Open generification

Abyot Asalefew Gizaw,* Bendik Bygstad & Petter Nielsen

Department of Informatics, University of Oslo, Oslo, Norway, email: abyotag@ifi.uio.no; bendikby@ifi.uio.no; pnielsen@ifi.uio.no

Abstract. To what extent can software ‘travel’ to organizations and countries for which it was not designed for, and how important are local contexts for a successful design and implementation of generic software? Information systems researchers have differing views on this, some emphasizing the strengths of the generic and others the importance of contextual aspects. Contributing to this debate, Pollock and Williams have coined the term generification in order to describe how large vendors succeed in globalizing software packages through management by community, content and social authority. In this paper, we explore an approach that we call open generification, which extends Pollock and Williams’ work in the sense that we acknowledge the need for and the feasibility of generic software, but propose an alternative model for the governance of it. Open generification is not about managing the community of users attached to a software package by homogenization or segmentation but aims at addressing the diverse needs of the community the software is expected to serve. Our empirical basis is a longitudinal study of the development of an open-source health information system software (District Health Information software version 2), which is being used in more than 47 countries. Its success is attributed to a continuous interplay between generic and specific software and continuous cycles of embedding (implementing the global in the local context) and disembedding (taking local innovations into the global). We identify and discuss the contingent mechanisms of this interplay.

Keywords: generification, health information system, global software package, free and open-source software

INTRODUCTION

During the past two decades, one persistent research stream in information systems (IS) has investigated the local, contextual and embedded aspects of development and use of information technology (IT). This contextual research stream emerged in the 1990s as a critique of positivist IS research for its emphasis on the transformative powers of IT and lack of regard to context. Much of what may be regarded as accepted insights in the business (and business school) communities subscribe to this positivist view: the strategic use of IT (Pearlson & Saunders, 2009), the IT alignment literature (Chan & Reich, 2007) and the enterprise systems research (Davenport, 1998).
The critique raised from the contextual stream was based on two assumptions. First, it argued that technology is not a thing, but rather an embedded part of a socio-technical network, consisting of a Web of human, social and technical elements (Kling, 1993; Suchman, 1987). Understanding and dealing with this network require sensitivity for history and context and also imply that the ‘impact’ of a technology may vary significantly depending on the local setting (Avergerou, 2002; Walsham, 2005). Second, it argued, from an interpretive epistemological point of view, that the use of IT is not determined by the technology itself but that the user has a large degree of interpretive flexibility (Pinch & Bijker, 1984; Knorr, 1981). Users interpret and use a technological artefact according to their own needs, and not necessarily as the designers intended. While not coalescing into one consistent research stream, a number of seminal articles share this focus on context and situated actions. For example, Star & Ruhleder (1996) investigated the growth of information infrastructures and found that there are no ‘genuine universals’ in large-scale information technologies (112). Other examples include Walsham (2001), Avergerou (2002), Ciborra et al. (2000) and Hanseth et al. (1996).

Then, in the mid-2000s, the contextual stream came under scrutiny from an unexpected camp, namely, Science and Technology Studies. Pollock et al. (2003, 2007) argued that the contextual stream tended to exaggerate the importance of local conditions and published a few papers that investigated the process of generification of software packages. Their interest was explicitly not to elaborate on how technologies are adapted to work within particular settings, but rather to understand how they are built to work across a diverse range of organizational contexts. Building on longitudinal case studies of the adaptation of SAP in the university sector, they uncovered a subtle and complex interplay between vendors, consultants and users that managed the balance between generic software and organizational diversity.

The concept of generification is attractive; it explains why many ‘generic’ solutions, such as SAP and Microsoft Windows, can be developed in interaction with a large user community, adapted over time and used globally. However, for us, it raises concerns that the process of generification is characterized by the following:

- It is a top-down process, dominated by the software vendor, such as SAP.
- It is a somewhat manipulative process; it treats user groups differently, not according to their needs, but according to their customer value.

As a consequence, generification is a closed process, controlled by the vendor with little, if any, room for innovation as a distributed activity. The development of the internet (Hanseth & Lyytinen, 2010), practices of free and open-source software (FOSS) and the emergence of open innovation (Chesbrough, 2003) present very different approaches. Our intention here is not to compare a software package produced by a global vendor against a free and open-source product. It is rather doing a qualitative analysis between design practices adopted in free and open-source product and commercial software product. A distinguishing aspect that we aimed to emphasize is this aspect of both software products where design is conducted at global level but used locally at multiple countries and organizations.

In this paper, we explore an approach we call open generification. It extends Pollock and Williams’ work in the sense that it acknowledges the need and feasibility of generic software while at the same time emphasizing the importance of analysing the diffusion of systems not
only from the producer’s perspective but also from the users’ point of view. We draw on the insights of the contextual tradition by focusing on the need for local adaptations and innovations. And we move beyond the localist’s perspective and examine practices of openness and collaboration for feeding back local innovations to a global process of producing generic software. Our research questions are as follows:

- How can we conceptualize a process of open generification?
- What are the contingent mechanisms for successful use of this process?

The paper proceeds by a review of the generification literature and related theoretical frameworks. To develop our argument, we draw on findings from a longitudinal case study in the health information system programme (HISP). HISP is a long-term action research programme coordinated by the University of Oslo, which designs and implements District Health Information software version 2 (DHIS2), an open-source health information system used in more than 47 African, Asian and Latin–American countries. Through an in-depth analysis of the case, we conceptualize the open generification process and its contingent mechanisms and suggest a general concept of open generification.

**GENERIFICATION – A LITERATURE REVIEW**

The starting point of the generification discourse within Science and Technology Studies is the observation that some software products, such as SAP and Microsoft Office, are successfully used across most of the world. Thus, they fit very different contexts. Pollock et al. (2003, 2007) noted that these products ‘travel’ successfully, to a great benefit for user organizations. The software is usually cheaper and technically more solid than locally developed solutions. In addition, the fact that such systems are widely used results in positive side effects such as a large base of supportive user communities, freelancers, consultants and other third-party agencies.

Discussion of IT design that works for multiple contexts is not new; it has been widely reported in the literature of large-scale global systems. A large part of this literature argues for the importance of articulating and balancing between global transportability and local adaptability (see, for example, Bjørn et al., 2009; Keen et al., 1982; Rolland & Monteiro, 2002). However, a recent theoretical construct, the concept of generification (Pollock et al., 2007), argues for more emphasis on global level practices. The premise of generification is to divert the attention away from local practices and organizational adaptations over to the origins and characteristics of globally generic software packages. It is based on observations of practices of global level software vendors and their strategies of producing generic software package that embodies characteristics common across a range of customers (Pollock et al., 2007). In generification work, vendors have a central role, as they are the ones responsible to make the technology work in a particular context as well as take it further globally to multiple other contexts (Grimm, 2009; Wang, 2007).

Technology working in a particular context is fixed in time and space (Berg, 1997). Transporting it to another context requires a complex work of disentanglement (Berg & Goorman, 1999). This calls for a process by which software packages are successively

© 2016 The Authors. Information Systems Journal Published by John Wiley & Sons Ltd., *Information Systems Journal* **27**, 619–642
emptied from references to local details to become standardized, generic solutions ready to travel globally’ (Kallinikos, 2009: 915). To shield generic software from local specification and focus on broader level standardized requirements, Pollock et al. (2007) identify three mechanisms. These are management by community, content and social authority. One key aspect in these mechanisms is a process where situated needs are dissociated from specific local needs and further negotiated towards a common set of requirements that serve a wider user group. We understand this as processes of disembedding. Giddens (1990) defines disembedding as ‘the lifting out of social relations from local contexts of interaction and their restructuring across indefinite spans of time and place’ (p. 21). While Giddens’s notion of disembedding is in the context of social relations, the same concept has been appropriated in the context of IS (Davison et al., 2008; Ignatiadis & Nandhakumar, 2007; Jin & Robey, 2008). Drawing from this, we see disembedding as the process of lifting out local software requirements out of their contexts and abstracting them to serve diverse user needs across space and time.

Although Pollock et al. have emphasized global practices, it is through implementation, configuration, customization and other appropriation mechanisms that the dissociated software package comes to work in local settings (Hong & Kim, 2002; Soffer et al., 2003). Thus, equally important is the process of (re)embedding the software in the local context that involves pinning down the disembedded system back to situated realities. Figure 1 presents our conceptual model of genericification.

In the management by community mechanism of genericification, vendors move design ‘from the private domain of each user site, where only particular needs could be articulated, to a public setting, where community or generic requirements could be forged’ (Pollock et al., 2007: 263). Clearly, a broader input channel brings a potential burden to vendors, as they could be flooded with numerous and diverse sets of requests. Management by content is aimed to overcome this, in particular its practice of crafting common sets of requirements by using configurable templates. Those requirements not compatible across sites are ‘rejected or sifted from the process’ (p. 265). In this process, the user base is homogenized, as some customers have no choice but to accept common requirements in spite of their specific needs. However, pushing organizations to fit into standardized software packages is no easy task; it has led to many misfit failures (Hong & Kim, 2002; Rajapakse & Seddon, 2005; Swan et al., 1999).

In the third mechanism, management by social authority, customers are ranked and treated according to their strategic value for the software vendor. The vendors focus on the needs of only the high-value customers, while the rest are ‘pushed to the margins … where they [are] not consulted or involved in design’ (Pollock et al., 2007, p. 269). This is a strategic approach in a vendor–customer relationship where the vendor forefronts business values and maximizing profits.

In IT governance terms¹ (Weill & Ross, 2004; ISACA, 2012), the decision rights are definitely centralized, although the mechanisms are relatively soft, and open to input from various actors.

¹IT governance is defined by Weill & Ross (2004) as the decision rights and accountability framework to encourage desirable behaviour in the use of IT.
The dominant vendor controls the policies of generification, the software architecture and the process of final product development.

With a critical perspective, Walsham (2008) has questioned these different approaches to generification: ‘which users get the most say in the generification process [...] how do power relations influence the design and make the system more suitable for some than others [...] what are the consequences [...] positive or negative’ (p. 17).

To address such concerns of openness and inclusion, the literature of FOSS offers various forms of governance mechanisms ranging from the top-down-centralized cathedral model to the bottom-up-distributed bazaar model (Raymond, 1999; Capiluppi & Michlmayr, 2007). In the cathedral model, participation is restricted to an exclusive group. This is a formalized practice that is typically seen with proprietary software products. In contrast, the bazaar model represents system development conducted online and as a public and transparent process. Linus Torvalds and his Linux operating system kernel project are credited for this governance model (Raymond, 1999). It is based on a democratic mechanism where the public is allowed to participate and establish authority and leadership through their contributions and under meritocratic
norms. Those showing an understanding of the software code as common good by proving outstanding technical competence, frequently providing feedback to user needs and undertaking bug fixes, are rewarded with positional authority (O’Mahony & Ferraro, 2007). The bazaar approach is a more informal and bottom-up-driven innovation model, potentially challenging traditional practices and established products such as Microsoft Windows operating system and enterprise resource planning (ERP) tools.

User-level innovation literature argues against the strategy of segmentation for serving diverse users’ needs as it ‘leaves many seriously dissatisfied’ (Franke & von Hippel, 2003 p. 1119). It is extremely difficult for a single vendor to cater for diverse requests coming from all corners of use. Technology innovation is increasingly becoming a distributed activity that takes place in networks and ecosystems (Boland et al., 2007; Selander et al., 2010; Yoo et al., 2008; Chesbrough, 2003; Enkel et al., 2009). This open innovation allows forging collaboration and integration with other vendors and innovators for better generification outcome that serves diverse user needs (Johannessen & Ellingsen, 2009). Franke & von Hippel (2003) argue that it is more effective to serve diverse user needs based on local innovations and governance mechanisms compared with doing it globally from the vendor side. Arguing against the practice of marginalization, many IS researchers further recommend inclusive strategies such as collaboration and user participation (Lamb & Kling, 2003; Braa et al., 2004; Puri, 2007; Titlestad et al., 2009). In a recent paper, Silsand & Ellingsen (2014) investigated an eHealth project and proposed an approach called generification by translation, which allows for a more symmetric relationship between vendors and the users than the cases permits of Pollock et al. Generification by translation includes three principles that a vendor needs to follow – being flexible to adjust design strategy during development; increase users’ knowledge of software design; and include professional users into the design of the generic software product.

In this paper, we build on and extend these insights. Our challenge is to conceptualize and test a generification process that is open and collaborative on the one hand (von Hippel, 2001; Silsand & Ellingsen, 2014) and on the other produces a generic software package that serves diverse needs in different contexts (Pollock et al., 2007). We did this through a longitudinal study of a large action research project.

Research Approach

Our empirical data are based on the design practices of the HISP. HISP is a global action research project that has been running since 1994 (Braa et al., 2004). Key activities in HISP are design, development and implementation of a software package called DHIS2. DHIS2 is a software for collecting, processing and visualizing healthcare data. Currently, it is in use by more than 47 countries as a national system and also by more than 15 international organizations such as United Nations Children’s Fund, World Health Organization, President’s Emergency Plan for Aids Relief and Centers for Disease Control and Prevention on project basis. It is developed and maintained by a team of global developers based at the University of Oslo, Norway and multiple local teams across countries such as India, Vietnam, Tajikistan, Kenya, Nigeria and Ghana to mention a few. The system works in the official language of each implementing country.
and runs on different platform and infrastructure that a country or implementing institution can afford. It works both offline and online, through local installation and cloud deployment and also on Windows, Linux or Mac platforms. Figure 2 presents a typical DHIS2 start-up screen.

Considering this case a fruitful ground to investigate our research question, we conducted an in-depth case study (Gerring, 2007) inside HISP for over a period of 7 years from 2007 to 2013. Following the biography of DHIS2 as a software artefact over several years is crucial to understand generification processes (Williams & Pollock, 2009). In addition, one of the authors has the experience of implementing DHIS2 locally at one point in time and later joining the global software development team. This has provided us an invaluable insider perspective, enriching our understanding of the design processes and practices related to DHIS2. With a focus on global and local design processes, we followed the case from three different angles. First, we followed the activities of the global designers who have the overall responsibility of designing, coordinating, maintaining and releasing the global DHIS2 software core. Second, we followed the activities of local level designers working with the DHIS2 core to make it fit to local requirements and situated use-cases. Complementing these two, we followed the interaction between global and local level designers.

Data collection

For data collection, we used interviews, participant observation and document analysis. We interviewed five designers: two from Oslo, two from India and one from Tajikistan. For the interviews, we used face-to-face conversations supported by an email questionnaire. We relied on interview notes and email archives to capture the data from the interviews. The second technique we used for data collection was participant observation where we actively participated

Figure 2. District Health Information software version 2 (DHIS2) start-up screen.
in design workshops and implementation seminars. In addition, one of the authors participated in DHIS2 design both as a part of the global team in Oslo and at local levels in India, Ethiopia, Tajikistan and Vietnam. The same author also participated in Skype conference calls that had DHIS2 design agendas. In our document analysis, we used a significant volume of data from design blueprints, training manuals and electronic archives from the HISP website, DHIS2 designers and users’ mailing list and Google Docs that are frequently used as a medium to share design documents and blueprints among designers. Table 1 provides a summary of the data collection and methods used.

### Data analysis

For data analysis, we used a general inductive approach (Thomas, 2006), which helped us to structure our analysis in three broad steps. These were the following: condensing our raw data into brief summary formats; developing relationships between our research objectives and summary findings; and finally, formulating a conceptual framework that captures the phenomenon we observed.

We started the first step of our data analysis by mapping out the case material as data displays (Miles & Huberman, 1994). We then moved to condense the raw data into a brief summary organized around key themes shown in Table 2 in the Analysis section. We organized the key events based on design processes that either created new software modules or changed existing ones. We also took the inception of DHIS2 as a key event as it has set the roadmap

| Data collection | Activities |
|-----------------|------------|
| Interview       | Face-to-face interview and email questionnaire of global and local level DHIS2 designers located in Oslo, India and Tajikistan. Designers each from Oslo and India, and one from Tajikistan |
| Participant observation | Diary notes, in some cases presentation slides, were collected from the following activities: Participation in seven seminars/workshops/trainings session (3 in Oslo, 2 in India, 1 in Vietnam and 1 in Tajikistan) Participation in numerous DHIS2 designers and implementers Skype conference calls with participants from Oslo, India, Vietnam, Tajikistan, Tanzania, Malawi, Sierra Leone and Zambia Participation in group and private email conversations regarding design, implementation and use of DHIS2 Participation in meetings regarding DHIS2 modules for recording, tracking and reporting, as well as support for mobile phones Participation in activities of designing recording, tracking and reporting modules, as well as support for mobile phones |
| Documents       | Design blue prints and data models Requirement specifications Training manuals Guidelines for developers, implementers and users Electronic archives (emails, chats, DHIS2 country databases, Google Docs and Launchpad) |

DHIS2, District Health Information software version 2.

© 2016 The Authors. Information Systems Journal Published by John Wiley & Sons Ltd., Information Systems Journal 27, 619–642
of the overall design process. Leaving the details aside and taking a broader view, we observed one, ‘generic’, DHIS2 being used in many countries. This convinced us to focus our analysis around the design process of DHIS2 by asking how the designers managed to keep one generic core. This helped us to move into the second stage of our analysis by making a connection between the empirical data and our research objective – which is investigating generification mechanisms that address the diverse needs of a user community. While our data concern a generic system for the public healthcare domain of developing countries, we believe that the analysis and theoretical contribution are also relevant elsewhere.

In the second step of our analysis, we identified central themes from each of the key events that contributed to the generification process. We facilitated this through data reduction by
repeatedly asking ourselves ‘how do all the codes and themes relate to each other? What is the big picture, and how does it relate to each theme or code?’ (Miles & Huberman, 1994: 69). This bottom-up coding procedure helped us to conceptualize the unfolding generification process. The concept of generification and its associated mechanisms from the literature provided us a lens to further analyse, code and abstract concepts as they became apparent. Table 2 in the Analysis section was the output from the second step of our analysis. Finally, we developed a conceptual framework that captures the underlying generification processes as observed in the empirical data. This is presented in Figure 3.

**CASE DESCRIPTION**

We now present the design process of DHIS2. Our presentation highlights four key events. To put things in context, we start by presenting the early history of DHIS2.

**Shift to generic, distributed and collaborative design**

Prior to DHIS2, HISP had an older version that was designed in 1998 by a small team in South Africa for the Western Cape Province (Braa & Hedberg, 2002). This system ran as a standalone application on top of a proprietary Microsoft platform. The system was expanded to Eastern

![Figure 3. Conceptual model of open generification.](wileyonlinelibrary.com)
Cape provinces in 1998, and it became a *de facto* reporting system for the entire country in 2000 (Braa & Sahay, 2012).

Based on the achievements in South Africa, HISP initiated implementation of the same system in Cuba, Mozambique, Malawi and India. The project in Cuba failed because the software was based on US-based Microsoft technology (Braa & Sahay, 2012). Mozambique and India on their side requested a Web-based system free of proprietary technology. Based on these experiences and a growing international user base, a team of researchers and students at the University of Oslo decided to develop a more robust and distributed design approach. This resulted in the launch of DHIS2 as Web-based FOSS in 2005. DHIS2 is based on a layered architecture, platform-independent Java frameworks and a specific programming strategy to support multiple languages. It also follows an open and distributed development strategy using internet-based code synchronization and coordination tools.

In the software documentation, DHIS2 is described as follows:

“… a generic tool rather than a pre-configured database application, with an open meta-data model and a flexible user interface that allows the user to design the contents of a specific information system without the need for programming” (http://www.dhis2.org, accessed April 2015)

Claimed to be generic, DHIS2 comes with no specific content when released globally. A member from a local HISP team in Zambia explains this as follows:

“… from an implementation standpoint … when new users attempt to use DHIS2, it is blank. There is nothing there. We have tried to fill this gap a bit with by answering questions over the public mailing list and also through the documentation…” (September 2011, Local HISP Member from Zambia)

Through mailing list, users exchange best practices, ask and respond to questions and request new features. For new features, users are encouraged to create blueprints documenting their requirements and detailed use-cases on Launchpad.2 A blueprint on launched receives public scrutiny and becomes part of DHIS2 development roadmap following approval. The approval is made by developers. However, if a blueprint comes from experienced users, it often goes straight into development roadmap. Experienced users have also an opportunity to assign blueprints to individual developers whom they think are experts on the requested feature.

Users are also provided a publicly accessible demo site where they can do testing and investigate available and missing features. In addition, users are encouraged to make code level changes if that is the only way to satisfy their needs. The following email exchange demonstrates this:

“Hello there, Is there a simple way to add sections or pagination to a custom form? The goal is to avoid scroll when dealing with long forms” (Local HISP Member from Mozambique, April 2013)

2Launchpad is a publicly available FOSS collaboration platform. It provides free hosting service. DHIS2 is hosted at https://launchpad.net/dhis2.
A member from the global HISP team replied:

“Hi, there is way for developers. In custom form editor, view the source. Wrap the HTML contents that you want to appear as a section in a div and add a class “formSection” to it becomes something like: <div class = “formSection”> … </div>. You should now be able to use sections…” (Global HISP Member from Oslo, April 2013)

There are also discussions related to generic features of the design. Later, we describe three such events.

Upgrading the specific to a generic

When DHIS2 was introduced in Ethiopia in 2007, the project was faced with a technology-context mismatch. DHIS2 was designed to handle single data values per data element, for example, the total number of people with malaria in a district. But in Ethiopia, reporting is done with multiple dimensions, for example, the total number of people with malaria is broken up by gender (male/female) and age (under the age of 4 years, between the age of 5 and 14 years etc.). Trying to overcome this mismatch, a local HISP team started to design for multidimensional reporting. In the process, the local team introduced major changes to the DHIS2 software core by embedding the age and gender breakups. When communicating their work to the global team, they received a mixed response. A coordinator from the global team sent an email saying: ‘… you’ve done a great job … really heartening to know that we can now handle [multidimensional]’. However, the lead designer from the global team replied:

“This works fine, but not for countries which don’t need the [multidimensional] functionality … [multidimensional] system should take care of its data separate from DHIS2 … By doing it this way, the core of DHIS2 wouldn’t have to be changed at all… also countries not needing the [multidimensional] functionality would not be affected …. the implications of this … is you extract all your code from the DHIS2 core modules and make a separate project.” (Global HISP Member from Oslo, February 2007)

The local team acknowledged the changes made were relevant only to Ethiopia but argued against the idea of not changing the core as it had limitations for multidimensional reports. Later, the global team agreed to change the core data model by using abstract Category and Option objects (replacing the age and gender additions from Ethiopia) based on the assumption that they could be used for any dimensions and disaggregation units.

While working on the changed multidimensional data model, HISP got the opportunity to support the implementation of DHIS2 in Tajikistan in November 2007. The Ministry of Health of Tajikistan came with complex multidimensional paper forms and ordered the implementation team to computerize them in the exact same layout:

“…since these forms are approved by the Central Statistics Authority, you are not allowed to make any change to the forms, even adding a logo.” (Tajikistan Ministry of Health, January 2008)

© 2016 The Authors. Information Systems Journal Published by John Wiley & Sons Ltd., Information Systems Journal 27, 619–642
After 2 months of work in Tajikistan, the team with members from HISP in Oslo completed the design supporting the reports asked by the Ministry. The new design offered menus allowing local teams to design their own dimensions and units using the Category and Option objects. While in Tajikistan, the global team also formed a local HISP team and provided trainings and guidance enabling the Tajiks to design their own data collection, reporting, analysis and presentation formats. Local participants played a key role in translating DHIS2 to the Tajik and the Russian languages.

Replacing the specific with a generic

In the end of 2006, a health commissioner from a state of India requested HISP to provide a dashboard module where he could pull important data of his choice, convert them to indicators, perform analysis and display the result in various formats. The global team was reluctant to develop what they understood as a too advanced feature. With this response from the global team and increasing pressure from the commissioner, the local HISP India team initiated their own design process.

As the leader of the Indian local team explained, they approached the design ‘as an independent module using only the DHIS2 database without using the core’. They justified this by arguing that they only needed to pull existing data and process, analyse and display it (i.e. no data input). With their approach, they quickly managed to satisfy the commissioner’s request. However, later, they found it problematic to maintain two separate links, one for their dashboard and the other for the core DHIS2, and started merging the two at the source code level. During merging, they created a local branch from the global DHIS2 trunk. This approach, they argued, provided them greater flexibility and independence from the slowly moving global trunk. In the process, they improved their dashboard with additional features for supporting integration with Excel reports and other key features that made the dashboard very popular in India. Later, global members started to investigate the possibility for merging:

“… what is the status of this module now? Is it ready to become an integrated DHIS2 module?” (Global HISP Member from Oslo, October 2007)

The Indian team saw the same potential and started the integration work. However, when testing the integrated solution outside India, it ended with errors. From the error report, global developers traced the problem to the dashboard module. They then recommended removing it from the global trunk until the problems were rectified. However, rectifying became a problem leading to a friction between the teams – the global team deeming the efforts of the local team as ‘a poor workaround and hack of the DHIS2 database’ and the local team accusing the global team for not understanding the pressure and immediate need they had to deal with. With the two branches kept apart, both teams went their ways in their design roadmaps – the global team improving the DHIS2 in terms of performance and new functionalities and the Indians improving the dashboard in their own setting based on demands from local Indian states. This situation quickly became a concern for some members of the global team as it forced them to maintain two branches, each missing out important changes and new functionality from the
other. Later, when one local HISP member from Vietnam attempted to combine the two, he also ended with errors and reported this and asked for help. At this point in time, the lead global designer decided to design a new module that was supposed to replace all the features from the dashboard module.

Following the announcement of the coming of a new module, another global designer left Oslo to India in order to sit together with the Indians to find possible ways of rectifying the problems. When evaluating the source code, it was found unnecessarily complex, difficult to maintain, filled with MySQL database code pieces and specific Excel reports. When this was communicated through the public mailing list, discussions of replacing the local module with the upcoming global module emerged. However, the head of HISP India responded by saying:

“[we would] like to stay with the Excel reporting module and dashboard as they are … it has been rolled out and training has been given to hundreds of people on that” (Head of Local HISP India Team, January 2009)

In the end, the Indians stayed with their local module, and the global designers completed a generic solution that has proved valuable for many other countries.

**Overhauling the specifics to a generic**

In addition to computer-based data processing and analysis, HISP has also embarked in utilizing the potential of mobile phones. In 2009, the National Rural Health Mission Director of India requested the local HISP team to extend DHIS2 to support mobile phone-based reporting. To meet the request, the local team designed an SMS-based mobile application for processing a monthly report. The application was piloted in five health administrative blocks in five different states. Following the pilot, the state of Punjab asked for a state-wide implementation for three different reports, using 5000 mobile phones. The local team responded with three different applications.

With state-wide implementation activities in Punjab, the global HISP team received a request to implement mobile-based reporting in Gambia and Tanzania. The global team delegated the task to the local Indian team as they had the experience. In parallel, an HISP member from Zambia also created his own application following the approach from India. As a result, seven separate DHIS2 mobile applications were developed. With mobile phone usage increasing in many countries, the global team anticipated numerous requests coming from countries and found maintaining separate designs for each country and report as unsustainable. The team wanted a single mobile application that caters for all reports. In a meeting to consolidate the different applications, one global designer suggested delegating the task of managing reports to local teams. He said:

“…we shouldn’t meddle with reports of local users at the code level; we just need to leave that for the users. All we need to do is provide a [mobile] platform that allows them to deal with their own reports.” (Global HISP Member from Oslo, June 2010)

The global team then developed new mobile reporting system based on client/server architecture using the DHIS2 as back-end server and mobile phones as front-end clients. From
the server side, the new design provided functionality for users to design reports in the language of their choice using the existing DHIS2 menus. For mobile phones, the solution provided menus for downloading reports from the server. Local users have the option to complete and send reports back to the server, including the opportunity to update, remove, and add reports, which was not possible in the earlier design.

ANALYSIS

In this section, we analyse the case of DHIS2 with a particular focus on the interplay between global and local level activities. We start the section by first summarizing our findings related to generification (Table 2) and discussing their implications.

Generification of District Health Information software version 2

District Health Information software version 2 was launched when the designers shifted from a traditional and centralized system development practice to open and distributed FOSS development. FOSS is a strategy for building and deploying large-scale software systems on a global basis as it facilitates the functioning of a globally distributed community of users and developers (Scacchi et al., 2006). The shift to a Web-based approach is another design strategy that contributed to the generification of DHIS2. The Web is in itself a cross-platform technology that is accessible from anywhere. The choice of Java, a programming language called ‘write once, run everywhere’ (Kramer, 1996), further contributed to the cross-platform nature of DHIS2. The internationalization and localization programming strategy to accommodate local languages was another factor contributing to the generic nature of DHIS2.

Framed as FOSS, Web-based, platform-independent and multi-language, the DHIS2 design was based on an abstract meta-data model providing menus and other resources open for further design. By being abstract and not tied to a specific context, DHIS2 obliges users to embed the software by configuring reporting units, periods, dimensions and other data definitions. In addition, the open source code allows local actors to go beyond traditional forms of configurations and perform actual design. By offering this opportunity, the global team has effectively opened up the design space for external actors. They have the permissions and the platform to innovate and to serve specific local needs.

Even though DHIS2 was able to fulfil the needs of multiple countries, the design process was not a smooth ride. It was also neither centralized nor top-down from the global to the local. Rather, it was an evolving process, where the initiative and the innovation process went back and forth between the global and the local. To make DHIS2 generic and transportable to multiple countries, global designers had to disembed local innovations from their local specificities. The link between DHIS2 innovation and local specificities was challenging for the global team. For example, while they found it encouraging that the multidimensional case was solved locally, they were frustrated with the way it was solved. As evident in both the multidimensional and mobile cases, the global team did not reject local innovations as such; what was rejected was the way the modules were designed. In both cases, global designers went through discussions with
local developers and brought better ways of incorporating the needed functionalities without compromising the generic aspects of DHIS2.

District Health Information software version 2’s design process also demonstrates that organizations, being inherently dynamic and emergent (Truex et al., 1999), require software that allows for various reconfigurations and adjustments. This was, for example, apparent in the mobile reporting case. During the initial phase, every time a new report was designed or an existing one was changed, a new solution had to be designed. To meet this sustainability and scaling challenge, the global team came up with a generic solution by conducting an architectural innovation – the client/server approach – which allowed users to design and specify reports depending on their needs. These cases demonstrate the importance of providing mechanisms and allowing local teams to perform situated design without affecting the ability to meet future needs.

Unlike the generification strategy of, for example, ERP systems where local actors are not a part of the design process and their requirements at risk of being curbed, DHIS2 appears as more open and collaborative. In the generification of DHIS2, users were not only customers but also designers who were encouraged to make changes to the source code if that was the best way to address their needs. The move from a centralized design practice to a distributed FOSS strategy also helped to establish collaboration between designers. Local designers, from Ethiopia, India and Zambia, were all able to innovate based on the DHIS2 source code and establish a local solution that worked for their contexts. Although the local solutions were not accepted as a global solution right away, they were operational locally. Over time, the local innovations also became a part of the generic DHIS2 core. This has also helped HISP to slowly build collective innovation capacity by utilizing and nurturing the expertise of global and local level designers in concert. In many ways, it is this collective capacity that has enabled HISP to design a software system relevant across more than 47 countries.

In summary, the case of DHIS2 illustrates a design process serving the needs of multiple countries. The key aspect of this generification process is its nature of being open and collaborative, through strategies of platform independence, multi-language, FOSS, meta-design, configuration of designers at global and local levels and processes of embedding and disembedding. This approach to generification made it possible for the global team to engage local teams and an opportunity for the local teams to design for their needs. This was a win–win situation – the global team was not carrying the whole burden of designing for all and local teams not left alone to develop their software. We believe that this is a unique practical example to learn from and approach to address challenges of generic technology design that serve diverse user needs.

**OPEN GENERIFICATION: A FRAMEWORK**

In this section, we theorize DHIS2’s design process with a particular focus on the configuration of social and technical elements at global and local levels. By conceptualizing this process as *open generification*, we return to the concerns expressed earlier in this paper regarding
generification primarily as a top-down process based on customer management through segmentation and market mechanisms.

The premise of open generification is not about managing a community of users attached to generic software; rather, it is a design strategy for developing software packages that serves diverse user needs. The key components of our concept of open generification are global and local developers and processes of embedding and disembedding. Figure 2 presents a conceptual model of the open generification process. Next, we elaborate on these key components.

The global developers often aim to stay away from directly designing for particular local needs. They strive for a broad perspective and design a globally relevant software package. This is different from the problem-closure approach of traditional system design practices where the intention is to directly tick-off specific user requirements. Open generification is about establishing the necessary resources – including a generic software core, design guidelines, training, communication platforms and other capacity building mechanisms – that enable locally situated developers to perform local innovations. The software package that comes from global developers is devoid of local details. However, for the package to work at specific user sites, it is crucial that its components match with local specificities, work practices, organizational structures and other socio-technical arrangements. It is therefore the task of local developers to ensure that the software is modified to satisfy local needs.

A valid strategy to guide local developers is traditional system design practices. They address specific user requirements to reach the stage of problem closure. However, a key difference with open generification is that local developers are not starting their work from scratch but working with an incomplete meta-solution. It is their task to customize and extend the generic package to meet the specific local requirements.

A global software package is only relevant after a process of embedding it in a local setting. We have identified two types of embedding processes in open generification. The first is for appropriating or pinning down the ‘empty’ global software in local settings. This involves filling out or configuring the missing parts left intentionally open by global developers. The second type of embedding is at a deeper level. It involves local innovation in order to make the generic software work for specific needs that were not anticipated by the global developers. It is possible because the initial design practices followed by global developers offer a design space for local developers.

The practice of embedding that leads to pure local level innovation is one of the key aspects of open generification. It allows for innovation outside the realm of a central actor. However, the local innovation becomes even more fruitful if it can be transferred to other settings. Disembedding is one such mechanism that open generification offers to facilitate this. One form of disembedding is where incoming user requirements are translated into common requirements through abstraction and negotiation and later implemented as parts of the global core. A local innovation that occurs during embedding is a different kind of opportunity to further improve a generic software package. It is the starting point for a bottom-up generification process that feeds back from the local to the global. Usually, however, it is not possible to take the local innovation ‘as is’ to the global level, as it will be tightly coupled with local realities, both in terms of requirements and design practices. Hence, local and global developers need to undertake
the complex work of disembedding the software through material restructuring, architectural innovation and negotiations. Sometimes, it might be difficult to disembed software from its local setting and make it part of the global core. In some instances, global developers will only reuse the concept and start from scratch with the software coding.

DISCUSSION

The strategies of conventional and open genericification share the overall aim, namely, to support a global and diverse user community. The means they deploy in order to reach their aim differs in many respects.

Extending conventional genericification

First, the genericification process is different. A distinguishing aspect of open genericification is the qualifying term ‘open’ that refers to openness in actors and contents of genericification. With openness in terms of actors implies for a process that opens participation for external stakeholders. This requires establishing coordination platforms and mechanisms for effective collaboration towards the generic product. With the contemporary context of globalization and recent advancement of technology, there is a considerable degree of innovation happening outside the realm of a single vendor. It is therefore important to tap into this reality, and genericification strategies need to take it into account. Openness in terms of content implies that it is not only selected requirements picked by a vendor but also innovation outputs that comprise the feedback from the local to the global.

With conventional genericification, the vendor and their central developers decide what goes into the generic software. This is done through a set of management mechanisms — summarized in the model as disembedding. In the course of disembedding, the supplier aims to establish a standardized set of requirements either by curbing, sifting and sometimes rejecting diversity or by answering the needs of only those considered surrogates that others would follow. Once a common requirement is established, what follows is a matter of standard software development practice towards a generic software package where design process, content and output are dictated by the vendor. Little is discussed on what is happening at local levels when the generic software meets use. Pollock et al. clearly describe their genericification perspective as turning the focus away from local processes to what is happening at the global vendors. Open genericification brings local practices back to the spotlight.

However, we see local level adaption practices different from processes of genericification. While adaptation is focused on making a software system work in particular local context, genericification goes beyond that. It also aims to take the ‘locally worked’ software to multiple other contexts through the process of disembedding. This implies open genericification involves both adaptation and disembedding. In addition, the adaptation process in genericification, we argue, is different from the ‘conventional adaptation’, where the aim is to make the software just work in the context under consideration. Adaptation in genericification also keeps an eye on the possibility of taking the solution to contexts other than the one being worked for. This
potentially calls for a different adaptation mechanism that lends for easier disembedding, restructuring and reembedding.

Looking to local practices in the course of generification, we argue, is crucial for two key reasons. First, it draws on the well-established contextual literature that the IS community has built over decades. This helps to address some of the concerns persistently attributed to IS failures in organizations – such as the challenges created by misfits and design-reality gaps that arise from systems ‘designed from nowhere’ (Suchman, 2002). The second point is the opportunity it brings to the perspective of local studies. Although important, the ultimate goal of open generification is not to make a software work in a particular context; it is rather to take the working solution further to multiple other contexts. As a result, it pushes researchers to focus on studies that transcend time and place and look for connections and similarities across organizations (Leonardi & Barley, 2008). It is only then that we advance our contextualist’s perspectives and account for standardized systems (such as ERP, Microsoft Office and other packaged software products) seen in multiple organizations.

Second, the roles of developers and users are different. Open generification allows local developers to exercise their agency and perform situated innovation. This is in stark contrast to the management of Pollock et al. by content strategy where users are ‘policing’ from making changes and convinced to fit into the common software package the vendor has designed. Convincing customers to fit into standardized software package is hardly a case of serving diverse user needs, but rather a case of implementation and organizational transformation for the purpose of matching the standard package. In open generification, it is the software package that is transformed and redesigned to serve the needs of multiple users. Open generification leaves the design open and encourages users to exercise their agency and think outside the box. Practices of meta-design, generative, participatory and FOSS designs are some mechanisms facilitating this (Avital & Te’eni, 2009; Fischer & Giaccardi, 2005; Fischer et al., 2004; McCormack et al., 2004). It is extremely difficult for a remote vendor to manage what goes into the needs of each and every customer. Local developers are well positioned to support this. At the same time, the local developers do not have to be an end-to-end expert and build everything from scratch – global level designers meet them on the half-way or so in addressing this.

Third, it supports user-centred innovation. While the existing generification research treats users and organizations mainly as consumers of technology, open generification considers them as a prime source of innovation. Local actors not only consume technology or provide requirement but also innovate by extending generic packages with new functionalities that fit into their situated needs. Recognizing users as innovators creates an opportunity to develop a collective innovative capacity that creates a more open, networked and what we believe is a more sustainable solution.

Information technology governance and power

The three characterizing attributes of open generification discussed previously are not only different from the generification process described by Pollock et al. (2007) but also imply an alternative governance regime. Current IT governance (Weill & Ross, 2004; ISACA, 2012) tends to
be centralized and tightly controlled, because of the need for coordination and scaling. Open
generification questions this approach and suggests a governance regime that is more bal-
anced; it reduces the power of the central actor and increases the influence of local initiatives.
In this sense, the open generification process is characterized by negotiation rather than
manipulation.

Unlike an exclusive form of power and control, open generification suggests for a transparent
and evolving approach where dispersed actors are invited in to participate and rewarded for
their contributions towards the common goal. Through various online tools and mechanisms,
why a particular developer has more power than others is made transparent. Mailing list ex-
changes, question and answer sessions, blog sites and amount and contrition of code submis-
sions document the power play. This power is also temporary, as it can be eroded by the efforts
of other contributors and thus depending on emerging merits.

There are certainly limitations to the transparency and openness of the approach. For exam-
ple, certain developers can establish themselves as central contributors, defining the roadmap
and controlling its implementation and by this effectively blocking others’ attempts to influence
it. This may be the result of certain developers being considered as the ‘founder’ of the soft-
ware or because they have managed to establish a stable source of funding for their contribu-
tion to the project. And while the roadmap may be defined in a democratic fashion where all
suggestions are treated equally, the focus of the individual contributors may be under the
control of external funding agencies. This however is not to emphasize the power of external
actors or those of the ‘founders’. Being FOSS, there is always an opportunity for those not
happy with the influence and control of a particular actor to fork out and create a parallel pro-
cess. This creates a necessary tension convincing the community to remain open, collaborate
and stay together.

**Future research: is open generification feasible outside the FOSS community?**

Our empirical evidence does not provide answers to this question, but we will briefly point to
three possible avenues of further research in this respect. First, the examples of Apple and
Google have shown that a platform strategy enables the division of labour between the de-
signers of the core of the platform and its third-party developers (Baldwin & Woodard, 2008).
While the technical and economic aspects of this development are well researched
(Ghazawneh & Henfridsson, 2013), the design aspects are not sufficiently investigated, and
the concept of open generification opens for a more balanced design strategy between the
central team and local developers. We can envision a situation where the platform is a shared
resource for a user community rather than commercial a vehicle for a platform owner.

Second, the research on open innovation (Chesbrough, 2003; Enkel et al., 2009) has shown
the potential of cooperation on equal terms. We see contractual agreements both at organiza-
tional and individual levels, cases of ‘feeder-organizations’ or ‘innovation hubs’, and also prac-
tices of securing intellectual property rights as some potential areas to investigate collaboration
and coordination mechanisms for open generification.

Third, the governance regime of open generification, as described in this paper, balances
successfully between centralized control and local innovation. As information infrastructure
research (Hanseth & Lyttinen, 2010; Henfridsson & Bygstad, 2013) has shown, this balance is crucial for continuous innovation in heterogeneous networks. Therefore, the concept of open generification allows for more research on the interplay of generification and innovation or, put differently, how innovations based on local experiences can be integrated into larger structures.

CONCLUSION

We started our study by asking how can we conceptualize a process of open generification and what are the contingent mechanisms for successful use of this process? We investigated these questions through a longitudinal case study of a large, international HISP operating in more than 47 developing countries. Based on the analysis of the case, we have suggested a particular concept of open generification.

Open generification denotes a process of continuous interplay between generic and specific software, based on the collaboration of global and local software designers, through the mechanisms of embedding and disembedding. The open generification concept is different from the extant generification approach in being open and collaborative. The emphasis is not on managing the community of users attached to the software, but in managing the diverse needs of the community the software is expected to serve and to facilitate and accommodate the outcomes of distributed processes of innovation. We have documented that this approach was effective, even if challenging, in a context of health systems in developing countries and as a part of FOSS design practices. Future research could investigate the merits of this strategy in other settings.

REFERENCES

Avgerou, C. (2002) Information Systems and Global Diversity. Oxford University Press, New York, USA.
Avital, M. & Te’eni, D. (2009) From generative fit to generative capacity: exploring an emerging dimension of information systems design and task performance. Information Systems Journal, 19, 345–367.
Baldwin, C. Y. & Woodard, C. J. (2008). The architecture of platforms: a unified view. Harvard Business School Finance Working Paper, (09-034).
Berg, M. & Goorman, E. (1999) The contextual nature of medical information. International Journal of Medical Informatics, 56, 51–60.
Berg, M. (1997) Rationalizing Medical Work: Decision-support Techniques and Medical Practices. MIT Press, Cambridge MA.
Bjørn, P., Burgoyne, S., Crompton, V., MacDonald, T., Pickering, B. & Munro, S. (2009) Boundary factors and contextual contingencies: configuring electronic templates for health care professionals. European Journal of Information Systems, 18, 428–441.
Boland, R.J., Lyttinen, K. & Yoo, Y. (2007) Wakes of innovation in project networks: the case of digital 3-D representations in architecture, engineering, and construction. Organization Science, 18, 631–647.
Braa, J. & Hedberg, C. (2002) The struggle for district-based health information systems in South Africa. The Information Society, 18, 113–127.
Braa, J. & Sahay, S. (2012) Integrated Health Information Architecture – Power to the Users. Matrix publishers, New Delhi.
Braa, J., Monteiro, E. & Sahay, S. (2004) Networks of action: sustainable health information systems across developing countries. MIS Quarterly, 28, 337–362.
Capiluppi, A. & Michlmayr, M. (2007) From the cathedral to the bazaar: an empirical study of the lifecycle of volunteer community projects, In: Open Source Development,
Adoption and Innovation, pp. 31–44. Springer US, Boston, USA.
Chen, Y. & Reich, B.H. (2007) IT alignment: what have we learned? Journal of Information Technology, 22, 297–315.
Chesbrough, H. (2003) Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business School Publishing, Boston, USA.
Ciborra, C., Braa, K. & Cordella, A. (2000) From Control to Drift: The Dynamics of Global Information Infrastructures. Oxford University Press, Oxford, UK.
Davenport, T. (1998) Putting the enterprise in the enterprise system. Harvard Business Review, 76, 121–131.
Davison, R., Kien, S.S. & Ying, D.X. (2008) Introduction to Open R&D and open innovation: exploring the phenomenon. R&D Management, 39, 311–316.
Fischer, G. & Giaccardi, E. (2005) Meta-design: a framework for the future of end user development. In: End User Development – Empowering People to Flexibly Employ Advanced Information and Communication Technology, Liberman, H., Paterno, F. & Wulf, V. (eds), pp. 427–457. Kluwer Academic Publishers, Dordrecht.
Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G. & Mehנדje، N. (2004) Meta-design: a manifesto for end-user development. Communications of the ACM, 47, 33–37.
Franke, N. & von Hippel, E. (2003) Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software. Research Policy, 32, 1199–1215.
Giddens, A. (1990) The Consequences of Modernity. Stanford University Press, Stanford CA.
Gerring, J. (2007) Case Study Research: Principles and Practices. Cambridge University Press, New York.
Ghazawneh, A. & Henfridsson, O. (2013) Balancing platform control and external contribution in third-party development: the boundary resources model. Information Systems Journal, 23, 173–192.
Grimm, C.F. (2009) Inside a secret software lab: an ethnographic study of a global software package producer. Unpublished doctoral thesis, University of Edinburgh, United Kingdom.
Hanseth, O. & Lytinen, K. (2010) Design theory for dynamic complexity in information infrastructures: the case of building Internet. Journal of Information Technology, 25, 1–19.
Hanseth, O., Monteiro, E. & Hatling, M. (1996) Developing information infrastructure: the tension between standardization and flexibility. Science, Technology and Human Values, 21, 407–426.
O’Mahony, S. & Ferraro, F. (2007) The emergence of governance in an open source community. *Academy of Management Journal*, 50, 1079–1106.

Pearson, K.E. & Saunders, C. (2009) *Strategic Management of Information Systems*. John Wiley & Sons, NJ, USA.

Pinch, T.J. & Bijker, W.E. (1984) The social construction of facts and artefacts: or how the sociology of science and the sociology of technology might benefit each other. *Social Studies of Science*, 14, 399–441.

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

Pollock, N., Williams, R. & Procter, R. (2003) Fitting standard software packages to non-standard organizations: the ‘biography’ of an enterprise-wide system. *Technology Analysis & Strategic Management*, 15, 317–332.

Puri, S.K. (2007) Integrating scientific with indigenous knowledge: constructing knowledge alliances for land management in India. *MIS Quarterly*, 3, 355–379.

Rajapakse, J. & Seddon, P.B. (2005). ERP adoption in developing countries in Asia: a cultural misfit. In Proceedings of the 28th Information Systems Research Seminar in Scandinavia, Kristiansand.

Raymond, E. (1999) The cathedral and the bazaar. *The Info-Technology*.

Suchman, L.A. (1987) *Plans and Situated Actions: The Problem of Human-machine Communication*. Cambridge university press, New York, USA.

Suchman, L.A. (2002) Practice-based design of information systems: notes from the hyperdeveloped world. *Information Society*, 18, 139–144.

Swan, J., Newell, S. & Robertson, M. (1999) The illusion of best practice in information systems for operations management. *European Journal of Information Systems*, 8, 284–293.

Thomas, D.R. (2006) A general inductive approach for analyzing qualitative evaluation data. *American Journal of Evaluation*, 27, 237–246.

Tittlestad, O.H., Staring, K. & Braa, J. (2009) Distributed development to enable user participation. *Scandinavian Journal of Information Systems*, 21, 27–50.

Truex, D.P., Baskerville, R. & Klein, H. (1999) Growing systems in emergent organizations. *Communications of the ACM*, 42, 117–123.

von Hippel, E. (2001) Innovation by user communities: learning from open-source software. *Sloan Management Rev*, 42, 82–86.

Walsham, G. (2001) *Making a World of Difference: IT in a Global Context*. Wiley, Chichester.

Walsham, G. (2005) Knowledge management systems: representation and communication in context. *Systems, Signs & Actions, 1*, 6–18.

Walsham, G. (2008) Information Technology in the Service Economy: Challenges and Possibilities for the 21st Century. In: *IFIP International Federation for Information Processing*, Barrett, M., Davidson, E., Middleton, C. & DeGross, J. (eds), Vol. 267, pp. 13–25. Springer, Boston.

Wang, M. (2007) Cultivating the ‘generic solution’ – the emergence of a Chinese product data management (PDM) software package. Unpublished doctoral thesis, University of Edinburgh, United Kingdom.

Weill, P. & Ross, J.W. (2004) *IT Governance: How Top Performers Manage IT Decision Rights for Superior Results*. Harvard Business Press, Boston, USA.

Williams, R. & Pollock, N. (2009) Beyond the ERP implementation study: a new approach to the study of packaged information systems: the biography of artefacts framework. In Proceedings of the 30th International Conference on Information Systems. Paper 6. Phoenix, Arizona.

Yoo, Y., Lyytinen, K. & Boland, R. J. (2008) Distributed innovation in classes of networks. In Hawaii International Conference on System Sciences, Proceedings of the 41st Annual (pp. 58–58). IEEE.

**Biographies**

**Abayot Asalefew Gizaw** holds PhD in Informatics from the University of Oslo, Department of Informatics. His study area is information systems innovation with special interest to scaling, integration, architecture and generification in the...
Abyot has received his MS and BS in Computer Engineering from Addis Ababa University, Ethiopia.

Bendik Bygstad is a sociologist who is currently a professor at the University of Oslo. His main research interests are IT-based service innovation and the relationship of IS and organizational change. He is also interested in IS research methods, in particular the philosophical and methodological implications of critical realism. He has published articles in MIS Quarterly, Information Systems Journal, Journal of Information Technology and Information & Organization.

Petter Nielsen holds a PhD in Informatics from the University of Oslo. Working for 10 years as a researcher and business developer in the international telecommunication industry, he is currently an associated professor at the University of Oslo. His research interests relate to the design and implementation of large-scale and complex information infrastructures, with a particular focus on architecture, control and innovation. His empirical research ranges from mobile content services and the mobile internet in Northern Europe and Asia to the emerging field of mHealth in developing countries. He has published articles in The Information Society, Journal of Information Technology, International Journal of IT Standards and Standardization Research and Information Technology for Development.