., 2019; Soh & Sia, 2008; Strong & Volkoff, 2010). However, in recent years, vendors of generic enterprise software are "opening up" their solutions for design and innovation by third-party actors (Farhoomand, 2007; Wareham et al., 2014). Design and innovation are no longer reserved for the vendor firm but supported and encouraged for a larger 'ecosystem' (Dittrich, 2014; Wareham et al., 2014) or 'design network' (Koch, 2007) of partner organizations that specialize in implementing the solutions on behalf of user organizations (Foerderer et al., 2019). Vendors move from building monolithic solutions or 'packages', to building platforms that are advertised as highly configurable and extendible to serve heterogeneous needs (Foerderer et al., 2019; Rickmann et al., 2014). This increasing emphasis on supporting design and innovation outside the boundaries of the vendor appears to offer the potential for more user-oriented design and innovation based on the needs of individual user organizations than what is earlier described in the literature. Still, to the authors' knowledge, no systematic analysis of the conditions for user-oriented design processes in the context of generic software implementation exists. Further, there is no literature examining how vendors may support and promote such design during the implementation of their solutions.

This paper addresses this gap by examining the following research questions:

- What conditions affect the potential for user-oriented design in the context of generic enterprise software implementation?
- What implications do these conditions have for vendors who want to promote user-orientation during implementation of their solutions?

We explore our two questions based on data collected through an ongoing engaged research project (Li, 2019), where we collaborate with a generic health software vendor and a set of implementation specialist groups (ISGs). The ISGs are independent consultancy firms that specialize in implementing the software for user organizations. The software solution, named DHIS2, is designed to support collection, and use of routine health information within organizations such as health ministries and non-governmental organizations. During the last two decades, the software has been implemented to serve a range of health-related use-cases and is now used by organizations in more than 80 countries. These user organizations have different practices and needs that, in many cases, would be best supported by IT solutions with custom functionality and user interfaces. Due to differences in needs, it is challenging for the vendor to design generic functionality and user interfaces that are considered usable and relevant across the vast audience of user organizations. A strategy the vendor increasingly pursues is that of supporting design and innovation based on specific organizational needs during implementation-level design. Part of this strategy is to make the solution configurable and extendible, and promoting the use of user-oriented approaches to design by the ISGs specializing in implementing it. In our work, we have, however, found that there are several conditions that make the implementation of a generic solution a challenging context for user-oriented design.

The rest of the paper is organized in the following manner: We first look at existing literature related to user-oriented design in the context of generic enterprise software implementation. We then describe our research approach before we present our analysis, where we examine the process of implementation-level design and challenges related to user-oriented design. In the discussion chapter, we articulate and discuss three conditions and their implications on vendors who seek to support and promote user-oriented design during implementation of their solution.

2 Related Research

The literature on user-oriented design in the context of generic enterprise software design and implementation is scarce, but with a few exceptions. We will first define what we mean by "user-oriented design" before turning to generic enterprise software implementation as a context for design.

2.1 User-oriented design

We employ the term 'user-oriented design' to refer to approaches to designing systems and technologies that orient the design process around the end-users needs and well-being, with the aim of making systems that are perceived as usable and relevant. A myriad of such approaches is conceptualized in IS and HCI literature. Readily available examples include User- or Human-Centered Design (Gulliksen et al., 2003; D. Norman, 2013), Participatory Design (Simonsen & Robertson, 2012), Activity-Centered Design (Gay & Hembrooke, 2004), Socio-technical Design (Mumford, 2006), and Usability Engineering (Nielsen, 1994; Rosson & Carroll, 2002). Although all are oriented towards the end-users of technology, they vary with regards to the ends and means of doing so, and the scope of what is to be designed (Kujala, 2003). For instance, Participatory and Socio-technical design are based on the idea that end-users should be involved in the decisions regarding the technology to be used in their work. Hence, a key aim of the process is to empower workers by giving them a voice in the design process. Means to achieve this naturally rely heavily on involving users in decisions regarding the IT project (Mumford, 2006; Simonsen & Robertson, 2012). In contrast to Participatory Design, the primary end of User-Centered Design is to build technology that is usable and relevant for end-users. Means of doing so do not necessarily include involving end-users in all decisions regarding the project, but often instead focus on understanding their existing practices and needs through interviews, observation, and iterative, evolutionary prototyping and evaluation (Norman, 2013; Norman, 2005).

Albeit differences in the ends, means, and scope of various user-oriented approaches to design, they share some key principles:

1. The features of technology are designed based on an understanding of the practices and needs of end-users in concrete contexts. Objectives of the design process are to establish 'what is the right thing to build' i.e., fundamental questions about the IT artifacts form and function, and 'building the thing right', i.e., defining the right form of the artifact (e.g., user interfaces, functionalities)
2. Iterative design and development with evolutionary prototyping and frequent end-user evaluations of form and function. Prototyping should ideally start with low-fidelity prototypes to ensure that the project avoids committing to a specific solution at an early stage. Rather, problems and multiple potential solutions are explored as the project evolves.
3. Emphasis on understanding the practices of and/or involving end-users in the design process, either in an informative role (as in User-Centered Design), or as active participants in fundamental decisions about the project and the artifact(s) of focus (Damodaran, 1996).

These principles form the basis for our understanding of user-oriented design. Accordingly, what we seek to identify in this paper are conditions that affect if, or to what degree, processes following these principles can take place. While many different user-oriented approaches to design are conceptualized, existing literature focuses little on the conditions that must be in place for such design processes to unfold (Baxter & Sommerville, 2011; Edwards et al., 2010; Svanæs & Gulliksen, 2008; Zahlsen et al., 2020). For instance, as noted by Svanæs & Gulliksen, (2008), the two ISO standards describing how user-centered design should be carried out *"describe an ideal situation where there are no obstacles to UCD, except for a possible lack of skills at the developer side. Although the two ISO standards on UCD are very useful as reference frameworks and ideals, they do not deal with the heterogeneous nature of real-world UCD projects, and the potential obstacles to user-centered design."* A few studies report how the 'boundary conditions' (Zahlsen et al., 2020) of the context where the design process takes place strongly impact the form of 'user-orientedness' that is relevant and possible. Such boundary conditions include internal factors in the developer organization, such as their structure, software engineering practices, and 'usability maturity' (Earthy, 1998). Further, it may include aspects of how

the project is structured with its actors, defined goals, and expected process (Martin et al., 2007). Others argue that a significant challenge is that user-oriented approaches are incompatible with widely used software engineering methodologies (Baxter & Sommerville, 2011).

2.2 Enterprise software implementation as the context of design

The development and implementation of generic enterprise software as a context of design differs from that of bespoke development (Li & Nielsen, 2019), often assumed by user-oriented approaches (Edwards et al., 2010). On the generic level of design, the vendor deals with significantly diverse and potentially incompatible needs when attempting to support a large audience of organizations (Sia & Soh, 2007). Design is reported to unfold as a process of aligning the needs of organizations seen as strategically important (Gizaw et al., 2017; Pollock et al., 2007), while neglecting needs that are relevant to only one or a few (Koch, 2007; Sia & Soh, 2007). Implementing the software into a particular user organization can be seen as another level of design, which we here refer to as implementation-level design (Li & Nielsen, 2019). During this process, the solution is configured and possibly extended according to the particular circumstances of the user organization (Sommerville, 2008). However, implementation-level design is based on the features and adaption capabilities of the generic solution, which do not provide endless flexibility for local adaption (Martin et al., 2007). Instead, the solutions are often adaptable in specific ways, dependent on the configuration facilities embedded by the vendor (Bertram et al., 2012). When generic features and adaption capabilities fall short, the source code of the software may be modified, but with the costs of additional work related to upgrading the software to new versions provided by the vendor (Hustad et al., 2016; Sestoft & Vaucouleur, 2008).

Research on user-oriented design in the context of generic enterprise software design and implementation is scarce, but with some exceptions. A few studies explicitly discuss user-oriented approaches such as User-Centered Design (Vilpola, 2008) and Participatory Design (Magnusson et al., 2010; Pries-Heje & Dittrich, 2009) as means of driving implementation-level design processes. However, these studies are more concerned with the use of such approaches to increase the user acceptance of the generic solution, rather than using them as the engine to design and innovate IT solutions based on insights into the end-users' particular practices and needs. There are few reflections on the conditions that affect the potential for conducting user-oriented design as we defined it in the previous section. Other studies discuss how design flexibility, as touched upon above, is a key challenge when addressing issues of usability and end-user relevance in implementations (Martin et al., 2007). Also, extendible platform architectures have been discussed as enabling "local" user-oriented design during implementation, as it allows custom applications to be built on top of the generic solution to address implementation-specific needs (Roland et al., 2017). Supporting custom app development also appears as a strategy followed by prominent vendors such as SAP to facilitate design and innovation during implementation of their widely used ERP solutions (Farhoomand, 2007; *SAP Fiori*, 2020)

To summarize, existing research emphasizes the importance of user-oriented approaches to design. Yet, one of the most common means of introducing technology in organizations is by implementing generic software solutions. We see the relatively limited knowledge on user-oriented design in the context of implementation-level design as an important gap in existing research.

3 Research Approach

We report from an ongoing engaged (Mathiassen & Nielsen, 2008; Van de Ven, 2007) research project (Li, 2019) where we collaborate with a software vendor – referred to as the 'core team' and a set of implementation specialist groups (ISGs). First, we briefly introduce some key information about the software solution and actors of focus before describing our methods for data collection and analysis.

3.1 Case – DHIS2, the core team, and the ISGs

We follow the generic health information software DHIS2. The "core team", situated in Oslo, Norway, is in charge of designing, developing, and maintaining DHIS2 as a generic solution. The DHIS2 is used by a diverse audience of organizations across more than 80 countries. In its primary use case, DHIS2 supports the collection, storage, and presentation of health management information. Due to its flexible and configurable data model, it is increasingly implemented for use in domains beyond health management, such as logistics management and education management. The generic solution comprises a software "core" with a configurable data model, and a set of "generic apps". The apps provide functionality and user interfaces that support activities common among end-users across implementations. Examples are reporting data and displaying information in reports, graphs, and maps. The core team decides what features to include in the generic solution by identifying shared needs across the user audience. Their means of doing so is beyond the scope of this paper. To support the adaption of the software when implemented in specific organizations, the core team embeds an array of configuration facilities into the software. This allows specific implementations to define certain aspects of the solution according to their particular circumstances. Examples include what data to report, when, where, and by whom, and how this data is to be presented in graphs, maps, etc. The generic solution is also extendible, meaning that so-called "custom apps" can be developed during implementation to extend the functionality and user interfaces beyond what is provided by the generic apps.

The ISGs we collaborate with in our study are independent consultancy firms that are contracted by (future) user organizations. Together they configure and extend DHIS2 according to their particular needs. The inner team of the ISGs typically includes what is called "implementers" in charge of working with the organization to identify their requirements and configure DHIS2 accordingly. The ISGs often also have a group of developers that build custom apps when needed.

3.2 Data collection and analysis

Our collaboration with the core team and the ISGs involves both diagnostic and interventionist research, and is interpretive in nature (Klein & Myers, 1999). As DHIS2 experiences increasing adoption by user organizations with diverse practices and needs, the usability and relevance of the generic solution is becoming an increasing concern. With this problem as the basis, we started exploring the nature and challenges of designing (with) DHIS2 during implementation. Our engagement started by participating as 'attached insiders' (Myers, 2019; Van de Ven, 2007) in a large implementation project in India together with an Indian ISG from August 2018 – November 2019, where we tried to address various usability issues that had been documented in the implemented DHIS2 solution. We participated in meetings (6), and in planning-activities, and conducted interviews (8) with stakeholders in the user organization, and the ISG team. Engagement in this project gave us insights into the process of implementation-level design and highlighted several challenges forming the basis for the findings presented in this paper. For instance, challenges found relate to how the defined scope of the project, and the mandate of the ISG therein restricted the relevance of and ability to interact with end-users to diagnose usability issues.

To get a richer understanding of the broader DHIS2 implementation practices beyond the Indian ISG, we continued to explore the nature of implementation-level design of DHIS2 by visiting three ISGs in Tanzania, Mozambique, and Malawi from May 2019 to January 2020. During these visits, which typically lasted for one week, we conducted interviews with implementation experts (6) and software developers (4). The interview subjects had 3 – 12 years of experience with DHIS2 implementation. We also arranged focus groups (3), including the whole or most of the ISG teams (6 – 12 participants) at their offices. The aim during interviews and focus groups was to understand the implementation-level design process, what activities it constitutes, and if, how, and when the design process is (not) oriented towards end-users. From March – November 2020, we also conducted interviews (4) with implementers and developers over Zoom with ISGs in the United States and Uganda, also focusing on the process of implementation-level design and user-orientation in the activities it constitutes. Engagement in

India, combined with interviews and focus groups with other ISGs allowed us to identify the common traits and differences between implementation practices and projects, and a variety of challenges they face related to user-oriented design. The findings are used as a basis to inform further diagnostic activities, and potential interventions in future phases of the research project. Data is collected and analyzed concurrently in our project. Data is documented through field notes (during participant observation and focus groups), transcriptions of interviews, and documents collected in the document analysis. An overall research diary is kept throughout the process, summarizing patterns and findings relevant to the various (and developing) research questions. The analysis is abductive in nature (Tavory & Timmermans, 2019; Van de Ven, 2007), comprising cycles of inductive analysis of empirical data (e.g., coding and developing themes related to practices and challenges which are presented in the case analysis chapter), and identifying similar phenomenon and related concepts in relevant IS literature (e.g., design and implementation of generic enterprise software, user-oriented design, enterprise software ecosystems).

4 Case Analysis

We now turn to our analysis of implementation-level design as a context for user-oriented design. We begin by looking at how the process unfolds, before we highlight some key challenges.

4.1 The process of implementation-level design

A typical implementation-level design process starts when the ISG is awarded a contract for a project following a tender process. The initial negotiation of how the project will be configured begins between the user organization, the ISGs, and potentially other involved actors. What we here name the 'project configuration' refers to how the project is defined in terms of the scope of the problem to be addressed and potential solution(s), structure and process of the project, and the mandate of different actors therein. The starting point in many projects is that an organization already has an existing digital or paper-based (or partly both) information system that supports the collection and presentation of some sort of data, often related to health management. The aim of projects of this kind is to design a coherent digital system based on DHIS2 to replace paper-based data reporting tools and by integrating various fragmented systems. Although the process of implementation-level design varies between ISGs and projects, we highlight five activities that typically make important parts of the process. These are illustrated in Figure 1, and we will use these to structure the first part of our analysis. We stress that these are not discrete steps of a linear process but rather activities that can be enacted several times, in various order, and often concurrently.

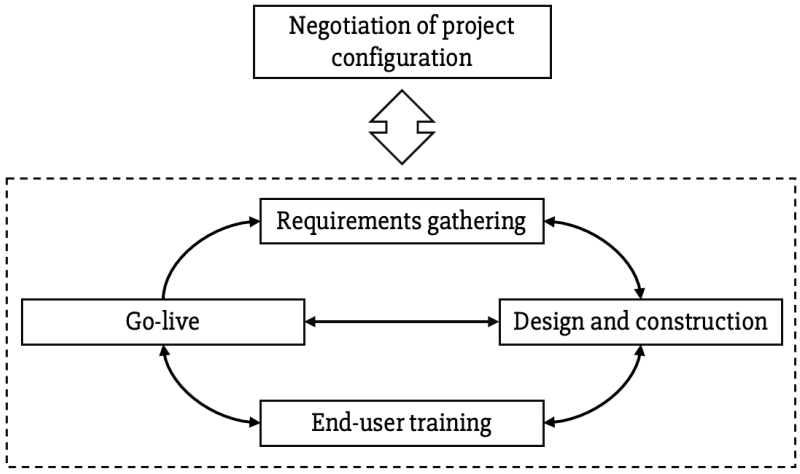

Figure 1. Five key activities part of the implementation-level design process of the ISGs

4.1.1 Initial negotiation of project configuration

Already in the initial negotiation of the project configuration, the generic features of DHIS2 play an important role in guiding and framing the discussions of the shape and form of the software solution for the particular user organization. The ISG tends to rapidly set up a running demo of DHIS2, including some configurations that show how the system could look and behave in the respective user organization. The ISGs explain that this is beneficial as it allows them to establish realistic expectations and show quick results early on, through what, at least, appears as a fully working prototype.

If, how, and which end-users are to be involved in the process is also partly established in the initial project configuration. Some ISGs explain how they push for making visits to health facilities and involvement of users in evaluations of the system part of the agreed implementation process. The relevance of engaging with end-users is often something that has to be advocated to the project managers of the user organization. For instance, the Mozambique ISG lead explains how he sees him and his team as *"fighting the battle on behalf of the end-users",* when negotiating if, when, and how end-users should be consulted and involved in the implementation process. In Malawi, they often try to negotiate the User-Centered Design methodology (as defined by the ISO standard) as a formal part of the implementation process. In contrast, other ISGs are not particularly concerned with promoting user-oriented activities, and prefer to primarily rely on communication with the project managers of the user organization. An implementer within one of the ISGs that falls under the latter category explains that user involvement is "painful" as the end-users seldom "agree" with the prototypes they show them. More so, he argues, users *"often quit their job even before the solution is launched anyway"*. The implementer explains how he sees end-user interaction as more about convincing the users to use the system, rather than for the end-users to give feedback for improvements – *"it's the [top level project managers] who decide anyway, it's them we have to please"*. He further explains that in the case of doing end-user interaction such as going to health facilities to talk to users, it's mainly when the project managers of the user organization ask for it explicitly.

4.1.2 Requirements gathering

The nature of how requirements are gathered and the role of the ISG in this activity differs substantially between projects. In some projects, the ISG only acts as a "technical partner" and merely a receiver of requirements defined by the user organization themselves or other consultancy firms. In other projects, the ISGs may be responsible for collecting, and defining the requirements throughout the development process. In the latter case, there is some variation among the ISGs and the projects on how requirements are collected and established. Some ISGs express the importance of doing extensive visits to the end-users' context, to *"map out current practices, tools, infrastructural conditions"* (Mozambique implementer), and other relevant aspects. As articulated by one implementer, *"If you're not doing it [field-visits], you go in blind [to the development process] […] you need to understand the context"*. Others rely exclusively on communication with project managers of the user organization.

Albeit differences in the means of requirements gathering, it is striking how similar the requirements gathering activity is in terms of what the ISGs look for. Either when visiting end-users in their context, and/or communicating with project managers, the focus is almost exclusively directed towards:

- what data is currently and/or in the future is to be reported, how and by whom
- what data is currently and/or in the future is to be used, how and by whom

"We identify the data elements, then we try to implement the data outputs and present them to the client […] We iterate between input and output several times" (Implementer India ISG).

The findings are documented along the requirements gathering process, and will normally be followed by new negotiations of the project configuration together with the project managers of the user organization (e.g., updating objectives and timelines).

4.1.3 Design and construction

The design and construction activity involves prototyping and constructing a working IT solution based on the collected requirements. In essence, this is about identifying how requirements can be accommodated by the generic features of the DHIS2, and if any custom development is required. Since most needs within projects are addressed by configuring DHIS2, the software is often used as a working prototype throughout the design and construction process. Regular evaluations of the prototype with representatives from the user organization are commonly carried out during the process to evaluate and adjust how requirements are to be addressed in the solution. Few of the ISGs involve end-users in this iterative process, and rather rely on frequent communication with the project managers of the user organization. A large portion of the process internally within the ISGs' technical team is to discuss how to configure the data model to best support the required data input and output. The configuration facilities of DHIS2 are described by the ISGs as highly supportive in this process, allowing for easy definition of organizational hierarchies, data elements to be collected, the layout of the reporting forms, and the various forms of data presentations that are needed.

When requirements and needs warrant changes or additions to functionality and user interfaces, the source code of the generic apps could be customized, resulting in a custom app for the specific implementation. Also, custom apps may be built from scratch. Through customization of generic apps, or by developing apps from scratch, the ISGs thus have extensive flexibility to shape and extend the generic features of DHIS2 with novel functionality and user interfaces. However, what typically limits the use of this flexibility is that it is costly to develop the apps (i.e., writing the code, designing the user interfaces, etc.) and maintaining them over time. An implementer reflects on this with an example from a recent project, where several modifications to the functionality and user interfaces were wanted, but avoided: *«If we had started to do modifications [to the generic features], we had lost the ability to update. That is, the benefit of being on the platform, and then you're suddenly alone. It gets difficult to update, and you cannot be part of getting new features together with the others.»*

4.1.4 End-user training and Go-live

ISGs tend to play a key role in end-user training. Users are gathered in workshops where they are trained to use the constructed solution. Some ISGs describe the activity as one of training and "convincing" the end-users in using the solution. Others regard the experiences from the trainings as valuable learnings related to the usability of the solution. Issues discovered may be addressed through more design and construction work. In this case, discoveries that could warrant more design and construction would often require a renegotiation of the project configuration. Finally, the solution is introduced for use in the organization, either starting with a small pilot with a few use-sites, or by introducing the solution to the whole organization all at once. The role of the ISG seldom ends here, and instead continue with maintaining and improving aspects of the solution, or providing "refreshment training" of end-users for several years to come. The further development of the system forms new cycles of the five activities outlined.

4.2 Some prominent challenges related to user-orientation

We now highlight some prominent challenges that came up during data collection and analysis.

4.2.1 Balancing flexibility and predictability

Fundamental to user-oriented design is that end-user practices and needs should inform the solutions being built. The understanding of the problem(s) end-users face, and potential solutions to these should thus evolve as the process is carried out. For this to be possible, there must be some flexibility in the project to (re)define requirements based on issues and needs that are discovered along the process. On the other hand, several of the experienced ISG implementers explain that too flexible scopes could end up with "scope creeps" where emerging requirements make the project unmanageable. A

major challenge for the ISGs is thus to balance flexibility versus predictability in the scope and requirements for the projects. Flexibility is needed for user-oriented processes where problems and solutions are explored as the process evolves, while predictability in terms of what the client organization will get out of the process, and the person-hours and workload it implies for the ISG is important. In many projects, most or all the requirements are defined and agreed upon in detail prior to any design and construction is initiated, and before any end-users are exposed to prototypes or even consulted regarding their practices and needs. A Mozambique ISG implementer reflects on a project: *"all requirements were defined from the outset – we had no room for requirements to emerge during the process"*. At the other end of the spectrum, some ISGs have worked with projects that are very flexible, as explained by the Malawi ISG lead: *"Sometimes they [the user organization] don't know what they want, but they know that they want something … this gives us more room to negotiate how the process should look like, and for the solution to emerge over time"*. The concern in the latter situation is that the flexibility might as well be misused by the managers in the user organization to *"constantly change and expand the scope"* (India ISG implementer) of the project. Flexibility might end up being exploited by project managers rather than providing room for problems and solutions to be explored and emerge over time through end-user interactions.

Some of the ISGs explain their strategies for dealing with this. The Mozambique ISG lead tells how they typically start with a rather strict and defined scope, but as the project moves along and needs that can be translated into useful features in DHIS2 are identified, he works to sell these ideas in to the user organization and expand the project scope "bits by bits". As he articulates, *"it's up to [him/them] to push to expand the scope based on opportunities along the way".* The Malawi ISG lead explains that during negotiation of the project scope and their mandate, he attempts to define *"some pockets for refinement of requirements"* along the process.

4.2.2 "Convincing" user organizations of the need for end-user inquiries

A challenge reported as common by the ISGs is that the user organization's IT project managers do not appreciate the relevance and importance of end-user-oriented activities in the implementation process. And more so, what it will involve in practice. As explained by the lead implementer in the Mozambique ISG, in many of their projects, *"the agreement with [the clients] often do not allow us to go to the field"*, or the budget limits user interaction activities. She explains that they sometimes finance field trips themselves, and hope that the findings will feed into new innovations that can be negotiated into the project at a later point. The project we were involved with in India provides an illustrative example. We requested the client managers for access to visit health facilities to observe and interact with end-users to better understand their practices, challenges, and needs. We wanted to use this information as basis when working to address usability challenges reported by the user organization. Our request did, however, meet significant resistance from the client. One of the IT project managers of the user organization explained his hesitation: *"I feel it is more important that the [team of researchers and India ISG] support on the technical part - solving the problem, rather than understanding more problems".* At the time of our inquiry, the project also suffered from many technical challenges, e.g., related to server performance. From a purely technical viewpoint their concern makes perfect sense. Yet, it illustrates a lack of knowledge – and sufficient clarification from our side on the relevance of working with end-users when addressing challenges related to usability.

4.2.3 The gravity of the generic features and adaption capabilities

To win contracts for projects, it is imperative for the ISGs to be competitive in terms of project costs. The ISGs aim to limit costs by leaning on the generic features of DHIS2, for which development and maintenance costs are taken care of by the core team in Oslo. This means that many needs and opportunities for innovation are deemed too costly to implement in the solution, if not possible to support with readily available generic features. An experienced implementer illustrates the challenge through an example from a prior project: *"They had contracted an interaction design agency to design the app without thinking about DHIS2. They had many great thoughts and ideas, but which was based on hav-*

ing blank sheets. In reality, we had to take tracker [DHIS2 generic module] as a starting point […] in the end; we didn't use much of the design made by the interaction designers."

More so, what is striking is how the generic features of DHIS2 play an important role in the negotiation of the project, the requirements gathering process, and the design and construction activity. In all of these activities, the focus shifts between what is needed in the particular context, and what is possible with the generic features and adaption capabilities of DHIS2 within the given budget and expectations defined in the project configuration. For experienced implementers, their knowledge of the features of DHIS2 forms a lens throughout the project, which seems to direct their attention towards aspects readily configurable, while drawing attention away from what is not. This is why the requirements gathering process in many cases only orients around data input and output needs – this is what can easily be configured using the configuration facilities. As articulated by a Ugandan implementer, through the whole process, they *"always think in terms of DHIS2 features and how to map the requirements to these".* Not only does this affect what the ISGs look for in the requirements gathering process, and what is built during the design and construction phase, but also how projects are configured in terms of the process, and goals of the projects.

The ability to develop custom apps significantly extends the space for design and innovation of functionality and user interfaces. This is, however, costly in terms of development and maintenance and must hence be an explicit part of the project configuration, either upfront, or negotiated as the process moves along. This is not straightforward as DHIS2 often is sold in as a ready-to-use solution. Many ISGs tend to avoid custom app development altogether, as articulated by an implementer: *"We have developed an eye for what is for us and what is not. This makes us avoid getting into projects beyond what [the generic] DHIS2 can handle"*. Being more experienced in app development and negotiate custom development as part of the projects may expand the possibilities that the ISGs see in the given implementation, and hence also expand the space for user-oriented design and innovation. A Tanzania developer reflects on this: *"building apps force us to look at other aspects, such as the kinds of layouts and functionality that best will support the user".*

5 Discussion and Conclusion

We started out with the following two research questions: 1) What are conditions that affect the potential for user-oriented design in the context of generic enterprise software implementation? And 2) what implications do these conditions have for vendors who want to promote user-orientation during implementation of their solutions?

We will first address the first question by articulating and discussing three conditions we see as prominent in our analysis before addressing the second by discussing their implications for vendors.

5.1 Three conditions for user-oriented design

Based on our examination of the implementation-level design process and the ISGs reflection on their practices and challenges, we define three conditions we see as fundamental to the potential for user-oriented design. We argue that the conditions represent what prior literature refers to as *boundary conditions* of the context of design (Zahlsen et al., 2020) in generic enterprise software implementations. The conditions are summarized in Table 1, and discussed below.

First, how projects are configured in terms of their scope, structure, and mandates affect the potential for user-oriented design. We see several examples in the analysis of challenges related to this condition, including the issue of balancing flexibility and predictability, and that of convincing the user organization of the relevance of user-oriented design. Many of the implementation projects have the primary aim of replacing existing paper-based systems. Within such a scope, there might not be much flexibility to explore and address challenges beyond that of what data is to be collected and how this should be presented. We see similar examples in other research, for instance, reporting how 'the con-

tract' strongly affects the possibility to address usability problems, and if, how and when users are involved during implementation-level design (Martin et al., 2007).

Condition	Description
The project configuration	The scope and structure of the project affects the relevance of and possibility to conduct user-oriented design
The implementation-level design practices of the ISG	If and how the practices (i.e., the "usual way of doing things" (Schatzki, 2019)) of the ISG is geared towards advocating, negotiating, and conducting user-oriented design.
The features and adaption capabilities of the generic software solution	The features and adaption capabilities of the generic software shape both the process and the product of implementation-level design

Table 1. Three conditions affecting the potential for user-oriented design during generic enterprise software implementation

Second, we see significant variation in the motivation for and competence in conducting user-oriented design in the ISGs. While some see themselves as "fighting the battle on behalf of the users", others prefer to avoid interaction with end-users during the process. This is possibly the most discussed obstacle to user-orientation in existing literature where low "usability maturity" of the development organization has been pointed out as a frequent challenge (Ardito et al., 2014; Earthy, 1998; Svanæs & Gulliksen, 2008). However, our study points to the importance of not only being motivated and able to conduct user-oriented design, but to negotiate it into the project configuration. We see some examples of how experienced ISGs are able to negotiate for flexibility to incorporate solutions to end-user challenges, even within rather strict project scopes.

Finally, the features and adaption capabilities of the generic software represent a powerful condition in our case. As discussed in existing literature, it largely determines what can be built within the financial bounds of the implementation project (Martin et al., 2007; Mousavidin & Silva, 2017; Sommerville, 2008). Beyond what is discussed in the literature, our findings indicate that the (limited) flexibility of the software not only affects what is built during the design and construction phase. Rather the 'gravity' of the generic features and adaption capabilities of the generic solution shapes how projects are configured in terms of scope and structure, and it acts as a lens during the requirements gathering process. As a lens, the features and adaption capabilities bring attention to the aspects that are supported and can be configured in the solution, while directing attention away from the aspects that cannot. If the adaption capabilities, as in the case of DHIS2, primarily orients around what data can be reported and how it is presented, this inevitably will be the major focus of the design process, leaving other aspects such as novel functionality and user interfaces in the dark. Aspects of the context of use and end-user needs that go beyond what is readily available might be deemed too costly to implement, or even overlooked as the software frames what to look for.

Prior literature discusses how the implementation-level design process is mainly about changing the organization according to the features of the generic software (Kallinikos, 2004; Martin et al., 2007; Vilpola, 2008). A more accurate description based on our findings is that the features and adaption capabilities shape the kind of design and innovation that takes place in the implementation-level design process. It directs attention towards practices, needs, and challenges within the specific user organizations that can easily be addressed with generic features, and leaves other aspects in the dark. It thus enables certain types of design and innovation, while constraining others. Where the generic solution directs focus seems to be manifested in the practices of the ISGs, how projects are configured, and the focus of requirements gathering.

5.2 Implications for vendors

We now address our second research question by discussing the implications of the three identified conditions for vendors who work to promote user-orientation during implementation of their solutions.

Literature report that vendors increasingly work to support and promote design and innovation beyond their own boundaries (Foerderer et al., 2019; Wareham et al., 2014). For instance, SAP appears to invest significant resources in supporting and promoting design and innovation based on the specific needs of user organizations. Their book 'Design Thinking with SAP', and a plethora of resources directly aim to cultivate user-oriented design practices among their partner organizations (the equivalent to the ISGs in our case) during implementation-level design (*SAP Fiori*, 2020). In their words, the aim is to bring implementation-level design with SAP from 'digitization'- to 'digitalization' projects (Prause, 2020). We thus argue that these implications are relevant to vendors of generic enterprise software beyond DHIS2, and other participants within such 'ecosystems' (Dittrich, 2014; Foerderer et al., 2019; Rickmann et al., 2014) or 'design networks' (Koch, 2007).

5.2.1 Implications for Capacity building

To support and promote user-oriented design, the simple advice of "involve the end-users" as, for instance, seen in the Principles for Digital Development (digitalprinciples.org, 2019), and promoting generic methodologies such as User-Centered Design is not sufficient. Rather, the methods and approaches promoted must be apt for integration into the existing practices of the ISGs. The conditions affecting the potential for various forms of user-orientation must be considered in this work. In our project, our studies of how implementation-level design of DHIS2 unfold provide a fruitful basis for developing methods and guidelines that are mindful of the actual context of where they will be used.

One aspect of this, stressed by prior literature, is building motivation and competence to conduct user-oriented design (Ardito et al., 2014). However, in our analysis, we see that the ability to *advocate* and *negotiate* user-oriented design as part of the project configuration is as relevant. In our analysis, we see some interesting examples of strategies employed by some of the representatives of ISGs. For instance, the lead of the Mozambique ISG seems to possess valuable skills in negotiating for user-oriented innovation to emerge, even within inflexible project configurations. Vendors and researchers alike should seek to learn from such experiences and skills to build capacity for others to follow. As projects and ISG practices differ, promoting one method or process to fit all would be of limited value. In our project, we have initiated the development of a design method toolkit, taking into consideration different types of project configurations, and the specific features and adaption capabilities of DHIS2. This will provide ISGs with user-oriented methods that are realistic to integrate into their projects and sensitive to the design flexibility they face with DHIS2. The toolkit aims to build capacity both for conducting and negotiating user-oriented design in implementation projects.

5.2.2 Implications for Software Design - building the 'right' design space

We see that the features and adaption capabilities of the generic solution largely affect the focus and outcome of the implementation-level design process. This means that the vendor, through the features and adaption capabilities of the generic software solution, shapes what is to be of focus, and what can be built during implementation-level design (Bertram et al., 2012; Mousavidin & Silva, 2017). If limited and rigid, the configuration facilities may constrain the innovative capacity of implementation-level design, reducing the process to a standardized 'set-up and install' procedure. This could have dire consequences for design and innovation, and brings resemblance to cautions made by Kallinikos (2004) regarding the effects of rigid IT systems:

Coping with urgent and ambiguous situations often presupposes the ability of responding innovatively to these situations. Such an ability in turn is inextricably bound up with the capacity of reading/framing such situations properly. Rigidly dissociated from framing, action loses its intentional component and tends to degenerate to mindless procedure of execution that may have devastating consequences (Kallinikos, 2004, p. 23)

Inflexible generic enterprise solutions may as such impede valuable IT innovation that could have emerged based on particular user needs within the organization. Implementation-level design is reduced to a 'mindless procedure of execution' rather than serving as an engine for user-oriented design

and innovation. On the other hand, greater flexibility may introduce development and maintenance costs for the individual user organization. If costs are too high, as seen related to custom app development in our case, they may not utilize this flexibility. This represents a challenge but also an opportunity. Given the 'right' features and adaption capabilities, generic solutions can be a fruitful enabler of design and innovation and even be designed to direct attention to aspects of importance to secure usability and relevance for end-users. Platform architectures that give a basis for custom app development seem to be relevant regarding this (Farhoomand, 2007; Foerderer et al., 2019; Roland et al., 2017). However, means of providing flexibility while keeping costs of utilizing it minimal must be found. In collaboration with the DHIS2 core team, we are currently exploring resources that may reduce the efforts needed to develop custom apps. Measures that we explore include user interface libraries, and web components that support designers in assembling apps faster, and which leaves the costs of maintaining the components in the hands of the core team. The ideal result is a space for design that is *generative* (Bygstad, 2017; Msiska & Nielsen, 2017), yet considered sufficiently 'cheap' to utilize. Overall, the aim is to offer a 'design infrastructure' of software features that can be configured and extended to drive and support user-oriented design and innovation. Technical flexibility is seen in relation to the method toolkit and other resources building capacity and giving support to the process of implementation-level design.

5.3 Contributions and Concluding Remarks

In this paper, we have explored conditions for user-oriented design during implementation of generic enterprise software solutions. The contributions of the paper lie in a) our empirical insights into an increasingly relevant yet little explored context for designing IT, and b) our conceptualization and discussion of three conditions with implications for vendors who want to support and promote user-oriented design. We find many of the same challenges underlying our three conditions in existing studies, including studies focusing on bespoke software projects. It is possible to argue that the three conditions we identify are general to any IT project, regardless of being based on a generic enterprise software solution, or if building solutions bespoke. Our findings may, as such, also be relevant to the stream of literature around boundary conditions for user-oriented design in general (Edwards et al., 2010; Zahlsen et al., 2020). Yet, the implementation of generic enterprise software differs from bespoke software development. A core project aim is to limit costs of custom development and maintenance by relying on generic features designed and maintained to be used across many user organizations. Our analysis shows how the "gravity" of the generic software solution pulls on the process of negotiating the project configuration, and the generic features and adaption capabilities act as a lens throughout the implementation-level design process. We argue that the conditions and their implications are relevant to researchers and practitioners engaged with the design of generic enterprise software (Bansler, 2021; Koch, 2007; Mousavidin & Silva, 2017; Pollock et al., 2007), and enterprise software ecosystems (Foerderer et al., 2019; Wareham et al., 2014).

Our study is limited to examining the practices and challenges related to user-oriented design during implementation within one software ecosystem. Studies focusing on the same aspects in other software ecosystems and implementation projects could be useful in elaborating and modifying the conditions and implications presented in this paper. Particularly, following ongoing implementation projects, or examining projects deemed as particularly successful could provide valuable findings.

To conclude, many prominent generic enterprise solutions have a rusty reputation of being difficult to use and constraining the flexibility for user organizations to design and innovate IT based on their specific needs (Berente et al., 2019; Kaipio et al., 2017). While our study identifies challenges that partly concur with this picture, we also see great potential for generic enterprise software solutions as supporting (as opposed to constraining) user-oriented design and innovation. Our study points towards that vendors may get rid of the rusty reputation of their generic solutions by seeing the aim, not as to develop a ready-to-use solution. Instead, the aim could be seen as to provide resources for a 'design infrastructure' supporting efficient user-oriented design and innovation during implementation into specific user organizations.

References

Ardito, C., Buono, P., Caivano, D., Costabile, M. F., & Lanzilotti, R. (2014). Investigating and promoting UX practice in industry: An experimental study. *International Journal of Human-Computer Studies*, *72*(6), 542–551.

Ardito, C., Buono, P., Caivano, D., Costabile, M. F., Lanzilotti, R., & Dittrich, Y. (2014). Human-centered design in industry: Lessons from the trenches. *Computer*, *12*, 86–89.

Bansler, J. P. (2021). Challenges in user-driven optimization of EHR: A case study of a large Epic implementation in Denmark. *International Journal of Medical Informatics*, 104394.

Baxter, G., & Sommerville, I. (2011). Socio-technical systems: From design methods to systems engineering. *Interacting with Computers*, *23*(1), 4–17.

Berente, N., Lyytinen, K., Yoo, Y., & Maurer, C. (2019). Institutional logics and pluralistic responses to enterprise system implementation: A qualitative meta-analysis. *MIS Quarterly*, *43*(3).

Bertram, M., Schaarschmidt, M., & von Kortzfleisch, H. F. (2012). Customization of Product Software: Insight from an Extensive IS Literature Review. In *Shaping the Future of ICT Research. Methods and Approaches* (pp. 222–236). Springer.

Bygstad, B. (2017). Generative innovation: A comparison of lightweight and heavyweight IT. *Journal of Information Technology*, *32*(2), 180–193.

Damodaran, L. (1996). User involvement in the systems design process-a practical guide for users. *Behaviour & Information Technology*, *15*(6), 363–377.

digitalprinciples.org. (2019). *Principles for Digital Development*. Principles for Digital Development. https://digitalprinciples.org/

Dittrich, Y. (2014). Software engineering beyond the project–Sustaining software ecosystems. *Information and Software Technology*, *56*(11), 1436–1456.

Earthy, J. (1998). Usability maturity model: Human centredness scale. *INUSE Project Deliverable D*, *5*, 1–34.

Edwards, W. K., Newman, M. W., & Poole, E. S. (2010). The infrastructure problem in HCI. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 423–432.

Ellingsen, G., & Hertzum, M. (2019). User participation in the implementation of large-scale suite systems in healthcare. *Proceedings of the 7th International Conference on Infrastructures in Healthcare*, *4*(3). https://doi.org/10.18420/ihc2019_002

Farhoomand, A. (2007). Opening up of the software industry: The case of SAP. *Eighth World Congress on the Management of EBusiness (WCMeB 2007)*, 8–8.

Foerderer, J., Kude, T., Schuetz, S. W., & Heinzl, A. (2019). Knowledge boundaries in enterprise software platform development: Antecedents and consequences for platform governance. *Information Systems Journal*, *29*(1), 119–144.

Gay, G., & Hembrooke, H. (2004). *Activity-centered design: An ecological approach to designing smart tools and usable systems*. Mit Press.

Gizaw, A. A., Bygstad, B., & Nielsen, P. (2017). Open generification. *Information Systems Journal*, *27*(6), 619–642.

gov.uk. (2019). *Government Design Principles*. GOV.UK. https://www.gov.uk/guidance/government-design-principles

Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J., & Cajander, Å. (2003). Key principles for user-centred systems design. *Behaviour and Information Technology*, *22*(6), 397–409.

Hustad, E., Haddara, M., & Kalvenes, B. (2016). ERP and organizational misfits: An ERP customization journey. *Procedia Computer Science*, *100*, 429–439.

Kaipio, J., Lääveri, T., Hyppönen, H., Vainiomäki, S., Reponen, J., Kushniruk, A., Borycki, E., & Vänskä, J. (2017). Usability problems do not heal by themselves: National survey on physicians' experiences with EHRs in Finland. *International Journal of Medical Informatics*, *97*, 266–281.

Kallinikos, J. (2004). Deconstructing information packages: Organizational and behavioural implications of ERP systems. *Information Technology & People*, *17*(1), 8–30.

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *Mis Quarterly*, 67–93.

Koch, C. (2007). ERP-a moving target. *International Journal of Business Information Systems*, *2*(4), 426.

Li, M. (2019). An Approach to Addressing the Usability and Local Relevance of Generic Enterprise Software. *Selected Papers of the IRIS*, *10*, 1–15. https://aisel.aisnet.org/iris2019/3

Li, M., & Nielsen, P. (2019). *Making Usable Generic Software. A Matter of Global or Local Design?* 10th Scandinavian Conference on Information Systems (SCIS), Nokia, Finland.

Magnusson, J., Klingberg, J., Enquist, H., Oskarsson, B., Nilsson, A., & Gidlund, A. (2010). All Aboard: ERP Implementation as Participatory Design. *AMCIS*, 253.

Martin, Dave, Procter, R., Mariani, J., & Rouncefield, M. (2007). Working the contract. *Proceedings of the 19th Australasian Conference on Computer-Human Interaction: Entertaining User Interfaces*, 241–248.

Martin, David, Mariani, J., & Rouncefield, M. (2007). Managing integration work in an NHS electronic patient record (EPR) project. *Health Informatics Journal*, *13*(1), 47–56.

Mathiassen, L., & Nielsen, P. A. (2008). Engaged scholarship in IS research. *Scandinavian Journal of Information Systems*, *20*(2), 1.

Mousavidin, E., & Silva, L. (2017). Theorizing the configuration of modifiable off-the-shelf software. *Information Technology & People*, *30*(4), 887–909.

Msiska, B., & Nielsen, P. (2017). Innovation in the fringes of software ecosystems: The role of socio-technical generativity. *Information Technology for Development*, 1–24.

Mumford, E. (2006). The story of socio-technical design: Reflections on its successes, failures and potential. *Information Systems Journal*, *16*(4), 317–342.

Myers, M. D. (2019). *Qualitative research in business and management*. Sage.

Nielsen, J. (1994). *Usability engineering*. Elsevier.

Norman, D. (2013). *The design of everyday things: Revised and expanded edition*. Constellation.

Norman, D. A. (2005). Human-centered design considered harmful. *Interactions*, *12*(4), 14–19.

Pollock, N., Williams, R., & D'Adderio, L. (2007). Global software and its provenance: Generification work in the production of organizational software packages. *Social Studies of Science*, *37*(2), 254–280.

Prause, J. (2020, November 18). *Digitization vs Digitalization*. SAP Insights. https://insights.sap.com/digitization-vs-digitalization/

Pries-Heje, L., & Dittrich, Y. (2009). ERP implementation as design: Looking at participatory design for means to facilitate knowledge integration. *Scandinavian Journal of Information Systems*, *21*(2), 4.

Rickmann, T., Wenzel, S., & Fischbach, K. (2014). *Software ecosystem orchestration: The perspective of complementors*.

Roland, L. K., Sanner, T. A., Sæbø, J. I., & Monteiro, E. (2017). P for Platform: Architectures of large-scale participatory design. *Scandinavian Journal of Information Systems*, *29*(2).

Rosson, M. B., & Carroll, J. M. (2002). *Usability engineering: Scenario-based development of human-computer interaction*. Morgan Kaufmann.

SAP Fiori. (2020). SAP. https://www.sap.com/products/fiori.html

Schatzki, T. R. (2019). *Social Change in a Material World*. Routledge.

Sestoft, P., & Vaucouleur, S. (2008). Technologies for evolvable software products: The conflict between customizations and evolution. In *Advances in Software Engineering* (pp. 216–253). Springer.

Sia, S. K., & Soh, C. (2007). An assessment of package–organisation misalignment: Institutional and ontological structures. *European Journal of Information Systems*, *16*(5), 568–583.

Simonsen, J., & Robertson, T. (2012). *Routledge international handbook of participatory design*. Routledge.

Soh, C., & Sia, S. K. (2008). The challenges of implementing" vanilla" versions of enterprise systems. *MIS Quarterly Executive*, *4*(3), 6.

Sommerville, I. (2008). Construction by Configuration: Challenges for Software Engineering Research and Practice. *19th Australian Conference on Software Engineering (Aswec 2008)*, 3–12. https://doi.org/10.1109/ASWEC.2008.4483184

Strong, D. M., & Volkoff, O. (2010). Understanding Organization—Enterprise system fit: A path to theorizing the information technology artifact. *MIS Quarterly*, 731–756.

Svanæs, D., & Gulliksen, J. (2008). Understanding the context of design: Towards tactical user centered design. *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges*, 353–362.

Sykes, T. A., & Venkatesh, V. (2017). Explaining post-implementation employee system use and job performance: Impacts of the content and source of social network ties. *MIS Quarterly*, *41*(3), 917–936.

Tavory, I., & Timmermans, S. (2019). Abductive analysis and grounded theory. *The SAGE Handbook of Current Developments in Grounded Theory*, 532–546.

Van de Ven, A. H. (2007). *Engaged Scholarship: A Guide for Organizational and Social Research*. OUP Oxford.

Vilpola, I. H. (2008). A method for improving ERP implementation success by the principles and process of user-centred design. *Enterprise Information Systems*, *2*(1), 47–76.

Wareham, J., Fox, P. B., & Cano Giner, J. L. (2014). Technology ecosystem governance. *Organization Science*, *25*(4), 1195–1215.

Zahlsen, Ø. K., Svanæs, D., Faxvaag, A., & Dahl, Y. (2020). Understanding the Impact of Boundary Conditions on Participatory Activities. *Proceedings of the 11th Nordic Conference on Human-Computer Interaction: Shaping Experiences, Shaping Society*, 1–11.