12-18-2019

An Approach to Addressing the Usability and Local Relevance of Generic Enterprise Software

Magnus Li

AN APPROACH TO ADDRESSING THE USABILITY AND LOCAL RELEVANCE OF GENERIC ENTERPRISE SOFTWARE

Research paper

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

Abstract

Designing for usability and locally relevant features for end-users in generic 'product' or 'packaged' enterprise software projects is challenging. On the generic level, designers must aim at supporting variety, which is contradictory to the specificity needed to make user interfaces usable, and functionality relevant to end-users within specific organizational contexts. Also, addressing these concerns during the implementation of generic software is difficult due to limitations in the design flexibility of the software, time and resource constraints, possible maintenance issues following customization, and a lack of design methods appropriate for the context of software implementation. Reporting from an ongoing Action Design Research project following a global generic health software, this paper conceptualizes a Generic Software Design Lab that aims to equip design on the level of software implementation with flexibility, tools, and methods to efficiently localize generic software. By conceptualizing the approach and discussing how it works to strengthen implementation-level designers' ability to address usability and local relevance, the paper contributes with learnings to research and practice related to design, development, and implementation of generic software.

Keywords: Generic Software, Packaged Software, Product Software, Implementation, Design, Usability, Design Infrastructure, Action Design Research, Misfits

1 Introduction

The challenge of making generic *packaged* or *product* enterprise software that is perceived as usable and relevant by end-users within the established practice of specific organizations has been well documented (David Martin, Mariani, & Rouncefield, 2007; Sia & Soh, 2007; Strong & Volkoff, 2010). A wealth of literature has described 'misfits' or 'misalignment' between enterprise software and organizational practice as a key challenge within IS (Sia & Soh, 2007; Strong & Volkoff, 2010). Also, an extensive body of usability-oriented literature has established that such software is associated with poor usability for the end-users, for example with Enterprise Resource Planning Software (ERP) (Topi, Lucas, & Babaian, 2005; Wong, Veneziano, & Mahmud, 2016), and Health Information Software (HIS) (Kaipio et al., 2017). This seems to be an inherent and inevitable aspect of generic software, due to the way in which they are designed, focusing on markets of several organizations rather than the specific practices to be supported within each (Bertram, Schaarschmidt, & von Kortzfleisch, 2012; Mousavidin & Silva, 2017; Sia & Soh, 2007). The required emphasis on the variety of use-cases challenges the traditional ways of usability and end-user oriented design (Harms & Grabowski, 2011). While software that is usable and relevant within end-users established practices is typically argued to be attained through designs that are specific to the context of use (Norman, 2013; Rosson & Carroll, 2009), generic software that aims to serve a multitude of such contexts must emphasize the opposite, and avoid specificity that may potentially make the software irrelevant in other organizations (Martin, Rouncefield, O'Neill, Hartswood, & Randall, 2005; Pollock & Cornford, 2002). The necessary lack of

sensitivity to organization and practice-specific requirements in generic software design thus contrasts much of the prescriptive knowledge developed around how usable and relevant technologies are designed, which emphasize bottom-up, 'situated' design processes where fit with established practices are at the center of concerns (Kujala, 2003; Norman, 2013; Rosson & Carroll, 2009; Suchman, 2002). Instead, the nature of generic software may enforce a rather top-down design process during implementation, where significant parts of user interfaces and functionality are pre-defined by the vendor, often geographically and culturally distant from the context of use (Davenport, 1998; Roland, Sanner, Sæbø, & Monteiro, 2017).

Together with collogues, I have earlier argued that one approach to addressing the issue is for generic software designers to increasingly emphasize the development of socio-technical resources that help implementation-level designers to efficiently localize the software according to local needs during implementation within specific organizations (Li & Nielsen, 2019b). I refer to this collection of resources as a design infrastructure for implementation-level design, including configurable and extendable software components, documentation, implementation-guidelines, and design-methods that are aligned with the adaption capabilities of the software (Li & Nielsen, 2019a). Similar arguments have been put forth by, for instance, Sommerville (2008), arguing that much of software development and design today is based on configurable generic packages. Thus, the configurability of generic software which provides the basis for processes of 'construction by configuration' during implementation is a core issue, but has received limited attention in IS and Software Engineering research. In the future, Sommerville argues, researchers need to *"explore the notion of configurability and to establish design principles and guidelines for developers of configurable systems"* (Sommerville, 2008, p. 9). Along the same line of arguments, a body of research calls for increased focus on the process of localization during implementation or *implementation-level design*. That is, the process where the generic attributes of user interfaces and functionality are shaped to mesh better with local practice (e.g., Dittrich, 2014; Dittrich, Vaucouleur, & Giff, 2009; David Martin et al., 2007). To facilitate this second level of design, the software needs to provide flexibility through various configuration and customization features, yet avoid compromising the ability for future software upgrades (Dave Martin, Mariani, & Rouncefield, 2004; Sestoft & Vaucouleur, 2008). As such, addressing usability is a joint effort between two levels of design (Li & Nielsen, 2019b); generic and implementation. There is however limited research on generic software implementation as a process of design (Pries-Heje & Dittrich, 2009), and especially how generic software vendors may support such localization processes with an emphasis on usability and local relevance (Baxter & Sommerville, 2011; Dittrich, 2014; Li & Nielsen, 2019a; Sommerville, 2008).

Based on experiences from an ongoing Action Design Research project (Sein, Henfridsson, Purao, Rossi, & Lindgren, 2011), this paper contributes to this landscape by conceptualizing an approach to such a multi-level design effort as a *generic software design lab*. A distinctive feature of the approach is its concern for both generic and implementation-level design as a joint effort to address usability. The lab has been established with the aim of strengthening the usability and potential local relevance of the generic health information software District Health Information Software 2 (DHIS2), which is currently implemented in more than 60 countries within several health-related domains. The lab attempts to strengthen the design infrastructure supporting implementation-level designers in their process of building software that works according to the requirements of a specific organization. Concretely, the research question explored in this paper is; how may usability and local relevance of generic packaged enterprise software be addressed through design? I will attempt to provide a possible answer to this rather extensive question by reporting from our ongoing efforts around the DHIS2 software. Specifically, the process we follow, key characteristics, and how it relates to the generic and implementation level of design, and the design infrastructure between them will be presented. The contribution of the paper directly relates to calls for research on the need for an increased understanding of how to support design and development during generic software implementation (Dittrich, 2014; Pollock & Williams, 2008; Sommerville, 2008), and more generally, to research related to misfits, usability, design and implementation of generic enterprise software (Bertram et al., 2012; Pollock & Williams, 2009; Sia & Soh, 2007; Strong & Volkoff, 2010).

The rest of the paper is organized as follows: first, I provide a brief overview of literature related to usability and generic software design. Then, the lab as a research and practical intervention is conceptualized before some key empirical findings and experiences are outlined. Finally, the lab as an approach and how it may contribute in addressing the usability and local relevance of generic software is discussed.

2 Related Research

The topic of this paper is positioned somewhat between the concerns of two overall streams of research-related information systems and design. First, literature related to generic software design and implementation, primarily within the field of IS are typically interested in large-scale system development, standardization, customization, and the issue of software-organization misfits (e.g., Rothenberger & Srite, 2009; Sestoft & Vaucouleur, 2008; Strong & Volkoff, 2010). Design in this perspective often emphasizes how organizational and technological change can be balanced to achieve the best efficiency and economic gains of generic software adoption. Often, 'best practice' design on the level of generic software development, and minimizing software customization while ensuring end-user acceptance for organizational change during implementation is a rule of thumb (Davenport, 1998). Second, an extensive body of use-oriented research within both Human-Computer Interaction (HCI) and design-oriented IS literature emphasizes how the design of technology should be based on the end-users needs and practices (Ehn, 2008; Kujala, 2003; Norman, 2013; Rosson & Carroll, 2009). Within this stream of research, what makes usable and relevant for end-users, and how to design for these desirable traits has been discussed extensively. However, the context of generic software design and implementation, or large-scale software products and infrastructures are rarely considered (Edwards, Newman, & Poole, 2010; Monteiro, Pollock, Hanseth, & Williams, 2013).

Mainly discussed within the latter strand of research, the concept of usability refers to how well an artifact works with a specific set of users within a specific context of use. The concept has a long tradition within the field of HCI, where it often has been quantified through facets such as *learnability, memorability,* and *user satisfaction* (Tractinsky, 2018). In this paper, I employ usability as a descriptive concept referring to the alignment between software' user interfaces and the users' mental models and established practices (Norman, 2013; Rosson & Carroll, 2009; Strong & Volkoff, 2010). Similarly, the term *local relevance* is used to refer to features or functionality that are perceived as useful and valuable to the activities and work of specific sets of end-users. A major driver for the implementation of generic software, such as ERP systems, is to change and standardize practice to make organizations more efficient (Strong & Volkoff, 2010). Although change is of interest, from a use-oriented perspective, these changes need to be sensitive to the practices, tools, and activities that already exists within the organization, to avoid systems perceived as irrelevant, uncoordinated workarounds, loss of desired efficiency gains and reduced worker satisfaction (Mousavidin & Silva, 2017; Soh, Kien, & Tay-Yap, 2000). To achieve such a fit, designers need knowledge on the end-users' established activities and use of technology (Rosson & Carroll, 2009). However, methods developed to support use-oriented design has been criticized for only considering small-scale software development and bespoke software design (Edwards et al., 2010; Monteiro et al., 2013). In contrast to bespoke or 'from-scratch' software development specific for a single organization and use-case, design of generic software products is about considering a market of several organizations, users, and contexts. Design can be described as a process of *generification*, where the shared traits between various implementations are considered, and particularities are filtered out (Pollock & Williams, 2009). Designers can thus often not be sensitive to specific user practices. Often, usability and local relevance suffer (e.g., Atashi, Khajouei, Azizi, & Dadashi, 2016; Kaipio et al., 2017; Topi et al., 2005).

As the sensitivity to the specifics within each organization promoted in use-oriented approaches cannot be addressed directly on the generic level of design, some, including my own prior work argue that the level of generic software implementation should be regarded as a design process in its own right, where generic components, functionality and user interfaces are shaped towards the local particulari-

ties of a single organization (Dittrich, 2014; Li & Nielsen, 2019b; Pries-Heje & Dittrich, 2009). This process is however described as difficult due to limited technical flexibility constraining desired changes in user interfaces (Li, 2019b; David Martin et al., 2007), due to the maintenance implications associated with software customization (Dittrich & Vaucouleur, 2008; Light, 2001; Pollock & Cornford, 2002), due to a lack of design methods appropriate for implementation-level design (Dittrich, 2014; Li & Nielsen, 2019b), and managers perception of software implementation as a rapid process requiring limited emphasis on design-activities (Krabbel & Wetzel, 1998). To facilitate design on the level of implementation, the generic-level designers must provide localization features in the software, and possibly, resources such as design methods in which implementation-level designers can leverage upon during localization. Addressing usability and local relevance of generic software will thus require a joint effort between generic and implementation-level designers (Li & Nielsen, 2019b). Generic-level design has to consider both design of user interfaces and features to the end-users through *generification*, and design of a set of socio-technical components or a *design-infrastructure* that supports the process of implementation-level design. The infrastructure could include localization features in the adaptable generic software such as configuration, customization, and extension of user interfaces and functionality (Li & Nielsen, 2019a; Light, 2001; Sestoft & Vaucouleur, 2008). Further, other artifacts that support design processes, and soft resources such as documentation and appropriate 'best practice' design methods that are aligned with the localization features embedded in the software may be relevant (Baxter & Sommerville, 2011; Li & Nielsen, 2019b). During implementation, these resources will be leveraged to shape the software sufficiently to achieve a usable software for the end-users.

2.1 Design labs

Existing research has established that functional and non-functional misfits are a prominent challenge in generic software, and that design to address it is difficult, both on the level of generic development and implementation. Design-approaches that are suited for strengthening implementation-level designers' capability of addressing usability and relevance during the implementation of generic software is, however, lacking (Dittrich, 2014; Sommerville, 2008). In regards to traditional means of designing for usability and local relevance, the well-established method of *usability laboratories* emphasize careful user testing of user interfaces in controlled environments, often after a working software has been designed (Nielsen, 1994). The approach is not well suited for the multi-level, and multi-organizational environment of generic software. Traditional usability laboratories have also been criticized as considering technologies fit with existing practice and context of use too late in the design process. For instance, Bødker & Buur (2002, p. 155) argue that with technologies that are highly intertwined with established routines, tools, and contextual factors, *"inviting users to evaluate the design of user interfaces of a single instrument in a lab setting does not make much sense*." Based on this, and inspired by the tradition of Participatory Design, the authors conceptualize what they refer to as a *design collaboratorium*, as both a place and process for collaborative design between different stakeholders. Three aspects characterize the approach: 1) a *meeting ground*, which is both a process and place where design unfold, 2) *participants* that take part in the activities, and 3) *design artifacts* that mediate existing practices and new alternatives between the participants. Based on a similar rationale, several researchers have attempted to conceptualize different types of 'design labs' that aim to address usability in a more process-holistic fashion, through continuous interaction between developers, designers, end-users and other project stakeholders throughout the software development process, including 'living labs', highly influenced by rationales and methods of Participatory Design (Kanstrup, 2017). For instance, Binder & Brandt's (2008, p. 121) Design:Lab, described as *"a platform for a collaborative inquiry that is based on design experiments.*" Albeit, these approaches are based on the assumptions of traditional bespoke software development and are not considering the multi-leveled nature of generic software design, where one level (i.e. generic-level design) needs to facilitate further context-specific design during implementation. 'Meta-design' or 'design for further design' (Ehn, 2008) has been discussed in the literature related to end-user-driven IS tailoring and end-user development, and conceptualize how designers build environment or 'spaces' that allow further shaping of technology on dif-

ferent levels and points in time (Fischer, 2008). Research in this area is exclusively interested in design driven by end-users during 'use-time', which differs from the context of generic software implementation and its particular concerns related to time and resource constraints, upgrade issues and maintenance (Sestoft & Vaucouleur, 2008; Soh et al., 2000). Knowledge developed within this stream, and particularly the concept of 'meta-design' could anyhow be relevant in understanding the multi-level type of design that unfolds in generic software projects (Bertram et al., 2012; Dittrich, 2014; Mousavidin & Silva, 2017; Sommerville, 2008).

3 Case: HISP and the DHIS2 software

The empirical case of this paper follows a generic health information software, called District Health Information Software 2 (DHIS2). The software is developed through the ongoing Action Research project Health Information Systems Programme (HISP), spanning over two decades of research on different aspects of health information systems development and implementation in developing countries (Braa, Monteiro, & Sahay, 2004). The University of Oslo is a major actor in this network, and several Ph.D. and master students are working on practical and theoretical problems in collaboration with local practitioners in different implementation efforts. The DHIS2 software was originally developed in close collaboration with end-users and other stakeholders from the public health system in South Africa in the nineties, to support routine health data reporting and use at the district level. With relative success in providing an integrated and usable system there, it has gradually been adopted in other counties and is today used on a national scale in 67 countries worldwide. Today, the software thus has to support significant variety in organizational and cultural contexts. Its flexibility for localization during implementation has also supported its implementation in an increasing number of new health sub-domains such as logistics management, decease surveillance, and patient follow-up. This means that the generic-level designers, referred to as 'the DHIS2 core developers', situated in Oslo, Norway has to work through a process of generification when designing functionality and user interfaces (Gizaw, Bygstad, & Nielsen, 2017). Around the DHIS2 software, the HISP network emphasizes local competence building through the 'DHIS2 Academies', which are regional learning events, organized by different implementation groups established in several countries. These groups include HISP South Africa, India, Bangladesh, Vietnam, Indonesia, Nigeria, Tanzania, and many more. The topics of the academies vary from basic use of the DHIS2 software to advanced customization and extension development. Together, the technical features for localization of the software, (i.e., configuration, customization, and extension (through the development of third-party apps)), and the social competence-building, communication, sharing, and learning resources form the *design-infrastructure* of and around the software. This provides a basis for the implementation-level design taken on by the local HISP groups and other implementing entities to shape the software according to specific local requirements.

The significant variation in use-cases and contexts for the DHIS2 has made usability an increasing challenge across implementations. I have been involved in the HISP network with research and practice for about four years, including activities at the generic level of design, and implementation activities in Uganda and India. My interest in usability triggered the establishment of a PhD-project focusing on how the generic DHIS2 software serving such variety in organizational contexts could be made usable and relevant to end-users with very different conceptual understandings and established practices. The planned research activities were to be involved in implementation processes to understand how the software is shaped towards local requirements and needs, and challenges related to this. After following an initiative attempting to address usability in an implementation in India it became evident that the ability to deal with the usability problems and specific functional needs encountered was highly affected by the adaptability of the software and other resources in the design infrastructure of and around the DHIS2 software. Thus, a relation to the generic level of design, and the shaping of the design-infrastructure supporting implementation-level activities would be relevant to address usability, as the design on the two levels is highly interdependent. Further, several implementation projects, especially in India, had expressed a need for more emphasis on usability in their ongoing implementation projects. This presented an opportunity to establish what is now called "the DHIS2 design lab",

where several researchers are involved in activities on both the level of implementation and generic development. This represents the *generic software design lab* conceptualized in this paper. In essence, the lab consists of a group of people sharing a common overall goal of strengthened usability, communicating using digital tools, and meeting on a frequent basis for discussions around experiences, challenges, ideas, and (possible) interventions.

4 The Lab as Method

The lab as a research approach is based on Action Design Research (ADR) (Sein et al., 2011). Exploring how usability can be addressed in the multi-level environment requires the ability to understand challenges by engaging in implementation-projects while participating in interventions both locally and in the design-infrastructure of DHIS2. As the DHIS2 software is a central component of the research, interventions will often revolve around this technical artifact. However, usability design is also an organizational process, where human and organizational aims, capacities, design methods, and institutional arrangements are at play. Thus, organizational interventions would necessarily be part of the process. Action Design Research is a methodology that supports these two kinds of interventions, in the words of (Sein et al., 2011, p. 40) by *"(1) addressing a problem situation encountered in a specific organizational setting by intervening and evaluating, and (2) constructing and evaluating an IT artifact that addresses the class of problems typified by the encountered situation."*

ADR is a cyclical process where problems are defined based on a diagnosis of an organizational setting. Although being grounded in a practical problem, it must also be related to a more general class of problems, making it relatable to research. Based on the problem of focus, cycles of '*building, intervention, and evaluation'* follows where '*theory-ingrained'* artifacts and interventions are made and evaluated. Parallel to these activities, there is an ongoing process of *reflection and learning*, conceptually moving *"from building a solution for a particular instance to applying that learning to a broader class of problems"* (Sein et al., 2011, p. 44). This involves a constant negotiation of the practical and theoretical framing of the problem. The ADR process provides the foundation of the process of the design lab.

4.1 Structure of the lab

The lab has been and is an evolving phenomenon where participants, arenas for communication, and activities and methods are continuously changing based on our experiences. This and the following sections will attempt to capture the current state of affairs, starting with what makes up the 'lab', and the process we are following.

4.1.1 The Participants

There are two types of participants in the lab. First, the 'formal' participants include me as a PhD-researcher and nine master students with a background in computer science and interaction design from the University of Oslo. My role as a PhD-researcher is twofold; 1) to be actively engaged in activities at the generic and implementation level, and in interventions attempting to strengthen usability, and 2) to coordinate the lab-activities and the master students work, each with their own project related to the overall goal of the lab. Second, the lab includes more 'informal' participants, depending on the concrete projects we are involved in, such as generic and implementation-level designers and developers, and participants from implementing organizations such as project managers, support personnel, and end-users.

4.1.2 Relations to the generic and implementation-level design

The formal participants make the lab an entity somewhat independent from both the DHIS2 'core developers' as generic level designers, and the groups of implementation-level designers such as HISP India. However, the formal participants are highly involved within implementation projects, collabo-

rating closely with the implementation-level designers. For the generic level, we attempt to find a balance between involvement and autonomy, where our learnings and interventions can inform and be informed by the activities of the 'core' developers and designers, while remaining flexible to make our own decisions, experimental interventions, and extend the design-infrastructure without too many restrictions from other generic-level concerns.

4.1.3 The meeting grounds

To adopt Bødker & Buur's (2002) notion of *meeting ground* as the common place of interaction within the lab, the meeting grounds of our design-lab are regular meetings, workshops, focus groups at the University of Oslo and during collective field-trips to implementation projects (e.g., in India), and digital communication tools (Slack, Jira). On a regular basis, the formal participants meet to discuss empirical experiences and interventions on a practical and theoretical level. Typically, such meetings are topic-centered, revolving around the different sub-problems each participant is working on. Further, interactions with the informal members happen through a variety of channels, from attending meetings, dedicated topic discussions, Skype, participating in design and development activities, group or one-on-one interviews, and all sorts of other interactions unfolding during the design cycles described in the following section.

4.2 The process

The process we are following in the design lab can be described as cycles of four elements:

1. ***Accumulation of empirical experiences***: an understanding of the phenomenon of focus that gradually develops through cycles of investigation, participation, and interventions within the organizational environment, the DHIS2 software, and the design-infrastructure.
2. ***Problem definition***: an overall problem and relevant sub-problems are derived and continuously renegotiated based on the empirical experiences. Problems can be small and concrete, or high-level, in which several 'sub-cycles' are ongoing to explore different dimensions.
3. ***Investigation, ideation, and intervention***: the process of building empirical knowledge is driven by investigation, ideation, and intervention. Problems and phenomenon are investigated in-depth, or potential interventions to address the defined problems are discussed and explored. In our approach, we use the term intervention in a broad sense, not limited to concrete material changes or large and well-planned organizational changes, but involving all types of interactions where the researcher's actions have noteworthy effects on how things play out in process and result related to the problem of focus. Investigation, ideation, and intervention are thus in many ways' inseparable, as highly participative investigation and ideation within a project may ultimately affect the course of action and is, therefore, an intervention in itself. In the words of (Walsham Walsham, 1995, p. 77), *"even if researchers view themselves as outside observers, they are in some sense conducting action research by influencing what is happening in the domain of action". Intervention* in our process does as such represent a point of contact with the client infrastructure, be it organizational or technical, which to our knowledge have noteworthy effects related to the problem of interest.
4. ***Evaluation:*** The more or less extensive interventions are continuously evaluated, further building the empirical experiences that form the basis for new cycles of re-formulated problem areas, sub-problems, and new ideas and interventions.

All stages of the process are informed by theory and related research or a "framework of ideas" (Baskerville & Wood-Harper, 1996, p. 239), both as a sensitizing device when making sense of and analyzing empirical experiences, and more explicitly in the process of framing problems, and generating ideas for and conducting interventions and evaluation. Further, the growing base of empirical experiences is continuously analyzed systematically to produce knowledge to feed back to research. The lab is as such constantly framing knowledge in terms of an instance domain (concrete experiences during implementation) and an abstract domain (class of problems relatable to the world of research)

(Gregor, Müller, & Seidel, 2013; Lee, Pries-Heje, & Baskerville, 2011). It can also be seen as a hermeneutic approach, where the understanding of the overall problem and related sub-phenomenon unfold in a mutual informing process (Klein & Myers, 1999). More systematically, the empirical experiences are often structured through thematic analysis (Braun & Clarke, 2006) by coding field-notes, creating themes and concepts, and reflecting on their relationship and relating them to the abstract domain of theory and existing research. Further, the regular meetings between the formal members of the design lab provide a fruitful environment for reflection (Daudelin, 1996) and to drive the process of abstraction and de-abstraction to conceptualize important phenomenon experienced. Figure 1 illustrates the process.

A central element of the process is the base of empirical experiences related to the problem area. Each cycle of sub-problem definition, ideation, intervention, and evaluation feeds back to the empirical experiences, which sheds new light on the problem area, potentially solving one sub-problem and revealing new ones. Each sub-project and participant within the lab have their own base of empirical experiences related to their overall problem. Further, the design lab as a collaborative entity shares a combined set of empirical experiences, built and shaped through the various *meeting grounds*, and formalized in writings and interventions in the design-infrastructure. Experiences often result in new problems to be explored and addressed through organizational or artifact-oriented interventions.

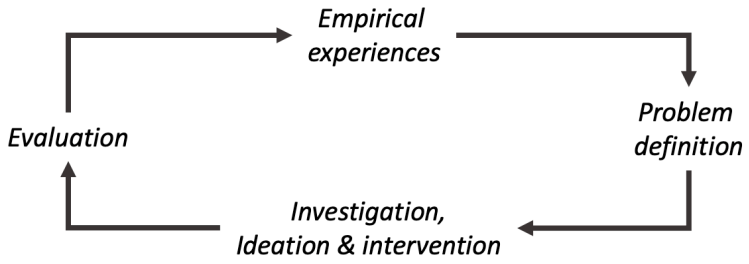

Figure 1. The process of accumulating empirical experiences in the design lab

5 Empirical Experiences and Findings

In this section, I will briefly introduce some of the activities the DHIS2 design lab has been involved in the last year, and the learnings it has yielded. This represents some of the empirical experiences built through the approach of the lab. The experiences are related to the two levels of design; generic and implementation, where direct involvement in implementation-level design inform activities on the generic level which may result in interventions in the design-infrastructure. Vice versa, the learnings through activities on the generic level shapes our involvement in the implementations. Two overall implementation projects are first introduced before some of the design infrastructure-related interventions are described.

5.1 Key projects

The HISP India group has been our main implementation-level 'partner' which we have been working with during this period. The HISP India group consists of about 40 software developers and implementers with thorough competence and experience with implementing the DHIS2 software. Their main office is located in Noida, outside of New Delhi, and they are engaged in a variety of implementation projects, both in India and neighboring countries. Most of these revolve around the implementation and maintenance of the DHIS2 software within public and private health organizations.

5.1.1 The UPHMIS project

One of the large projects of HISP India is the implementation and continuous maintenance of the state-wide routine health data reporting system named 'the UPHMIS portal' in Uttar Pradesh. A state with an estimated population of over 200 million. The portal is based on the DHIS2 software and has been implemented over the last four years. I got involved in the project as usability and user experience have become an explicit goal of the current phase of the project. This has been motivated by a variety of usability problems related to the UI-design of the portal, reported by end-users and managers, making the system hard to learn and use. As expressed by a technical project manager in the state; *"we train the same users over and over on the same aspects".* This resonated well with my research focus, and it was agreed that I would participate in the process of addressing these usability-related problems together with the 'UPHMIS-team' within HISP India. The process started with a diagnostic phase where we together discussed the problems in detail, and how these could be addressed in the DHIS2 software. It quickly became apparent that the usability problems were hard to address on the level of implementation due to the limited ability to shape the generic user interfaces of DHIS2 according to local needs. Through rounds of discussions, we developed a preliminary plan for how to approach eight key usability problems related to user interfaces, including inappropriate terminologies, inconsistent design, and difficult navigation. Further, a more fundamental problem of informing design decisions based on end-users' practices and needs was articulated. A period with contractual negotiations between HISP India and the implementing organization followed. Meanwhile, I engaged a set of four master students at the University of Oslo, where each was assigned one or two problems of focus. Two visits to India followed, where the master students spent one month together with HISP India to explore their respective problems together. The process unfolded as several rounds of discussions with implementers, client managers, and end-users, and ideas for technical solutions were explored. The process is ongoing, and the students are collaborating with HISP India while being in Norway. A longer trip to India is planned in the fall. In these processes, several sub-problems are continuously defined which we return to after a brief introduction of the second implementation project.

5.1.2 The AMR project

During my engagement with HISP India on the UPHMIS project, a new project was taken on by the HISP India team, where the DHIS2 software will be implemented as an antimicrobial resistance (AMR) reporting system. Initially for a set of large hospitals in Delhi, but the plan is to scale to support the whole country, and also support data reporting and use in domains other than health such as agriculture and veterinary. From the start, a prominent goal has been to develop what many project-actors refer to as a 'user-friendly' system. The case is highly relevant to the lab's focus on usability as the generic DHIS2 here will meet a highly generic implementation case, including several organizations and domains. Further, the initial hospitals already have a reporting system that has been designed in close collaboration with end-users within the hospitals, and which is regarded as highly usable. Due to the software's limited ability to scale, DHIS2 is seen as a better candidate when moving forward, but its generic nature could potentially make the new system less usable than the existing. Two master students and formal members of the design lab are working closely with HISP India on design and development.

5.2 Generic-level involvement

While being deeply involved in the ongoing implementation projects, the master students and I represent the formal participants of the DHIS2 design lab. Through frequent discussions of empirical experiences, problems, ideas, and evaluations of interventions we try to generalize our findings to 1) discuss how to best approach problems within the implementations, and 2) to inform interventions on the generic level by strengthening the design infrastructure. Three concrete examples here follow:

5.2.1 The design system and the 'drag-and-drop' design-environment

Experiences from our collaboration with HISP India showed that a major obstacle to achieving usability was the limited design-flexibility in the DHIS2 software. In addition to configuration features, the DHIS2 allows for the development of third-party apps with significant design-flexibility. This does, however, require substantial time, resources and competence, and implementers thus tend to avoid this. A significant concern in the lab is now to create resources that will lower the hurdle of creating such third-party apps. One concrete intervention is a "design system," a library of UI components, which quickly can be combined and rearranged to speed up the process of development during implementation. The intervention is informed by existing research on end-user development and component-based tailorability (e.g., Wulf, Pipek, & Won, 2008). By creating sets of standardized components, the approach is also relevant as a solution to the issue of design consistency within the UPHMIS project. Thus, the master student working with the design-system is using learnings from problem-solving within the UPHMIS project in developing the design-system to be part of the design-infrastructure. Further, one master student in the AMR-project utilizes and extends the design-system while designing and developing user interfaces within the implementation. Activities and interventions on the level of implementation are thus linked to interventions on the generic level, experiences from each informing the other.

Based on the same problem of design-flexibility versus ease of development and the experiences and interventions on the design-system, a new master student has recently started exploring how the component library can be used together with modern JavaScript-frameworks to create a drag-and-drop user interface design-environment for data entry applications. The new addition to the design-infrastructure has the potential of lowering the hurdle associated with app development significantly. The example illustrates how the evolving base of empirical experiences give rise to new sub-problems and ideas for interventions, which are taken on as new design-challenges by the participants in the lab.

5.2.2 Dashboard widgets

In a similar multi-level fashion, a master student engaged in the UPHMIS-project is exploring how so-called 'dashboard widgets' can make the portal easier to navigate and make relevant and useful information more visible to groups of end-users. The 'DHIS2 dashboard' provides a space where end-users or implementation-level designers on their behalf can add shortcuts and maps, graphs, and tables of the information most relevant to their work. The 'widget' functionality affords the design of more custom dashboard components using JavaScript and HTML. The dashboard widgets thus provide a flexible space to meet end-user needs on the level of implementation. The design-space is explored by designing widgets based on different groups of users of the UPHMIS portal by applying different methods and techniques to learn about their practices and needs. The learnings are, in turn, discussed in the design lab to inform documentation and best-practices in the design-infrastructure to make the widget functionality more usable to implementation-level designers in other projects. Problem definitions, ideation, and interventions are informed by several strands of research, such as 'meta-design' (Fischer, Fogli, & Piccinno, 2017), scenario-based design (Rosson & Carroll, 2009), and 'participatory customization' (Kimaro & Titlestad, 2008). The learnings will also be used in our involvement in the AMR project, where dashboard widgets are planned to play a major role in making the system 'user-friendly'.

5.2.3 Design methods and institutional arrangements supporting design-processes

What we learn in terms of process is a major aspect of the lab. This includes the approach we are exploring in the AMR-project, where we attempt to address usability through activity-specific apps, which we coin as *platform appliances* with varying degrees of contextual genericness (Li, 2019a). This involves methods for understanding end-users' activities, to mirror these within the technology. Further, one master student focuses on concrete methods and techniques that are suitable for investigating users' practices and needs in the UPHMIS-project. Her findings feed into our overall experi-

ences, which over time will be used to formulate suggested implementation methods for DHIS2, added to the design-infrastructure as learning resources and advice for best practice.

6 Discussion

The aim of this paper is to conceptualize the activities of the DHIS2 design lab as a generic software design lab, and how the approach relates to the aim of addressing usability and local relevance of generic enterprise software and to outline its characteristics as a research-driven intervention-based approach. While the structure and process of the lab has been outlined in chapter 4, and concrete activities exemplified in chapter 5, this chapter will discuss the lab's role in addressing usability and local relevance, directly related to the research question put forth in the introduction. Also, five key characteristics of the approach are summarized.

The overall goal of the DHIS2 design lab is to strengthen the usability and local relevance of the generic software DHIS2 for end-users, while systematically analyzing our activities and results to contribute to research. It has similarities to other 'design lab' approaches such as Bødker & Buur's (2002) *design collaboratorium*, and the Design:Lab of Binder & Brandt (2008) where the aim is to design for usability throughout design processes rather than limited to summative evaluations. However, the DHIS2 design lab differs by being concerned and involved with design on two levels. While the main targeted beneficiaries of focus are end-users through more usable user interfaces and locally relevant features in the technology of their workplace, a major part of our work concerns 'design for design', or 'meta-design' (Ehn, 2008; Li & Nielsen, 2019b). That is, to provide a usable socio-technical design infrastructure to the implementers of the software, with the rationale that this will increase their ability to localize the software towards the needs within the use-case at hand. The lab thus attempts to facilitate the 'joint effort' we and others have earlier argued to be required to address usability in generic software (Dittrich, 2014; Li & Nielsen, 2019b), and to better support design within a process of 'configuration by construction' (Sommerville, 2008) or 'deferred design' (Dittrich, 2014). In contrast to *generification* as a process of aligning use-requirements from different organizations discussed in prior research on generic software (Gizaw et al., 2017; Pollock & Williams, 2009), the generification process of the design lab is aimed at requirements for implementation-level design. Concretely, the learnings within the lab thus far have concerned three major aspects, highly relevant to the problems outlined and discussed by related literature:

1. Exploring and strengthening the material design-space or localization features of generic software. That is, the configuration, customization, and extension-features utilized during implementation (Li, 2019b; David Martin et al., 2007; Sestoft & Vaucouleur, 2008; Sommerville, 2008), while limiting their negative impact on software maintenance and making them sensitive to time and resource constraints (Light, 2001; Pollock & Cornford, 2002).
2. Exploring and defining appropriate implementation-level design methods, where main aspects include techniques that are sensitive to the social, cultural and economic nature of implementation projects, and the alignment between methods and the design-space of the generic software (Baxter & Sommerville, 2011; Dittrich et al., 2009).
3. Establishing facilitating institutional arrangements where these methods can take place within implementation projects, (Pries-Heje & Dittrich, 2009), and also how to facilitate the 'joint effort' spanning implementation and generic-level design to strengthen the design infrastructure of the software (experiences from the organizing of the design lab itself) (Dittrich, 2014; Pollock & Williams, 2008)

In the process of understanding and addressing these issues, knowledge is accumulating within each lab-participant's base of empirical experiences, the labs collective base, the informal participants' knowledge, and in the design-infrastructure of the software. Formalized knowledge, related to a class of problems within the abstract domain is, in turn, contributing to research. The research and design processes for the overall lab, participants own projects, and sub-problems within the different levels can be described as cycles of the process presented in section 3.4. The lab is thus part of three dynamic

environments and seeks to learn from and contribute to all. First, it works as an engine to learn from attempts to address functional and non-functional misfits (Strong & Volkoff, 2010), usability design obstacles (Li, 2019b) and experimental interventions within the implementation-level design processes. Second, learnings inform the engagement in the generic-level design, much through interventions in the design-infrastructure. By utilizing these resources, and through the knowledge and ideas of the lab's participants, generic-level learnings feed into the activities on the level of implementation. Third, theory and research inform the whole process on both levels from methods, problem definitions, interventions, and evaluations, and the empirical experiences from both levels of intervention are analyzed, conceptualized, and reported as research. Figure 2 summarizes these environments and their relation to the lab.

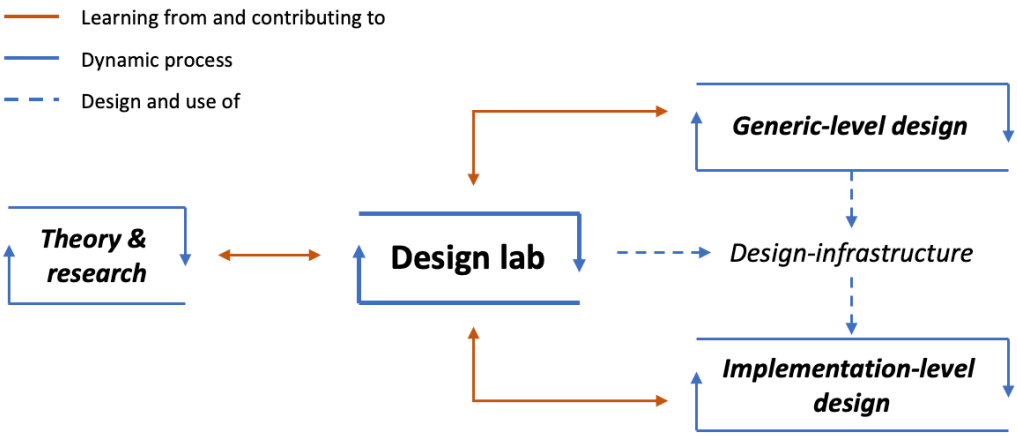

Figure 2. *The Generic Software Design Lab 'DHIS2 design lab' as engaged in three dynamic environments*

As a summary, the generic software design lab has five core characteristics. First, it is part of three dynamic environments. The lab is involved in and seeks to contribute to both generic and implementation-level design, and is driven by and contributing to related theory and research. Second, the accumulation of empirical experiences is central. Empirical experiences related to a problem is accumulating based on cycles of investigation, ideation, intervention, and evaluation. The lab represents a shared collective of empirical experiences conveyed through various meeting grounds. Third, accumulation happens through problem-driven, recursive, and interrelated cycles of intervention. As empirical experiences evolve, problems are continually reformulated. Each sub-problem gives rise to sub-cycles that feed into the empirical experiences. Cycles may be interrelated across problems, formal participants, and generic and implementation-level design. The lab learns through participation and interventions within generic and implementation-level design. Interventions include all researcher's actions that have noteworthy effects on how things play out in process and result related to the problem(s) of focus. Fourth, the activities revolve around the design infrastructure. Cycles are often either based on utilization of the current design-infrastructure or making interventions in it. Utilization and intervention are mutually informing each other. Finally, while the lab's aim is to drive problem-focused and research-driven investigations and interventions, it is also an intervention in its own right by being established within the organization of study. Contributions to research thus relate both to the interventions unfolding within the lab, and learnings from organizing the lab itself (e.g., this paper).

7 Conclusion

Based on initial experiences from an ongoing Action Design Research project, this paper has conceptualized a *generic software design lab* as an approach to address usability and local relevance for end-users in a generic packaged enterprise software. Designing for usability and local relevant features in generic software is difficult, due to its multi-contextual scope of implementation, and addressing usability during implementation-level design require a usable design-infrastructure to support the localization of the software towards local needs. The design lab attempts to strengthen implementation-level design through interventions in the design infrastructure. The process we follow entails four major elements; *accumulation of empirical experiences*; *problem definition*; *investigation, ideation & intervention*; and *evaluation*. A central element is the accumulated base of empirical experiences, which provides a basis for continuous cycles that further extends the empirical experiences, and feed into interventions in the design-infrastructure of the software. The role of the lab is to work as an engine for understanding opportunities and obstacles within the implementation-level design, to strengthen the design-infrastructure accordingly while contributing to research and theory.

The approach presented is based on the preliminary experiences of establishing the DHIS2 design lab, involvement in two major implementation-level projects, and about four more or less concrete design-infrastructure interventions the last year. Its further evolution will yield more extensive learnings that can help to build the conceptualization of the approach further, possibly including the development of design-principles regarding both the lab as approach, and concrete design infrastructure interventions. However, the preliminary experiences presented in this paper may serve as valuable input for researchers and practitioners concerned with generic software design. For researchers, the paper provides interesting experiences on how to investigate and explore methods and techniques for the design of and around generic software. For practitioners, the approach and experiences are relevant to managers and designers engaged in generic software development either at the generic or implementation level of design.

References

Atashi, A., Khajouei, R., Azizi, A., & Dadashi, A. (2016). User Interface problems of a nationwide inpatient information system: a heuristic evaluation. *Applied Clinical Informatics*, *7*(1), 89.

Baskerville, R. L., & Wood-Harper, A. T. (1996). A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, *11*(3), 235–246.

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

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.

Binder, T., & Brandt, E. (2008). The Design: Lab as platform in participatory design research. *CoDesign*, *4*(2), 115–129.

Bødker, S., & Buur, J. (2002). The design collaboratorium: a place for usability design. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *9*(2), 152–169.

Braa, J., Monteiro, E., & Sahay, S. (2004). Networks of action: sustainable health information systems across developing countries. *Mis Quarterly*, 337–362.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101.

Daudelin, M. W. (1996). Learning from experience through reflection. *Organizational Dynamics*, *24*(3), 36–48.

Davenport, T. H. (1998). Putting the enterprise into the enterprise system. *Harvard Business Review*, *76*(4).

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

Dittrich, Y., & Vaucouleur, S. (2008). *Customization and Upgrade of ERP systems: An empirical Perspective*.

Dittrich, Y., Vaucouleur, S., & Giff, S. (2009). ERP customization as software engineering: knowledge sharing and cooperation. *IEEE Software*, (6), 41–47.

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. ACM.

Ehn, P. (2008). *Participation in design things*. 92–101. Indiana University.

Fischer, G. (2008). Rethinking software design in participation cultures. *Automated Software Engineering*, *15*(3), 365–377.

Fischer, G., Fogli, D., & Piccinno, A. (2017). Revisiting and broadening the meta-design framework for end-user development. In *New perspectives in end-user development* (pp. 61–97). Springer.

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

Gregor, S., Müller, O., & Seidel, S. (2013). Reflection, Abstraction And Theorizing In Design And Development Research. *ECIS*, *13*, 74.

Harms, P., & Grabowski, J. (2011). Usability of generic software in e-research infrastructures. *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, *1*.

Kaipio, J., Lääveri, T., Hyppönen, H., Vainiomäki, S., Reponen, J., Kushniruk, A., … 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.

Kanstrup, A. M. (2017). Living in the lab: an analysis of the work in eight living laboratories set up in care homes for technology innovation. *CoDesign*, *13*(1), 49–64.

Kimaro, H. C., & Titlestad, O. H. (2008). Challenges of user participation in the design of a computer based system: The possibility of participatory customisation in low income countries. *Journal of Health Informatics in Developing Countries*, *2*(1).

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.

Krabbel, A., & Wetzel, I. (1998). *The customization process for organizational package information systems: A challenge for participatory design*. *98*, 45–54.

Kujala, S. (2003). User involvement: a review of the benefits and challenges. *Behaviour & Information Technology*, *22*(1), 1–16.

Lee, J. S., Pries-Heje, J., & Baskerville, R. (2011). Theorizing in design science research. *International Conference on Design Science Research in Information Systems*, 1–16. Springer.

Li, M. (2019a). Making Usable Generic Software - The Platform Appliances Approach. *Workshop on "Platformization in the Public Sector"*. Presented at the Twenty-Seventh European Conference on Information Systems (ECIS2019), Stockholm-Uppsala, Sweden.

Li, M. (2019b). Usability Problems and Obstacles to Addressing them in Health Information Software Implementations. *IFIP Advances in Information and Communication Technology*, *552*. Dar es Salaam, Tanzania: Springer.

Li, M., & Nielsen, P. (2019a). *Design Infrastructures in Global Software Platform Ecosystems*. Presented at the 6th Innovation in Information Infrastructures (III) Workshop, Guildford, UK.

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

Light, B. (2001). The maintenance implications of the customization of ERP software. *Journal of Software Maintenance and Evolution: Research and Practice*, *13*(6), 415–429.

Martin, Dave, Mariani, J., & Rouncefield, M. (2004). Implementing an HIS project: everyday features and practicalities of NHS project work. *Health Informatics Journal*, *10*(4), 303–313.

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.

Martin, David, Rouncefield, M., O'Neill, J., Hartswood, M., & Randall, D. (2005). *Timing in the art of integration:'that's how the bastille got stormed'*. 313–322. ACM.

Monteiro, E., Pollock, N., Hanseth, O., & Williams, R. (2013). From artefacts to infrastructures. *Computer Supported Cooperative Work (CSCW)*, *22*(4–6), 575–607.

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

Nielsen, J. (1994). Usability laboratories. *Behaviour & Information Technology*, *13*(1–2), 3–8.

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

Pollock, N., & Cornford, J. (2002). Fitting standard software to non-standard organisations. *Proceedings of the 2002 ACM Symposium on Applied Computing*, 721–725. ACM.

Pollock, N., & Williams, R. (2008). *Software and organisations: The biography of the enterprise-wide system or how SAP conquered the world*. Routledge.

Pollock, N., & Williams, R. (2009). Global software and its provenance: generification work in the production of organisational software packages. In *Configuring User-Designer Relations* (pp. 193–218). Springer.

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.

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. (2009). Scenario-based design. In *Human-computer interaction* (pp. 161–180). CRC Press.

Rothenberger, M. A., & Srite, M. (2009). An investigation of customization in ERP system implementations. *IEEE Transactions on Engineering Management*, *56*(4), 663–676.

Sein, M., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action design research. *Mis Quarterly*.

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.

Soh, C., Kien, S. S., & Tay-Yap, J. (2000). Cultural fits and misfits: is ERP a universal solution? *Communications of the ACM*, *43*(4), 47–47.

Sommerville, I. (2008). Construction by configuration: Challenges for software engineering research and practice. *19th Australian Conference on Software Engineering (ASWEC 2008)*, 3–12. IEEE.

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

Suchman, L. (2002). Located accountabilities in technology production. *Scandinavian Journal of Information Systems*, *14*(2), 7.

Topi, H., Lucas, W. T., & Babaian, T. (2005). *Identifying Usability Issues with an ERP Implementation*. 128–133.

Tractinsky, N. (2018). The usability construct: A dead end? *Human–Computer Interaction*, *33*(2), 131–177.

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, *4*(2), 74–81.

Wong, W.-P., Veneziano, V., & Mahmud, I. (2016). Usability of Enterprise Resource Planning software systems: an evaluative analysis of the use of SAP in the textile industry in Bangladesh. *Information Development*, *32*(4), 1027–1041.

Wulf, V., Pipek, V., & Won, M. (2008). Component-based tailorability: Enabling highly flexible software applications. *International Journal of Human-Computer Studies*, *66*(1), 1–22.