Generification by Translation: Designing Generic Systems in Context of the Local

1. Introduction

The implementation of large generic systems in organizations is associated with many benefits. Some of these are institutional-wide coverage, streamlining of work practices, and the possibility to reuse systems across many institutional settings. However, many studies have noted how organizations are different and therefore may have diverging needs (e.g., Berg, 1999; Berg & Goorman, 1999; Star & Ruhleder, 1996). Accordingly, it is crucial for an organization's vendors and project managers to align local needs with technical opportunities in order to establish a well-working system. Pollock and Williams (2008) have coined this process "generification", and describe it as the vendors' strategy of making a generic system work in several settings. Together with customization capabilities of the software, generification involves social processes of ordering, prioritizing, and persuading users in order to motivate them to use similar versions of the same system that is installed in different organizations (Pollock & Williams, 2008)

Currently, the mechanisms of generification during implementation and use are well known (see, e.g., Pollock & Williams, 2008), but we have less insight about the generification processes in the design phase and to what extent local knowledge is exploited in the process. Therefore, this paper extends the notion of generification to the formative stages of generic systems and provides insight about the key mechanisms at play in this crucial phase.

A key characteristic of generic systems is that they contain a standardized core supplemented with a customizable part that is a range of clearly defined building blocks (Baldwin & Woodard, 2008; Beale & Heard, 2007, 2008). The idea is that designers can take a step back from the users' context and leave the tailoring of the building blocks to skilled users and domain experts. This sustains a more or less distinct boundary between the designers' technical domain and the users' work domain. An interesting example from healthcare is the emerging openEHR architecture (Beale & Heard, 2007, 2008) developed by the openEHR foundation and standardized by CEN and ISO in the EN/ISO 13606 standard series (Chen, Klein, Sundvall, Karlsson, & Ahlfeldt, 2009, p. 2). Electronic patient records developed in accordance with this architecture are supposed to offer both a high degree of interoperability of data (through so-called archetypes) between different healthcare domains, and a high degree of customization in various use contexts. The openEHR architecture consists of a two-level modeling approach for electronic patient records (EPRs) (Chen et al., 2009, p. 2; Garde, Knaup, Hovenga, & Heard, 2007, p. 333) that separates a system's technical design from clinical concerns. Many pilot projects have so far reported on openEHR pilot projects (Bernstein, Tvede, Petersen, & Bredegaard, 2009; Chen et al., 2009), but very few (if any at all) have studied large-scale implementations (Wollersheim, Sari, & Rahayu, 2009).

We pose the following research question: what characterizes the socio-technical process of designing generic software? We organize our discussion along the following three dimensions: First, we analyze how and to what extent local practice is embedded in the design of generic systems. By this, we explore what domain knowledge the designers need and how they use it for generification purposes. Second, we examine the process whereby users' needs are translated into generic functionality, and, in turn, how this functionality is presented to the users in a way that makes sense to them. Third, we look into how the design of generic software tends to widen the gap between users and designers by requiring considerable planning and up-front design.

Theoretically, our paper draws on the information infrastructure literature (Aanestad & Jensen, 2011; Hanseth & Lyytinen, 2010; Star & Ruhleder, 1996). We emphasize how design of generic systems always faces infrastructural challenges and opportunities in their formative stages. Empirically, we draw on the initial stages of a large EPR project, run by the Northern Norway Regional Health Authority and referred to as BigInvestment, that began in 2012 in the North Norwegian health region and is scheduled to run to 2016. The involved software house, BigVendor, is the largest EPR vendor in the Norwegian healthcare market. BigVendor has started to develop a new EPR infrastructure that will allow skilled users to define and tailor standardized user interfaces and screen workflows (in accordance with clinical workflows) based on an openEHR/archetype methodology (Kalra, 2006).

This paper is organized as follows: in Section 2, we outline our theoretical perspective on generic systems and information infrastructures. In Section 3, we present the Northern Norway Regional Health Authority and the BigInvestment project, and reflect on methodological issues. In Section 4, we present the involved vendor's design strategy. In Section 5, we present the case where we elaborate on several steps of the project's history. In Section 6, we discuss the case followed by the conclusion in Section 7.

2. Theory

2.1. Generic Systems

Large-scale generic systems typically encompass extensive parts of the organizations they operate in, and employees use the systems in various ways. Moreover, large-scale generic systems have an "ability to transcend their place of production" (Pollock, Williams, & D'Adderio, 2007, p. 255). One fundamental idea inherent in large-scale generic systems is that their generic capabilities means that organizations need less resources to implement a system used by other organizations. Some examples of such systems are enterprise resource planning (ERP) systems and, in healthcare, electronic patient record systems (EPRs).

Two concerns have emerged as particularly important when designing generic systems; namely, their ability to support customization and their ability to support interoperability. The degree of interoperability with other systems has so far proved to be limited (Wang, 2007, p. 108), and, if supported, the integration is typically asymmetric (i.e., the integration mechanisms are outlined solely by those who control the generic system) (Sahay, Aanestad, & Monteiro, 2009), which potentially leads to less-robust integrations. A system's customization capability is a fundamental issue when implementing new practices because it deals with some of the rigidity associated with generic systems (Hanseth, Bygstad, Ellingsen, & Johannessen, 2012; Pollock et al., 2007). The ability to customize a system enables domain experts in user organizations to tailor it to local use. Examples of customization capability might be locally generated variables, templates, and definitions of rule-based workflow support. Then, a generic system can be defined as consisting of *the standardized core* in which certain components remain stable and *the complements* that are encouraged to vary across practices or over time (Baldwin & Woodard, 2008, p. 2).

For users, this might be seen as a big leap forward because extensive parts of generic systems can be tailored to their practice. For designers, this also might be beneficial because they would not need to know all the peculiarities and nuances of each specific healthcare practice because tailoring a generic system to an organization would be handed over to domain experts (Chen & Klein, 2007). Thus, an essential issue is to exercise control of the generic software's core. Here, the vendors need to establish some principles of what should remain the standardized core and what should be offered as customizable components to the users (Baldwin & Woodard, 2008).

Still, normally these systems exercise an inevitable impact on routines, practices, and collaboration in organizations. This implies a tremendous amount of generification work, described by Pollock and Williams (2008, p. 129) as "the supplier strategy of taking a technology that has worked in one place and attempting to make it work elsewhere, and, in principle, everywhere". Central in a generification process is the trade-off between particularization and generification. For the developers, typically questions include: what are particular/spesific system requirements from a few customers compared to what are general system requirements from a larger customer group; how particular are those requirements from the few customers; should diversity be built into the system or should functions meeting these particular needs be customized at each site?

While we appreciate Pollock and Williams' (2008) perspective, our study differs in two principal ways. Firstly, their account of generification focuses primarily on the implementation and adaptation of generic systems in organizations. In comparison, our study deals with generification in the design phase of these systems. Second, in their analysis, Pollock and Williams conceptualize the generic system as "the standard" in the form of a standardized package. In our study, we focus on how the new generic system is expected to adhere to an international standardized framework (i.e., openEHR) in order to

ensure interoperability between systems built in accordance with this framework. Accordingly, this scales the complexity.

2.2. The OpenEHR Architecture—A Global Standardized Framework

The openEHR architecture was developed by the openEHR foundations and standardized by CEN and ISO in the EN/ISO 13606 standard series (Chen et al., 2009, p. 2). It was created to support a high degree of interoperability of data between different healthcare domains and a high degree of customization in various use contexts (Beale & Heard, 2007, 2008). It includes a two-level modeling approach for EPRs (Garde et al., 2007, p. 333; Chen et al., 2009, p. 2) that separates the system's technical design from clinical concerns. A standardized reference information model represents the first level, while the openEHR archetypes based on the reference model represent the second level (Garde et al., 2007, p. 333; Beale & Heard, 2007, p. 8). A "blood pressure (BP) archetype, for example, represents a description of all the information a clinician might need or has to report about a blood pressure measurement" in a patient's record (Garde et al., 2007, p. 333). The actual blood pressure value is accompanied by additional data on whom (who measured the BP), how (which type of equipment was used, if the patients was sitting/bed resting), when (related to datum and time of day), and where (refers to "where" on the patients body e.g.intra-arteria BP, right/left arm or leg etc.) as a way of describing the context around the blood pressure measurement. Archetypes are therefore "metadata used to define patterns for the specific characteristics of the clinical data, for example the blood pressure, in this case" (Kalra, 2006, p. 138).

Skilled users are encouraged to embed internationally agreed-on archetypes in systems based on the openEHR architecture to ensure interoperability, but are also free to define their own local archetypes. Archetypes can be seen as generic building blocks (i.e., customizable components) in the hands of clinical personnel or domain experts. In turn, these building blocks can be used to construct templates and compose archetypes into larger structures that often correspond to screen forms, documents, or reports (Beale & Heard, 2007, p. 8). In this way, archetypes are supposed to support a high degree of local customization for users and domain experts:

> *A fundamental aim of the archetype approach … is to empower domain experts to create and change the knowledge inherent in archetypes, thus controlling the way EHRs are built up using designed structures to express the required clinical data and assuring that all necessary constraints on the values of record components are observed.* (Garde et al., 2007, p. 336)

Beale & Heard (2007) also point to that an archetype approach will ensure an easier development process for developers because it separates the technical design and clinical concerns. Hence, a system's developers would not need to know all the organizational peculiarities in every different context because the use of archetypes enable easy reuse of the software across different healthcare organizations. Moreover, the separation of concerns enable system's developers to build stable EHR systems without knowledge about specific clinical content necessary in different fields. The specification of clinical content can be authored and amended later (Chen & Klein, 2007).

The literature has so far reported many successful pilots on the openEHR approach, but very few (if any at all) have applied it on a large scale (Wollersheim et al., 2009). This makes large-scale projects in this domain extremely interesting because there is no definite solution on how these systems should be designed. As such, the information infrastructure concept is a way of gaining more understanding about this process.

2.3. Information Infrastructures

The theoretical framework of information infrastructure has been used to study the design, implementation, and use of large-scale information systems (Aanestad & Jensen, 2011; Hanseth & Lyytinen, 2010; Star & Ruhleder, 1996). These systems are never seen as standalone entities, but are integrated with other information systems and communication technologies, and with non-technical elements (Aanestad & Jensen, 2011. p. 162). Therefore, analyses of information infrastructures need

to consider a broad range of socio-technical issues shaping the implementation process.

A basic principle of an information infrastructure is that it is never built from scratch; rather, it evolves from the installed base, the existing information system (IS) portfolio in specific contextual practices. As a part of this, the infrastructure shapes and is shaped by the work practice in an on-going co-construction process between technical and social elements (Monteiro, Pollock, Hanseth, & Williams, 2012; Star & Ruhleder, 1996). During the progression of an information infrastructure in any given context, the installed base may become very large and will shape its environment to an increasing degree. Similarly, the size and complexity of the installed base means that it becomes difficult to replace or change. Therefore, newer versions are adjusted or changed carefully in order to maintain backward compatibility with previous versions (Bowker & Star, 1999). This is a process of on-going negotiation and compromises for achieving stability or alignment (Latour, 1987).

In this regard, many studies do not refer to infrastructural design as construction, but rather conceptualize it as an "evolving socio-technical system" (Hanseth & Lyytinen, 2010, p. 4) or infrastructuring (Karasti, Baker, & Millerand, 2010; Pipek & Wulf, 2009) that needs to be carefully cultivated (Aanestad & Jensen, 2011). Hence, it is crucial to seriously engage with local contexts when designing information systems (Fitzpatrick & Ellingsen, 2012).

However, few researchers have explicitly addressed design strategies for infrastructure development, but Hanseth and Lyytinen (2010) specifically suggest:

- Designing simple IT capabilities that are initially useful
- Mobilizing many users, which frequently is promoted through the slogan "users before functionality"
- Drawing on the existing installed base, and
- Modularizing the II by building separately its principal functions and sub-infrastructures.

To some degree, this insight has spilled over into modern design methods. Typically, agile methods such as SCRUM, Extreme Programming (XP), and Kanban lean heavily on frequent interaction between users and designers (Kniberg, 2011). The essence of an agile development methodology is that users' needs are important for changing the course along the way and for ensuring a robust result. A principal communication tool between users and designers in these methods is "user stories", which are short narratives formulated by the users. The stories inform the vendor regarding the users' needs and enable the developers to design and deliver working software early on in the development process.

However, the design of generic systems adds a new dimension to the above-mentioned insights. While local practice is important for design, so also is the need to carve out generic elements and to adhere to global principles such as the openEHR framework. This inevitably creates a local/global tension that needs to be resolved (Star & Ruhleder, 1996). The move from local insights to developed generic concepts appears to be a crucial generification process that needs to be considered.

3. Method

Empirically, this paper presents the initial stages of a large electronic patient record (EPR) project, referred to as BigInvestment. After a prolonged bid for tendering process, the North Norwegian health region decided in 2011 to invest in new clinical ICT systems from BigVendor for all 11 hospitals in North Norway. The BigInvestment project was established for the 2012-2016 period. Moreover, it is estimated to cost 82 million EURO, which currently makes BigInvestment one of the most ambitious healthcare-related ICT projects in Norway. The Northern Norway Regional Health Authority employs about 12,500 person-years, which means there are many users that will make use of the new clinical systems and have to be heard in the development process. The University Hospital of Northern Norway (UNN) is by

far the largest of the 11 hospitals and has around 5,900 employees and 600 beds.

A key goal in the BigInvestment project was to make the new EPR more generic to allow users to tailor the EPR software to their specific needs, such as the possibility to define more structured content of the EPR, in line with an openEHR approach (Beale & Heard, 2007, 2008). The new EPR would also allow users to define templates for the support of specific user needs, such as standardized patient pathways for specific diagnoses. Accordingly, a core element outlined in the BigInvestment project was the involvement of end users, the healthcare practitioners who hold clinical expertise, in the development process. Therefore, over 100 clinicians from different healthcare professions and geographical locations were recruited to participated with BigVendor in six different EPR development tracks: decision module for psychiatry; surgery planning; process support, decision support, and structured record[1] (herafter named PDS); authorization and access control; e-prescription; and nursing care plans. The first five first tracks started in February 2012 and were expected to end in the beginning of 2014. The latter track, nursing care plans, started in spring 2013.

This study applies the interpretive approach to case study research to provide insight about the key mechanisms at play in the formative stages of generic systems (Klein & Myers, 1999; Walsham, 1995), and narrowed to the PDS development track. The epistemological belief in interpretive research emphasizes the understanding of social processes by getting involved inside the world of those generating them, and not by hypothetical deductions or predefined variables. The approach also assumes that social realities are not discovered, but interpreted (Orlikowski & Baroudi, 2002).

In line with the interpretive approach, the first author primarily collected the empirical data by becoming involved in the development process through different settings such as user-designer workshops, video conference meetings, participant observation at the vendor's site, formal and informal discussions with project members, document studies, and formal semi-structured interviews. The second author contributed in some of the semi-structured interviews. The data collection lasted from January 2012 to January 2013. In addition, the first author has worked as a nurse in different fields of the Norwegian healthcare service over the last 15 years. The second author has a long history of studying the implementation and use of ICT in healthcare, particularly about EPRs in hospitals.

Furthermore, the interpretive approach calls for detailed case descriptions covering the different sites involved (see Table 1), followed by an analysis of the data for potential analytical themes guided by the philosophical perspective of hermeneutics and the information infrastructure framework. The chosen philosophical perspective implies considering the entire data collection in an iterative and interpretive process. Therefore, our examination has been a back-and-forth process between fieldwork, case descriptions, and the use of relevant literature emphasizing the concepts of information infrastructure, generic systems, archetypes, agile development methodology, and the notion of translation. In addition, we discussed the data and case description with other members of the IS community in healthcare. The interpretive process, or the hermeneutic circle, constituted evolving issues that provided a new understanding about the development process. Accordingly, we need to understand the presented case as a complex whole "from preconceptions about the meanings of its parts and their interrelationships" (Klein & Myers, 1999; Walsham, 1995).

[1] Structured record points to the clinical use of pre-defined clinical contents items from a library of such definitions, in this case "archetypes". Then all recorded data will ultimately just be instances of the standard content definitions, which affords a basis for standardised querying to work (www.openehr.org).

Table 1. Timeline of Data Collection

Period	Formal meetings, workshops, video conference meetings	Informal meetings, interviews, participant observation
February-June 2012	• One formal meeting with key members of the BigInvestement project and employees in the trust (3 hours) • Eight days of workshops • Three video conferences: BigVendor, BigInvestment project, and users (approx. 6 hours)	• Three informal meetings with key members of the BigInvestment project (approx. 8 hours) • One interview with the hospital Department Manager
August-December 2012	• Four days of workshops • Three video conferences: BigVendor, BigInvestment project, and users (approx. six hours)	• One informal meeting with key members of the BigInvestment project (approx. 2 hours) • Three interviews with developers • Three occurrences of participant observation at BigVendor (approx. 9 hours)
January 2013		• Two interviews with managers at BigVendor

4. BigVendor's Design Strategy

During the last 25 years, BigVendor has accumulated high-level expertise and a great deal of knowledge about the Norwegian healthcare service and the complexity of developing and implementing ICT systems that support the heterogeneous healthcare domain. The vendor currently enjoys approximately 73 percent of the hospital-based EPR market in Norway relative to the number of institutions that run the BigVendor portfolio (Ellingsen & Monteiro, 2012).

Over several years, BigVendor has applied an agile development approach such as Scrum and Extreme Programming (XP). Short development cycles lasting two to four weeks and continuous release of working software characterize the agile approach. The approach implied working in close collaboration with customers (users) in earlier development projects. A principal communication tool between the users and developers was user stories informing the vendor about the users' needs. User stories are short (2-3 lines) but sufficiently detailed stories, based on requests from the users, and formulated in non-technical language. Both users and developers could formulate these stories, although they were created in order to add value for the user. A developer at BigVendor described it by saying: "Every time we make software based on a user-story, somehow we make the users satisfied, because the users will gain a value of the delivered software".

Relying on an agile development methodology, Big Vendor found it appropriate to organize the developers into several teams mainly supporting specific software development. Each team included six to eight people who had daily meetings and worked in close collaboration with each other. Each team used its own backlog, which was a collection of user stories in prioritized order, to guide the development process. While discussing software in progress, the developers frequently turned to the specific user story as a guideline for ensuring that the software produced or modified was valuable to the users.

Approximately five or six years ago, BigVendor started to use a model-driven architecture, which culminated in 2011 by its decision to use the openEHR architecture. The vendor's decision also conformed to the National Norwegian strategy on archetypes. By committing to the openEHR architecture and the notion of archetypes, BigVendor started to use the the open source library from Ocean Informatics[2]. According to BigVendor, openEHR provides an excellent model (see Figure 1) of the generic core of the intertwined and heterogeneous healthcare domain, and the model serves as a fundament for model-driven design.

OpenEHR's model illustrates how the overall clinical workflow proceeds, independent of the healthcare domain or specific clinical professions. However, it is easy to target the model to a specific clinical context (e.g., when a physician (investigator) prescribes a new medication, a nurse (investigator agents) receives this message, and forwards "the baton" to the next person in line—another nurse, who gives the medication to the patient). The patient, nurses, and physician observe the effect of the new medication (observations) and inform the physician (investigator) if other "instructions" have to be made. The whole process demands coordination and communication by and between clinicians and administrative personnel directly and indirectly involved with the actual patient. Moreover, how the workflow in particular proceeds will differ depending on the department. Another argument for using the openEHR architecture was the separation of technical and clinical concerns. According to the BigVendor's strategy this would make it less resource-demanding to maintain the system after it was developed. Just consider the following statement:

> *The profit by using the "archetype approach" is that it allows us (the developers) to live in "our own little developers' world"—though, not the developers who implement the system. (…) the designers don't need so much clinical contextual knowledge, and the domain experts don't need extended technical skills—but we have to know a little bit of each other's domains.* (Manager, BigVendor)

The openEHR architecture implies that the vendor offers a generic platform, also promoted as an important benefit for the clinical community as the separation enables the users to source and build archetypes (standardized clinical information), and recommend screens that is workflow description tailored to different local needs.

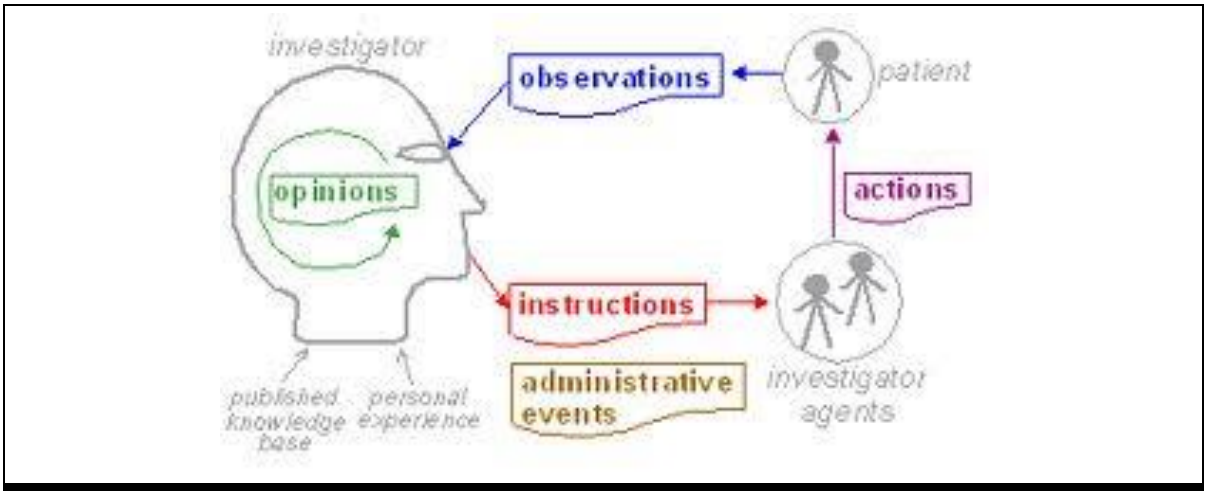
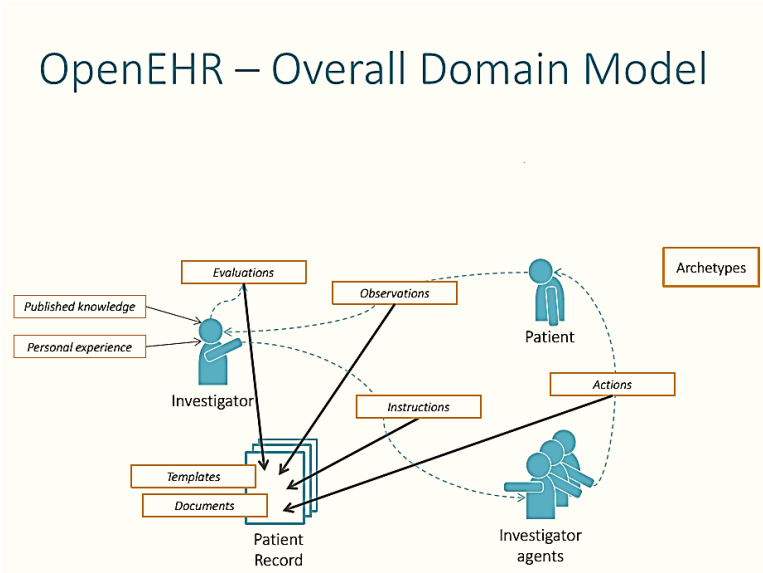


Figure 1. openEHR Clinical Process

[2] www.oceaninformatics.com



The Figure 1. openEHR Clinical Process

5. Case

5.1. Phase 1: Invitation to Write User Stories for Generic Functionality

The PDS development track started out ambitiously: The PDS development track's overall goal was to carve out the system's generic platform, which is the fundament for enabling flexible process- and decision support. As a point of departure for the collaboration, BigVendor invited the users to make user stories addressing needs for generic functionalities by linking the invitation to openEHR's clinical process (see Figure 1). To emphasize the innovation potential of the new technology, BigVendor encouraged the users to think about their future needs. One developer said: "You should forget about your current EPR and make wishes for what you really need".

The aim of separating the users' from their present EPR system was to make them more emancipated when communicating their needs. Yet, the strategy of making requests as generic as possible turned to be rather difficult, as exemplified in the following user story: "As a healthcare practitioner, I want the system to make it possible to easily display all compulsory treatment given" (Therapist, Hospital).

This particular user story started with the line "As a healthcare practitioner..." that address a general healthcare role. Nevertheless, the required need of "an easily display of all *compulsory treatement* given" strongly reflected a need for healthcare practitioners working in specialist psychiatric care because the Norwegian legislation only permit use of compulsory treatment for patient within jurisdiction of compulsory psychiatric care. However, this user story was just randomly chosen to examplify the difficulties of making generic requests separated from their professional occupation and clinical work. Other user stories were incompatible with each other. For example, in a department for rehabilitation,

the clinicians typically approached a patient's problems in interdisciplinary manners, as indicated by the following quote:

> *As users of the system, we like to write and approve parts of a shared document and at the same time have the opportunity to see what other healthcare practitioners have written to ensure the quality of the document and make a comprehensive presentation.* (Occupational therapist, Hospital)

In comparison, several of the participating physicians from different specialities expressed that, from their point of view, they were overwhelmed by too much information. One physician proposed another user story that was clearly incompatible with the occupational therapist's user story: "As a physician, I like to have the opportunity to avoid seeing letters, therapist notes, etc., but must have access if I like to see them."

In addition, there were user stories pointing to general issues of different clinical roles as illustrated by the following examples:

> *As a secretary, I want a warning on the display if different kind of patient information has to be verified or/and an update is necessary.*

> *As a nurse, I want to get an overview of the examinations, lab tests, etc ordered by the physician in one screen shot on the computer's display.*

These user stories addressed definite needs in a more general manner, but, unfortunately, along with several other user stories, they were too fragmented in that they were not connected to a spesific moment in the clinical process (see Figure 1) **when** the clinicians needed EPR support to perform a particular task.

The users contributed with about 100 user stories during the first collaborative phase, but the developers thought that they lacked coherence. The fragmented, context-bound user stories were not a convenient communication tool anymore because the user stories did not give the developers the information necessary to build the generic platform. BigVendor realized that approaching the user-developer collaboration with traditional short and to-the-point user stories did not facilitate the two-level modelling approach. A developer from BigVendor stated: "We are not going to develop specific functionality for surgery planning, but rather generic functionality that makes surgery planning possible".

To develop a generic EPR platform enabling local tailoring, the developers needed a new way of approaching the development process.

5.2. Phase 2: Longer User Descriptions—A Foundation for Shared Concepts

Turning toward a two-level modeling approach meant shifting from developing specific software based on specific user needs to developing generic software enabling local user needs. Still, the developers called for insight into the "reality" in which the generic software intended to provide support. Instead of asking for more user stories, the vendor changed strategy and asked the users to make descriptions of clinical processes and activities surrounding the processes. At the vendor's request, a participating specialist physician (SP) made a comprehensive description, referred to as a clinical narrative, related to a specific clinical symptom. It proved to be successful and the user-developer collaboration proceeded along similar tracks. The selected clinical symptom, well known from the specialist physician's practice, was urination disorders among elderly men. The clinical narrative started with the specialist physician receiving an elective referral from the general practitioner (GP):

> *NN is a 73-year-old man. Previous diseases: hypertension, cardiac arrhythmias, diabetes—insulin dependent. Medication: Metoprolol 100mg*1, Warfarin dosage by list, Insulin dosage by list. Current medical problem: Increased urination affiliations latest years. Increased need to urinate, 3-4-time every night and at daytime every second hour.*

> *Small micturition volumes. Weak urine flow. The patient express a lasting need urinate, even when urination is done. No urine leakage, but sometimes "urgent incontinence". On two occasions, the patient has observed blood in the urine. PSA slightly increased to 6.9, controlled by two measurements. Refer the patient for further examinations. Best regards, GP BB.*
>
> *Based on the GP's referral, the specialist physician (SP) addresses three problems: urination disorder, haematuria and elevated PSA—all together making the SP to suspect prostate cancer. The SP refers the patient to the outpatient clinic for an extended examination. As a part of the process, the patient has to go through several examinations, for example, different blood tests, x-rays, cystoscopy and biopsy. The result of these examinations give way for how to proceed with the patient's problems. According to the GP's information about the patient's medication, the patient has to cease Warfarin three days in advance of the appointment at the outpatient clinic. Warfarin exposes the patient to severe bleeding during surgical intervention, like the planned cystoscopy and biopsy. In addition, when making the referral to the outpatient clinic, the SP has to ensure the urgent matter of the patient's problem in accordance with the national guidelines of prioritized treatment.*

Although the patient pathway description was very thorough, the SP emphasized that the story reflected her point of view and not the intertwined story of the healthcare practitioners involved. However, the presentation initiated a rich discussion where healthcare practitioners added their perspective and concerns to the clinical description:

> *The patient uses anticoagulation and this medication has to be ceased before the intervention. In practice, the physician gives the instruction to "cease medication", and the nurses put the instruction into action. So who is going to give the patient this message?* (Nurse, workshop)
>
> *When the patient arrives the hospital for surgery, who has the role of ensuring that the patient actually has ceased the medication?* (Nurse, workshop)
>
> *Maybe we could have some pop-up alarms for patients using medication that have to be ceased as well as pop-ups for patients approaching deadline of the waiting-lists?* (Secretary, workshop)

By discussing the clinical narrative, the users contributed with information about the collaborative work necessary to bring a patient through the clinical process. For instance, while the physicians performed interventions and determined the next step, nurses and secretaries had important roles in coordinating different examinations, in preparing the patients for different procedures, and in ensuring that the necessary equipment was available (cf. Figure 1). From the developers' point of view, the context-bound information gave them important insights into the intertwined clinical collaboration and contextualized tacit knowledge necessary to take the patient through the clinical process. One BigVendor developer said

> *We* [the designers] *have to open the "magic black box" of what actually happens from the time the patient get his notice of admission till he shows up, and the way through his pathway of treatment and care. We ought to know who ensures and enables that the clinical process comes through.*

The discussions "opened" the "magic black boxes", which revealed a very complex network of people and tasks—much more complex than the OpenEHR clinical process model managed to picture. A BigVendor developer underscored the complexity by stating that "One by one the steps in a clinical process are quite simple, but putting the steps together makes the process very complex".

By taking the complexity of clinical processes into the developers' context, the narratives served as a

map to develop the generic platform. In addition, the shift from single-user stories to narratives forced the vendor to evaluate the internal team's organization. In earlier development projects, the developers had "zoomed" into "bits and pieces" of the particular functionality to be developed. The two-level model approach addressed a higher degree of complexity for every single developer because developing the generic software called for an overall understanding of **what** a clinical process actually was and accordingly an overall understanding of the concept "generic platform" .

5.3. Phase 3: Engaging with the openEHR Framework in Design

Looking into the user-developer collaboration, the discussions about the context-bound narrative materialized a variety of needs from the different healthcare professions. However, even if the healthcare practitioners discussed software support from different clinical experiences, the steps in the process were recognizable for every clinician involved. In that way, the discussions about software supporting specific clinical processes generated desired software support on a more general level. In other words, the different health personnel could agree upon relevant software independent of a specific clinical context. Moreover, the involved healthcare practitioners and developers had come to a common understanding of what a process- and decision system actually is, and the potential of such a system. In this sense, the user-developer collaboration was "on-track" and benefitted the development process.

Still, the adherence to the two-level modelling approach escalated the challenges for BigVendor's internal development process. Generally, a key innovation of the openEHR framework is the possibility of reuse of clinical information in numerous contexts. The reuse is related to the concept of a two-layered model that separates the clinical content from the reference model (information model), in which the clinical content strictly must be based on the pre-defined archetype "library".

Though, in the initial development process, the developers had not taken into account the "concept of reuse", and the consequenses this concept claimed for developing the generic platform. For example, initially in the development process, a measurement registration (e.g., blood pressure) in the patient's medical curve would appear as a stand-alone registration not connected to a document in the patient's EPR. Moreover, the new system did note provide access control for stand-alone registrations of clinical information, and consequently these registrations did not fullfil the national legislation's claim of reuse of sensitive personal information.

To comply with this claim, the developers had to use the openEHR's notion "composition" as a overall, collective term for different documents (for example discharge summaries, antenatal visits or operative notes). Then, by connecting every single registration to a composition made it possible to manage access control for all clinical information about a patient, and consequently to reuse information in other documents. This example highlights the complexity the developers had to face: the openEHR framework, the clinical workflow, and how the clinical information had to relate to the national legal framework. Another technical implication in the initial stage was the interaction between the vendor's software and the open source library from Ocean Informatics. The clinicians addressed the need to register several procedures connected to, for example, a surgical intervention or clinical information (height, weight, allergy, etc) connected to an examination in one document, but initially the integration between the open source software and the vendor's software could not support multi-registrations of clinical information by archetypes in one document.

Nevertheless, the developers characterized the initial work with a two-level system as an enormous conception to perceive:

> *If you make software for surgery planning only, you can (easily) design a screen and add necessary fields…* [However] *if you are going to make something generic, you don't know the all pieces inside because what you make should not only work for surgery planning, but also for the laboratories and so on … and the concerns and complexities becomes much bigger.* (Developer, BigVendor)

To overcome the challenges of carving out the generic process-supporting elements of a clinical

process in general, the two-level modeling approach called for a lot of work in-house and a massive upfront design in contrast to earlier development projects characterized by short development cycles and early delivery of working software.

5.4. Phase 4: Changing Communication Patterns with the Users

The necessary up-front design resulted in less regular feedback and user-developer interaction; consequently, the developers raised another concern. One said: "There are lots of small user stories that are impossible to implement, because the basic framework is not ready yet—and maybe the users feel that they haven't gotten any feedback on their requests".

This made it even harder for the developers to maintain an overview of how the generic elements matched the users' practices, and increased concern about whether the generic elements inherit the qualities of making local tailoring possible later on in the development process. When discussing one of the generic software modules, a developer complained:

> *We don't know the context where this will be used... We need feedback from the users to guide us... What is the use case and what do the clinicians require? Do we make a functionality that nobody needs?*

Accordingly, the vendor had to find a way to present generic software-in-progress to the clinicians, and a way to facilitate dialogue between clinicians and developers during the presentations. To face the concern of an extended development process with no working software to present, the vendor saw it as important to increase the users' understanding of the two-level model approach and its inherit qualities of local tailoring, and the vendor aimed to trigger the users' ability to give feedback on software "in-progress". Moreover, BigVendor made a presentation of the two-level model concept by using the LEGO® analogy (see Figure 2), and broadcasted the presentation to users at all the hospitals. As one developer said: "We are going to build a LEGO® city, but at this stage we are making the description of how to put the single bricks together".

Once again, the context-bound clinical narratives became crucial in the development process by serving as a clinical backdrop for the generic "LEGO®" elements. The users welcomed the presentation of linking the two-level model concept to LEGO® constructions, and furthermore the translation of generic software into imagine local software based on clinical narratives and earlier discussions.

However, rewriting the context-bound clinical narratives into generic information to guide the vendor's internal development process and translating the generic software back again into a clinical context added a layer of translation work for the developers to keep the development process on track. One BigVendor developer said:

> *It is challenging to design something general, but we ought to make the generic in the context of a specific context. In that sense, we enable locally tailored functionalities. In addition, I have to ask, if we don't make it related to a specific context—what will we be able to present to the users?*

Moreover, as the development process proceeded, the vendor placed a stronger focus on testing generic software in smaller groups of users. When testing, the users actually tried functionalities from the new EPR system to support tasks they normally did during their daily practices, but, of course, in a test-environment, not influencing the working EPR system. The users tried the software by themselves and with instructions from the vendor's test personnel and gave feedback directly to the vendor.

Finally, the involved clinicians initiated an interesting discussion related to reorganization of the clinicians' workflow as a response to the new evolving system that paved the way for supporting clinical work. For instance, the new system made it possible for physicians to set up an appointment for a patient in an outpatient clinic by themselves, in contrast to the limitations of the existing EPR where

physicians have to send a referral to the secretary who scheduled the appointment. In addition, documenting patient information by using archetypes simplifies the documentation process (e.g., by reusing information and check-offs), and this triggered the physicians to do more of the documentation by themselves. Besides, implementing the new system calls for tailoring the system to different clinical contexts. Accordingly, the secretaries may play an important role as domain experts because they have high-level knowledge about clinical workflow conditions because of their everyday practice.

An archetype compared with a LEGO® brick and represent structured clinical information (e.g. blood pressure that can be shared and retrieved in several different documents in the EPR system).

When numerous healthcare practitioners are going to use and share the same bricks, it is important to ensure that the bricks fit together.

A common setup for the bricks is required and must be displayed for the users as template.

Figure 2. Archetypes as LEGO® brick constructions

6. Discussion

6.1. Generification Strategies with a Foothold in Local Practice

From the outset, it is easy to believe that designing a generic system implies building it from scratch. In our case, for example, the vendor emphasized building the new system in accordance with the emerging openEHR framework and asked the users to forget about their existing functionality and think of something completely new. However, it is important to have in mind that the vendor controlled the principal existing information infrastructure in the hospital, including thousands of associated users. In such a perspective, the commissioned new openEHR-based system represented an extension and a development of the existing installed base of EPRs in the North Norwegian hospitals.

The decision to apply the OpenEHR framework spawned a set of generification strategies. In the first workshops, the users were invited to forget about their existing EPR system and asked to formulate user stories describing generic functionality with a clear reference to the OpenEHR framework. Unfortunately (alas as expected), this did not work out because it did not provide any meaning for the users to think in conceptual terms not bound to a specific context. In this regard, several studies have pointed out how a local foothold is essential in developing information infrastructures (see, e.g., Berg & Goorman, 1999; Star & Ruhleder, 1996). Our case illustrates how detailed and context-bound insight appeared to be a much better way of identifying what the users' actually needed.

An interesting finding from our case is how the users not only reflected on needs supported by the new system, but also on the current state of their local practice and how it might change. As technical advancements and new functionality was presented to the users, their feedback to the vendor emphasized the need for adjusting, changing, and improving current work practices: physicians could do more writing, ordering of tests, and examinations, and then secretaries and nurses who had done these tasks earlier could do other things. A suggestion emerging from the reflections was that of training the secretaries to be clinical domain experts. This indicates and underscores how the design of generic software occurs in co-construction with local practice (Monteiro et al., 2012), a striking contrast to the ideals of a two-level modelling openEHR approach.

While openEHR represents a relatively complete model of healthcare, it is interesting to note that a shared conceptual understanding between users and designers gradually emerged through insight from local practice and the installed base. An illustration is the long narrative about the patient with the urine disorder. The narrative was used repeatedly for developing a conceptual understanding of what a clinical process actually is: how different health personnel engaged with each other in various stages of the process, what kind of process and decision support was needed during the process, and by whom. As a generification strategy, this suggests that local processes of learning and the subsequent development of shared concepts are quite powerful compared to just applying ready-made models such as the openEHR, even if these are thoroughly worked out. For example, from the outset, users and designers interpreted the key concepts decision support and patient pathways quite differently, but, through interaction in the workshops, they negotiated and reached a shared understanding. This also indicates why the initial focus on short user stories vanished because these failed to capture the bigger contextual picture—the infrastructural interdependences across the various practices. The developers instead needed longer descriptions of current practice in which the users reflected on the support they received today and the improvements they would like to have. The vendor increasingly understood the infrastructural interdependences and organized the development teams according to contextual areas. In this way, concrete results, as the translation of generic software, were presented and discussed with specific user groups.

6.2. Generification by Translation: Archetypes, Templates, and Building Blocks

In Section 6.1, we discuss a range of generification strategies that circle around a shared understanding and that sometimes imply a change in the users' practice. However, it is also clear that, due to its global foundation, generic software creates a tension with local practice that is hard to reconcile. Star and Ruhleder (1996, p. 114) argue that "An infrastructure occurs when the tension between local and global is resolved". In our case, we see users' work in daily practice as local, and the design in accordance with the international openEHR framework as global. On one side, the users in local practices have their specific needs, and, on the other side, the openEHR model prescribes design to be done in a certain way, that put restrictions on the design according to how the customizable components of the generic software should be modelled and developed. We believe that the notion of translation (Carlile, 2004) helps to resolve the global/local tension and see it as an essential generification strategy because the develpopers used translation as a mean to maintain and sustain several perspectives simultaneously. Still, as our case described, the translation processes created a lot of work for the developers.

The vendor found itself in a middle position between the users and the technical limitations within the international openEHR framework (the open source library, the clinical process model, and two-level modeling approach).These two positions were difficult to align in the development process. Because the designers developed the software in accordance with the archetype model, the consequence of contextual user input was that the designers could not use the user stories or workplace descriptions into the design work directly or unreserved. The designers had to translate the context-bound user stories or workplace descriptions into technical or conceptual counterparts that could inform the design of the customizable components in openEHR—an extremely complex and cumbersome task. As one of the vendor's developers mentions: "We got a lot of experience with the work processes at the university hospital and see things we could solve quite straightforwardly, but then "the generic train" goes in a complete opposite direction… This is very challenging for us".

Later, when the designers should present the developed software (i.e., the generic components), they had to explain how the new features of the customizable components/generic building blocks could be tailored to a particular context. A developer notes:

> *We make it generic, but we try to make it in the context of the specific in a way that we always enable the specific, but this has been challenging (…) if we don't manage to concretize it for a specific context, what should we present otherwise?*

Increasingly, the vendor was not able to present working software for the users and therefore had to compensate by explaining how the archetype-based framework might support customization for experienced users. This represented a translation process between IT used in the users' existing practice and IT used in an imaginary future practice. BigVendor frequently used metaphors such as the ability to build a LEGO® city of LEGO® bricks. In turn, the participants in the BigInvestment project created their own metaphor by using PowerPoint as an illustration: "BigVendor offers us a PowerPoint, but we have to make the presentation by ourselves". In this way, both the vendor and the project management wanted to appeal to the users' imagination of how the system might be customized and used.

While the users welcomed the presentation, it still represented a gap between possible functionality and working functionality. In some stages of the development process, the designers had very little working functionality to present to the users. Similarly, the developers increasingly questioned whether the generic functionality they were developing was useful or requested by the users.

6.3. Generification by Widening the Distance to the Users

Several studies have pointed to the shortcomings of traditional design methods characterized by a clear distinction between design and use (Hanseth & Lyytinen, 2010; Karasti et al., 2010; Pipek & Wulf, 2009). In particular, Hanseth and Lyytinen (2010) promote several guidelines for intervening in large-scale information infrastructures. Some of these are stepwise design and short iterations: design for usefulness, make things simple, and produce working software from the very first release. Today, many of these guidelines have been incorporated in agile design methodologies and expressed through the agile manifesto: "Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation" (Beck et al., 2001).

However, this strategy does not necessarily pay off in the design of generic systems. Our case illustrates that the design strategy gradually changed throughout the project period. From initially being characterized as a lightweight design process, it increasingly turned towards heavy up-front design.

The two-level modelling OpenEHR architecture required implementing support for a number of different functionalities often not relevant at the current stage in the development process. In addition, designing a tool (e.g., the building blocks) instead of a tailored product meant that the designers had to invest a huge amount of work in-house to lay the groundwork for, and to carve out, the generic elements of the software. This suggests that the design strategy of large-scale generic software tends to head in the opposite direction of agile development because it increasingly requires a lot of planning and up-front design.

In one way, this resembles a traditional design strategy that splits designers and users. However, framing the process as a traditional design strategy would be a mistake because our study illustrates the necessity of a close and transformative design/use interaction. We rather see this as a generification strategy where the vendor needs to take a step back and strategically plan how to conceptualize and develop the generic system. A characterization of this process is that it widens the gap between designers and users, which makes the actual development process increasingly "resilient" in proportion tovarious user input. Accordingly, the process of developing a generic system has questioned the traditional agile development methodology, but confirmed the importance of a user/developer collaboration in one way or another. This seems to carve out some characteristics of the relationship between design and a developed information system. An agile design tends to offer a flexible design process, but may lead to a system with fewer customization capabilities. Agile design clearly puts the user at center stage, and it requires the developers to be adaptive to the users´ changing and emerging demands. Accordingly, software has been closely tailored to a specific local practice, which for the same reason may be more difficult to use in a different practice. On the other hand, heavy up-front design of a generic system may allow few actual changes in the design process, but will offer (if the promises come through) more customization capability of the completed generic system for the users.

7. Conclusion

Not surprisingly, our case is yet another example of solving socio-technical challenges during the development process of a large-scale EPR system. However, this paper particularly focuses on how the generification process plays out in the formative stages of generic systems by illustrating the socio-technical key mechanisms at play in the initial phase. The promoted idea of a two-level modelling approach is a more distinct boundary between technical development and clinical particularities, and, as a striking contrast to the idea, our case demonstrated how the design of generic software occurred in co-construction with local practice. Furthermore, the evolving co-constructive collaboration captured the socio-technical key mechanisms at play. These mechanisms are the vendor's generification strategies of solving the local/global tension by translation and adjusting the design strategy from lightweight methodology to a modified up-front design.

Our findings have implications for practice: First, the vendor realized that the development process entered new terrain, which meant the vendor had to experiment with it's design strategy. Accordingly, we suggest that designing a generic system calls for a flexible vendor willing to change and adjust its development strategy along with the evolving project. Moreover, in our case, it is important to notice how the vendor managed to resolve the global/local tension by translation in the meaning of allowing several perspectives to be maintained and sustained simultaneously. The vendor made a significant effort to keep the user-developer collaboration on track, and, by the effort, the vendor accentuated the necessity of the users' involvement during the process. Second, to strengthen the user-developer collaboration, we highly recommend giving the user-participants, at the very early stage of a development project, a basic understanding of software design and raising their skills in making precise contextual narratives. Because designing generic process- and decision-supporting software addresses a need for contextual information that reveals all the particularities and concerns in a specific work process. Consequently, the users should make the clinical contextual knowledge available for the designers by describing what every professional role in a work process actually do, how professionals perform the activities, and importantly—but often forgotten—why they act the way they do. Additionally, it is a well-known issue that clinical personnel often do not have time or do not want to spend time in projects with no direct influence on their everyday practice. Third, even if the case did not put a particular focus into the BigInvestment management's role, the management's engagement in recruiting clinical personnel and in making it possible for the clinicians to participate in a project is important.

Regarding implementation, it is tempting to announce a new dimension of the local/global tension likely to come into play; namely, the clinical specification of the customizable components – the openEHR archetypes. The new dimension of the local/global tension addresses a need, on one hand, to tailor openEHR archetypes to a specific local practice, and, on the other hand, to make sure that the tailored local archetypes adhere to the standardized framework to ensure interoperability. Accordingly, local user tailoring can cause risks to vendors, as they have to ensure backward compatibility on a system that they don't totally control anymore.
In addition, local/global tension raises other interesting issues related to which socio-technical challenges will emerge in the transition from free text documentation to a structured process- and decision-supporting EPR system. Consequently, the implementation of the new generic EPR system calls for further research.

References

Aanestad, M., & Jensen, T. B. (2011). Building nation-wide information infrastructures in healthcare through modular implementation strategies. *The Journal of Strategic Information Systems*, *20*(2), 161–176.

Baldwin, C. Y., & Woodard, C. J. (2008). The architecture of platforms : A unified view (Harvard Business School Finance Working Paper No. 09-034). Retrived from http://ssrn.com/abstract=1265155.

Beale, T. & Heard, S. (2007). openEHR release 1.0.1. Archetype definitions and principles. *openEHR Foundation.* Retrieved from http://iten.behdasht.gov.ir/uploads/256_1197_archetype_principles.pdf

Beale, T. & Heard, S. (2008). openEHR release 1.0.2. Architecture overview*. openEHR Foundation.* Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.204.1693&rep=rep1&type=pdf

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001). *Agile manifesto*. Retrieved February 08, 2013, from http://agilemanifesto.org/

Berg, M. (1999). Patient care information systems and health care work: A sociotechnical approach. *International Journal of Medical Informatics*, *55*(2), 87–101.

Berg, M., & Goorman, E. (1999). The contextual nature of medical information. *International Journal of Medical Informatics*, *56*(1-3), 51–60.

Bernstein, K., Tvede, I., Petersen, J., & Bredegaard, K. (2009). Can openEHR archetypes be used in a national context? The Danish archetype proof-of-concept project. In K.-P. Adlassnig, B. Blobel, J. Mantas, & I. Masic (Eds.), *MIE* (pp. 147-151). IOS Press.

Bowker, G. C. & Star, S. L. (1999). *Sorting things out: Classification and its consequences.* Cambridge: The MIT Press.

Carlile, P. R. (2004). Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization Science*, *15*(5), 555–568.

Chen, R., & Klein, G. (2007). The openEHR Java reference implementation project. *Studies in health technology and informatics*, *129*(Pt 1), 58–62.

Chen, R., Klein, G. O., Sundvall, E., Karlsson, D., & Ahlfeldt, H. (2009). Archetype-based conversion of EHR content models: Pilot experience with a regional EHR system*. BMC Medical Informatics and Decision Making*, *9*(33). Retrieved from http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2715396/

Ellingsen, G., & Monteiro, E. (2012). Electronic patient record development in Norway: The case for an evolutionary strategy. *Health Policy and Technology*, *1*(1), 16–21.

Fitzpatrick, G., & Ellingsen, G. (2012). A Review of 25 Years of CSCW research in healthcare: Contributions, challenges and future agendas. *Computer Supported Cooperative Work, 22*(4-6), 609-665*.*

Garde, S., Knaup, P., Hovenga, E., & Heard, S. (2007). Towards semantic interoperability for electronic health records. *Methods of information in medicine, 46*(3), 332–43.

Hanseth, O., Bygstad, B., Ellingsen, G., & Johannessen, L. K. (2012). ICT standardization strategies and service innovation in health care. Paper presented at the International Conference of Information Systems (ICIS), Orlando, FL*.*

Hanseth, O., & Lyytinen, K. (2010). Design theory for dynamic complexity in information infrastructures: the case of building internet. *Journal of Information Technology, 25*(1), 1–19.

Kalra, D. (2006). Electronic health record standards the need for generic and requirements for represent. *Medical Informatics*. Retrieved from http://eprints.ucl.ac.uk/2292/1/schattauer_30_2006_1_136.pdf

Karasti, H., Baker, K. S., & Millerand, F. (2010). Infrastructure time: Long-term matters in collaborative development. *Computer Supported Cooperative Work*, *19*(3-4), 377–415.

Klein, H. K. & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, *23*(1), 67-93.

Kniberg, H. (2011). *Lean from the trenches: Managing large-scale projects with Kanban.* The Pragmatic Bookshelf. Dallas, Texas Raleigh, Raleigh North Carolina.

Latour, B. (1987). *Science in Action*. Harvard University Press, Cambridge, MA.

Monteiro, E., Pollock, N., Hanseth, O., & Williams, R. (2012). From artefacts to infrastructures. *Computer Supported Cooperative Work*, *22*(4-6), 575-607.

Orlikowsi, W., & Baroudi, J. (2002). Studying information technology in organizations: Research approaches and assumptions. In M.D. Meyers, & D.E. Avison (Eds.), *Qualitative research in information systems* (pp. 51-78). Sage Publications UK.

Pipek, V., & Wulf, V. (2009). Infrastructuring: Toward an integrated perspective on the design and use of information technology. *Journal of the Association for Information Systems, 10*(5), 447–473.

Pollock, N., Williams, R., & Adderio, L. D. (2007). Global software and its provenance : Generification work in the production of organizational software packages. *Social Studies Of Science*, *2*(4), 254–280.

Pollock, N., & Williams, R. (2008). *Software and Organisations*. Routledge.

Sahay, S., Aanestad, M., & Monteiro, E. (2009). Configurable politics and asymmetric integration: Health e-infrastructures in India. *Journal of the Association for Information Systems, 10*(5), 399–414.

Star, S. L., & Ruhleder, K. (1996). Steps toward an ecology of infrastructure: Design and access for large information spaces. *Information Systems Research*, *7*(1), 111-134.

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

Wang, M. (2007). C*ultivating the "generic solution" the emergence of a Chinese product data management (PDM) software package*. PhD thesis. University of Edinburgh.

Wollersheim, D., Sari, A., & Rahayu, W. (2009). Archetype-based electronic health records: A literature review and evaluation of their applicability to health data interoperability and access. *The HIM Journal*, *38*(2), 7–17.

About the Authors

x

x

x