Scandinavian Journal of Information Systems

Volume 29, No. 2

Publication date: 31 December 2017

eISSN 1901-0990

P for Platform

Architectures of large-scale participatory design

Lars Kristian Roland, Terje Aksel Sanner and Johan Sæbø
Department of Informatics, University of Oslo
roland@ifi.uio.no; terjeasa@ifi.uio.no; johansa@ifi.uio.no

Eric Monteiro
Department of Computer and Information Sciences, Norwegian University of Science and Technology (NTNU)
Eric.Monteiro@idi.ntnu.no

Abstract. Participatory Design (PD) has traditionally been committed to extensive interaction between developers and situated users to mitigate the disempowering consequences of computerization, such as the deskilling of labour workers. However, the widespread adoption of off-the-shelf software and the emergence of complex information system architectures with interdependencies across user groups and organizations challenge the applicability of traditional custom PD. Pressure is put on PD to scale with initiatives that span an increasing number and distribution of heterogeneous settings, developers, users and uses over time. In this article, we follow a PD project that started out in post-apartheid South Africa more than two decades ago. The project, which centres on the development of a software product for decentralised public health care management, has since grown into a venture with a significant footprint in the Global South. In order to problematise the scaling of key aspects of PD, such as the politics of design, the nature of participation and participatory design techniques, we first review extant literature and develop a classification of four different types of PD with respect to scale. We then apply the typology to our empirical case to discuss PD in relation to architectural traits at different stages of project scale. We contribute to PD literature by addressing the exploratory research question: What role does architecture play in large scale PD? Specifically, the study highlights how an emergent platform architecture and its surrounding ecosystem co-constitute a platform for participation in design.

Keywords: participatory design (PD), healthcare management, architecture, platform

Accepting editor: Magnus Bergquist

1 Introduction

Two decades ago, Neumann and Leigh Star (1996) questioned to what extent Participatory Design (PD) could be applied in large-scale information system projects. Unfortunately, the question remains largely unanswered (e.g., Kyng 2010) despite its rapidly increasing pertinence. Today, PD researchers and practitioners increasingly find themselves in the midst of complex IT mega projects with many stakeholders that potentially dilute, derail or overrun small-scale participatory efforts (Karasti 2014). Yet most studies of PD still focus on the development of a single custom system that supports work within one particular client organization, department or group (Obendorf et al. 2009; Oostveen and Van den Besselaar 2004; Pilemalm and Timpka 2008). The conditions for small-scale politically motivated prototype-based approaches (Bødker and Pekkola 2010) are increasingly absent in contemporary software development processes in general (Kyng 2010; Shapiro 2005) and in health care in particular (Balka 2006; Pilemalm and Timpka 2008).

To remain relevant, PD needs to find ways to accommodate more distributed development settings (Gumm 2006; Obendorf et al. 2009; Titlestad et al. 2009) and interplay with complex and evolving ecosystems of ICT capabilities (Hess et al. 2008; Monteiro et al. 2012). We refer to this as the problem of scale in PD. In relation to PD, we define scale as the number and distribution of heterogeneous settings, developers, users and uses of a (software) product over time. As Karasti (2014) notes in a recent literature review, challenges to the sustainability of short-term PD projects have been explicitly problematised, while spatial scale, as we have defined it above, has received less attention. However, as our literature review in the following section highlights, contemporary PD projects encounter severe challenges with physical, temporal and organisational scale in relation to distribution of both designers and users (Gumm 2006).

To study PD in relation to scale, we draw on a longitudinal case study of a software development project that started out in support of decentralised health care management in post-apartheid South Africa more than two decades ago (see Braa and Hedberg 2002). Throughout the duration of the project, key individuals such as lead software developers and project managers from Norway, our colleagues, have explicitly sought to reflect on the applicability of PD (Braa and Hedberg 2002; Braa and Sahay 2012; Shaw and Braa 2014; Titlestad et al. 2009). The software has evolved from a custom-made desktop application to a modular web-based platform used globally. Consequently, the project has accumulated complexity along each dimension in our definition of PD scale: From a handful to tens of thousands of heterogeneous users, from a pilot in one district to distributed implementations in more than 50 countries, from clearly defined work routines of monthly public health data reporting and analysis to a diverse range of application domains.

To mitigate complexity and manage dependencies, information system architectures have been developed into modular and loosely coupled systems of systems that interact through standardised interfaces (Yoo et al. 2010). A prominent technical architecture employed to tackle complexity is the platform (Tiwana 2013), where reusable and generic functions are bundled as a platform core while specific services are developed as compliments, called apps. However, with the term architecture, we here refer not only to software layers and modules and the interfaces between them (Van Schewick 2012), but also to the structure of a corresponding socio-technical ecosystem (Tiwana 2013) that potentially both enables and constrains participation in design. Contemporary PD takes place in this new architectural environment, for which traditional small-scale participatory approaches do not necessarily fit well (Shapiro 2005). This challenge motivates us to ask:

What role does architecture play in large-scale PD?

The contribution of this study is twofold. First, in section two, we provide a narrative review of extant literature and develop a typology of PD with respect to scale. Second, we employ this topology to our empirical case to discuss the enabling and constraining role of architecture in relation to PD. Section three outlines our research approach based on longitudinal involvement with one large-scale PD project and our theoretically informed sampling of three case vignettes. The vignettes are presented in section four. Our discussion in section five highlights how the architecture of the software product and the ecosystem that surrounds it shapes the politics of design, the nature of participation and the use of PD tools and techniques. We offer concluding comments in section six.

2 Scaling participatory design

Traditional PD in information systems development, with roots in Scandinavia, was driven mainly by two key concerns: workplace democracy and usefulness of design (Bjerknes and Bratteteig 1995). Firstly, end-user involvement was an explicit democratic goal (Bjerknes et al. 1987; Clement and Van den Besselaar 1993; Sandberg 1979). It was argued that the workplace, where computerised tools came into play, should follow principles of democracy that applied elsewhere in society. Secondly, PD was also seen as a means to increase the quality, efficiency and usefulness of new IT-solutions (Kyng 2010). This was based on the pragmatic view that there is a great deal that users can contribute to design (Shapiro 2010). Early PD techniques included mock-ups, prototyping, workshops and scenario building (Ehn 1988). In the seminal Utopia project 1985-1987 at the newspaper Aftonbladet, the developers went through several rounds of prototyping to articulate the needs of the graphics workers for which the system was intended (Bødker et al. 1987; Ehn 1988). These PD techniques have remained essential to date, despite their apparent shortcomings in addressing some of the challenges to user involvement in contemporary software development processes characterised by the physical, temporal and organisational distribution of both designers and users (Gumm 2006; Obendorf et al. 2009). These broad dimensions align with our definition of PD project scale as the number and distribution of heterogeneous settings, developers, users and uses of a (software) product over time.

One apparent problem with multisite and multi-user PD is logistical: how can developers engage intimately with situated practices spread out across a large number of sites? The more fundamental problem arises when the supported practices also become increasingly heterogeneous (Obendorf et al. 2009). The current pressure to modernise PD was debated in an issue of the Scandinavian Journal of Information Systems. The geographical distribution of projects and the heterogeneity of users and uses were problematised as obstacles to PD (Balka 2010; Kyng

2010; Karasti 2010; Shapiro 2010;). New forms of PD are needed to ensure that diverse user interests can be represented and recognised in large-scale projects with powerful stakeholders, novel funding mechanisms, and complex inter-organisational information system dependencies (Kyng 2010; Oostveen and Van den Besselaar 2004). In the following subsection, we review literature concerned with approaches to overcome challenges to PD project scale and synthesise this into a typology.

2.1 Singular, serial, parallel and community PD

We use the term 'singular PD' to denote key principles and techniques associated with custom PD such as use of prototypes, mock-ups, and workshops to facilitate end-user participation in design (Clement and Van den Besselaar 1993; Ehn 1988; Kensing 1987). In short, singular PD emphasises the one-to-one relation between developers and situated users. In recognition of the shortcomings with the classical singular approach to PD in contemporary projects, we review literature concerned with PD at scale to discern three additional types, which we dub 'serial PD', 'parallel PD' and 'community PD'.

Titlestad et al. (2009) consider challenges in a distributed PD environment and describe a software product that gradually improves through the accumulation of local innovations and learnings. They highlight the role of software implementers and customisers who, over time and across organizational and geographical settings, mediate requirements to a global development process (Figure 1). We dub this sequential model of PD across sites and over time as serial PD, where situated design iterations build on previous PD in another setting through distillation of global standards. Through accumulation of experiences from distributed yet relatively similar settings, implementation teams gain capacity to configure the software in collaboration with users. In the words of Titlestad et al. (2009, p. 33), the gap between designers and users "can best be bridged by technically conversant people who engage in implementing the system and training end users, and who are also adept at communicating with the core developer team."
Traditional PD techniques are difficult to employ when co-location and homogenous work practices are no longer given. Obendorf et al. (2009) identify two techniques that facilitate PD across distributed sites. They refer to one technique as *inter-contextual workshops* that enable experience sharing, communication and requirements negotiation between dispersed users who have similar interests towards the same software product. The nature of user participation in such workshops takes the form of representation. Skilled implementers and local experts who understand the needs of different user groups professionalise participation. The second technique identified by Obendorf et al. (2009) is referred to as commented *case studies.* Case studies constitute descriptions of the appropriation and local customisation of a flexible software product to specific uses. Power users typically develop such case studies for circulation to the wider software community. On the downside, rich qualitative descriptions are time consuming to produce and may quickly become obsolete following software releases with new or revised functionality (ibid). Both of the identified PD techniques reflect a transition from situated design to representation of distributed and heterogeneous sites, users and uses.

Hansson et al. (2006) report on a distributed software development effort that spans multiple end-user groups and settings. They note that due to the relatively small size of the software

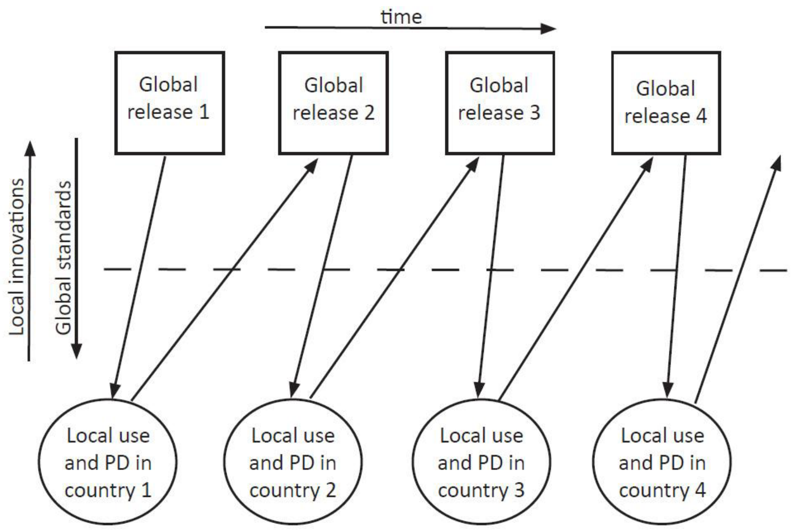
time
Global release 1
Global release 2
Global release 3
Global release 4
Local innovations
Global standards
Local use and PD in country 1
Local use and PD in country 2
Local use and PD in country 3
Local use and PD in country 4

Figure 1. The Titlestad et al. (2009, p. 42) model of PD across multiple country sites informing the longitudinal development of a generic software product

development team and the fact that end users are located at different workplaces all over Scandinavia, "[users] affect the development through user meetings, technical support and courses, but they never actually participate in design activities like workshops or mock-up evaluation" (ibid, page 5). The authors further note that as design and use becomes distributed, user participation shifts from designing a system to *using* and *evolving* it through various feedback channels. Similarly, Obendorf et al. (2009, page 21) point out that when PD was applied "[i]n increasingly distributed contexts, the focus shifted from custom software development to empowering users in assessing their own practice and technology use". Balka (2006) also notes the importance of customisation work in contemporary PD, as the process of getting off-the-shelf software products to work across multiple sites strongly resembles custom design. We note a transition from serial to parallel PD, as the overall design approach shifts from iterations of situated PD in a sequential manner (Figure 1), to collective filtering of requirements and local appropriation of generic features rather than custom design. Parallel PD emerges when software developers can no longer facilitate participation by travelling and engaging with users on site.

Parallel PD encompasses workshops and the circulation of software implementers, case studies, software documentation and the software itself. However, with a focus on even greater scale, a few PD projects have looked at how communities, with thousands of users, inform design. Hess et al. (2008), for instance, discuss the formation of online wikis and web forums to support a steering committee and a user parliament that represent more than 130,000 potential users in the design of a web-based net-TV solution. They found that outputs from the two user-centred governance bodies, the committee and the parliament, were difficult to coordinate, and that specific needs were hard to translate into general requirements. More specifically, their study found that potential end users who participated in online forum discussions started to doubt their influence on the design process because most actualised suggestions originated with the project's product manager. Oostveen and Van den Besselaar (2004) note that conflicting interests and misunderstandings are likely to be commonplace in very large PD projects. Their study highlights that mutual understanding, rather than consensus, within the community, is a key priority, albeit difficult to achieve. Beyond PD, the transition towards community distillation of generic requirement, by increasingly heterogeneous and distributed users, has been observed in the development of global commercial software packages such as Enterprise Resource Planning (ERP) software (Pollock and Williams 2007).

Table 1 summarises key points from the narrative literature review provided above and sorts PD into four types: singular, serial, parallel and community PD. The typology characterises PD at different levels of scale. For each type we describe three key aspects of PD as synthesised by Kensing and Blomberg (1998): (1) the politics of design, (2) the nature of participation, and (3) methods, tools and techniques for carrying out design projects. The politics of design concerns "the empowerment of workers so they can co-determine the development" (Clement and Van den Besselaar 1993, p. 29). Politics of PD also relates to the type and range of skill development a technology affords, as well as how issues of power and inequality are dealt with, for instance through system flexibility (Brigham and Introna 2007). The nature of participation concerns how and to what extent user interests may be accommodated in design by, for example, mitigating the constraints of hierarchical structures, narrowing the distance between developers and users, or by bridging design-use knowledge gaps (Balka 2006). PD methods, tools, and techniques, which we hereafter refer to as PD techniques, are practical measures that facilitate user participation in design.

Although the literature we have reviewed here provides rich examples of PD facilitation across distributed settings and within a few very large communities, few scholars have investigated the role of architecture in large scale PD. We hold that architecture is a key feature of both contemporary software products and the ecosystems that surrounds them. In the following subsection, we present concepts that we find relevant to study the role of architecture in large scale PD.

2.2 Balancing flexibility for design and use through architecture

As the literature review in section 2.1 highlights, with increasing scale of the PD venture, the distinction between design and use of the software product becomes increasingly blurred. Appropriation by new users and for new uses can take place either out of the box, through customization of generic features, or through the local development of custom extensions. However, this flexibility partly depends on architecture, which can exert a strong influence on software development and design (Baldwin and Clark 2006). In accordance with Van Schewick (2012) we understand architecture as the inner structure of a system, the components, what they do, and how they interact. This description of architecture applies well not only to software products and technical artefacts, but also to the structure of the surrounding ecosystems, including

Type of PD	*Key characteristics*	*Literature examples*
Singular PD	**Politics of design:** Users make feature-oriented design decisions based on concrete work tasks. **Nature of participation:** Situated and iterative face-to-face interaction between developers and users. **Methods, tools and techniques:** Prototypes, mock-ups, role-play, and workshops allow for hands-on evaluation of features by end users.	Braa (1995) Ehn (1988) Muller and Kuhn (1993)
Serial PD	**Politics of design:** Developers and implementers configure and customise software to fit distributed but relatively similar work practices in collaboration with end users. **Nature of participation:** Developers and implementers work closely with end users and make on-site visits. **Methods, tools and techniques:** Implementers act as design mediators between users and developers. Implementers move from site to site and iteratively contribute to enhance the core system.	Finck et al. (2004) Gumm (2007) Titlestad et al. (2009)
Parallel PD	**Politics of design:** Power users and local experts play a central role in eliciting requirements. Multiple projects undergo parallel design processes. Generic features receive priority over particular ones. **Nature of participation:** Software developers, customisers and implementers make short field visits and arrange workshops with user representatives. **Methods, tools and techniques:** Interaction between developers and users takes place through meetings, trainings, mailing lists, support calls and participant observation.	Hansson et al. (2006) Karasti (2010) Obendorf et al. (2009)
Com-munity PD	**Politics of design:** The broader community negotiates generic features. Local adaptations meet specific needs without involvement from the core development team. **Nature of participation:** Power users and domain experts represent end users in meetings and workshops. The focus is on appropriation of the existing software across diverse uses and settings. Core developers make final generic product decisions. **Methods, tools and techniques:** Documented case studies and best practices concerning software appropriation circulate through workshops, newsletters and online resources. Inter-contextual workshops, online forums and social media facilitate feedback from user representatives and the wider community of users.	DiSalvo et al. (2012) Hess et al. (2008) Karasti and Syrjänen (2004) Oostveen and Van den Besselaar (2004)

Table 1. Typology of PD

community management, requirements filtering, development processes and use of information and tools to support interaction around design (Markus 2007).

Here we are concerned with how architecture influences key aspects of PD such as the politics of design, the nature of participation and the techniques employed to facilitate participation in design. For instance, as we noted above with reference to the politics of large scale PD, the power to influence the design and use of software tends to transition to a cadre of design mediators such as customisers, implementers and power users who represent end users as projects move from singular to serial and parallel PD. These design mediators understand how local needs can be accommodated through the configurable features of a software product (Titlestad et al. 2009). They are thus in a position to leverage an architectural understanding of the software to operationalise and give priority to design suggestions. Furthermore, end-user participation in development and adaptation of a software product can be enabled or constrained by the level of flexibility with the software itself (Wulf et al. 2008), which largely hinges on its architecture.

Despite the blurred distinction between design and use in practice, we here note a conceptual difference between flexibility for further development, which we refer to as 'design flexibility', and flexibility out of the box for use across a range of different, even unanticipated purposes, which we dub 'use flexibility'. Design and use flexibility have been described as relational in the context of large-scale information systems, as a high level of use flexibility reduces the need for change through design flexibility, and vice versa (Hanseth and Lyytinen 2004; Hanseth et al. 1996). Furthermore, the actual design flexibility of large and complex systems has been linked to modularity (Braa et al. 2007; Edwards et al. 2007). Modularity is the idea that rather than developing one complex system or product where functional interdependencies are hard to discern, one should strive to develop small and reusable parts with clearly defined interfaces between them (Baldwin and Clark 2006; Baldwin and Woodard 2008; MacCormack et al. 2006).

Platforms constitute one prominent example of scalable modular and layered software architecture that can be leveraged to balance design and use flexibility (Olleros 2008; Yoo et al. 2010). Platforms are architectures comprising three key elements: core components with low variability, complementary components with high variability, and interfaces for modularity between core and complementary components (Baldwin and Woodard 2008). Hence, platform architectures may allow PD practitioners to address the age-old challenge of catering for new users that were not part of early design process and allow them to adapt software in unforeseen ways (Ehn 2008). The separation of systems into a stable core and a complementary set of components where variability is high allows for greater design flexibility at the outer layer without compromising the core (Baldwin and Woodard 2008; Olleros 2008). The key purpose with a platform is to facilitate an economy of scale of the lean core, while further development can take place through the core's boundary resources (Ghazawneh and Henfridsson 2013; Yoo et al. 2010), such as standardised Application Programming Interfaces (APIs) and Software Development Kits (SDKs). Standardised interfaces allow platforms to be open for participation, especially through the development of third-party apps (Gawer 2009; Ghazawneh and Henfridsson 2013). In essence, platform architecture defines how innovation is partitioned between app developers and the developers of a platform core, thus shaping the space for participation in local innovation (Tiwana 2013, p. 38).

To develop software product features that are both desirable and feasible, a great deal of knowledge is needed, both about situated contexts of use and about the software architecture.

For instance, as Hess et al. (2008, p. 36) report from a large scale web-based community PD project, "[t]he technical preview, which had been introduced at the beginning of the project, had some unchangeable aspects that restricted the possibilities for innovation." Hence, the initial assumptions and design values of developers and early users both enable and constrain further design flexibility as they lay the foundation for the software architecture (Plantin et al. 2016). Historically, the technical aspects of software development have not been taken sufficiently into consideration in PD literature (Hansson et al. 2006). This, we argue, is particularly true when it comes to architecture. In this paper, we highlight the role of architecture in shaping the politics of design, the nature of participation and the use of PD techniques. We do this by following, over time, the differences in PD approaches along with the growing scale of a PD project. With respect to the implications architecture has on PD, we focus on the emergence of a platform and its surrounding ecosystem, which we in accordance with Tiwana (2013) consider to be inseparably intertwined.

3 Method

We report from a longitudinal case study of an ongoing, international development effort that currently involves health information system strengthening initiatives in more than 50 countries. We begin by describing the initial growth of a PD project 20 years ago, which introduces the context and background for our empirical research. Subsequently, we present our research approach, data collection, and data analysis.

3.1 Case context and background

Our empirical case concerns the participatory design of the District Health Information Software (DHIS), which primarily supports data collection, data analysis and visualization to inform decision making in public health care organizations. A group of scholars with the Health Information System Programme (HISP) at the University of Oslo in Norway mainly coordinates DHIS development and implementation. The software development efforts are intertwined with research and capacity building in the 'Global South'. The initiative has, from the very beginning, been guided by Scandinavian principles and methods of PD and action research (e.g., Braa and Vidgen 1995; Checkland 1991). Throughout the history of DHIS development, university employees have played multiple roles as researchers, software developers, project managers, support staff, trainers and evaluators. Members of the core development team in Oslo hold software engineering positions at the University of Oslo and receive funding primarily through donor grants.

The DHIS software started out as an MS Access-based desktop application to support tasks of public health care management in three pilot districts in Cape Town, South Africa, in 1994, following the fall of apartheid. At the time, emancipation was a central political theme, both in terms of equitable health service provision to all people, irrespective of background and 'race', and in terms of empowering local health authorities to offer services in the ways they saw most appropriate. Thus the democratizing aims of Scandinavian PD found fertile ground, and DHIS evolved through rapid prototyping with local district staffs (Braa and Hedberg 2002). Over the next few years, DHIS saw a relative success in South Africa, emerging as the de facto national health management information system. DHIS slowly gained a foothold in other countries along with the cultivation of a network of stakeholders including universities, ministries of health, global regulatory agencies like WHO, international development funders, and emergent regional implementation partners such as HISP South Africa and HISP India (Braa et al. 2004).

In 2004, due to technological advances, growing computer literacy, and increasing access to the Internet by many potential DHIS users, work started on DHIS version 2 (DHIS2), a flexible and generic web-based tool developed entirely as free and open source software (FOSS). The new version involved a complete re-engineering and rewriting of the software code. The PD efforts forming the empirical basis for this paper primarily concerns the years after the launch of DHIS2 in 2006. We focus on this period, as this is when the traditional Scandinavian PD approach encountered marked challenges of scale in terms of the number and distribution of heterogeneous settings, developers, users and uses. DHIS2 implementations have scaled to more than 50 countries and different domains such as logistics, patient follow-up and disease surveillance, which in turn has affected the politics of design, the nature of participation and the use of PD techniques.

3.2 Research approach

The research reported on in this article has grown out of the four authors' individual analyses and collective discussions and reflections concerning the role of PD across different DHIS2 development and implementation settings. The case study is interpretive (Walsham 1995; 2006) and builds on the authors' experiences from DHIS2 project activities as well as contextual data obtained through participation in implementations, pilot projects, DHIS2 workshops and trainings. We present three vignettes from our longitudinal case study in section 4.1 - 4.3 to highlight how different types of PD have emerged in response to challenges of scale. Our combined research experiences cover multiple heterogeneous implementation sites, user groups and uses that relate to the same software development effort over a relatively long period. This allows us to explore the relationship between architecture and large scale PD, which itself is multi-site and long-term in nature. Our vignette selection represents a continuum of scale by zooming in on different, albeit overlapping, trajectories and settings.

3.3 Data collection

Table 2 below summarises the sites and activities that constitute the authors' sources of data for each of the three vignettes. The table also details our role-based access to secondary longitudinal and contextual data. More specifically, Author 4 has collected longitudinal data about the overall scale-up of DHIS2 development and implementation since 1996. This has taken place through research collaborations, participation in workshops, discussions in person and email, and field trips to countries with DHIS implementations.

Primary data collection for Vignette 1 from Sierra Leone constituted six field visits in the period 2008-2011. Author 3 organised two three-week training courses at the time of the initial introduction of DHIS2 to Sierra Leone, for all district monitoring and evaluation (M&E) officers and managers of health districts, as well as six M&E officers working at the Ministry of Health central office. More than 40 people participated in each course. The courses also served as venues for software customisation and requirements elicitation. Four district offices received subsequent visits twice for follow-up observations and interviews. Author 3 engaged in participatory observation and development activities during course sessions and district visits. In addition, Author 3 carried out informal interviews and group discussions with M&E officers and managers. The interviews were concerned with information needs in relation to current work practices, aspects of data quality, and the suitability of the new system including software functionality. Later visits followed up on user experiences with the system, as well as any changes made to software user interfaces, reporting forms, and software functionality. Field notes and photographs recorded observations in Sierra Leone. Extensive email exchanges documented software development and customisation efforts while Author 3 was not in the country.

	Primary data generating activities (case vignettes)	*Secondary longitudinal data (case context and background)*
Author 1	Implementation, customisation, training and software development with DHIS2 'Tracker' in Uganda, 2012-15 (Vignette 2) Participation in DHIS2 Academies 2012-15 (Vignette 3)	PhD student within the HISP and lead development coordinator of DHIS2 mobile phone-based features, 2011-2015
Author 2	DHIS2 Academies 2013-15, DHIS-Mobile workshops and strategy meetings 2009-15, DHIS-Mobile implementation, training, evaluation (Vignette 3)	PhD student, Postdoctoral fellow and project coordinator with the HISP 2010-2016 Involved with Implementations of DHIS2 mobile solutions in India, Malawi, Zambia and Uganda
Author 3	Implementation, customisation, training, management of development in relation to Sierra Leone 2008-10 (Vignette 1) DHIS2 Academies 2010-13 (Vignette 3)	PhD student, Postdoctoral fellow and project coordinator within the HISP 2002-2016, Involved with Implementations of various DHIS2 features in Cuba, Botswana, India, Malawi, West Africa
Author 4	DHIS2 Academies 2014-15 (Vignette 3)	Implementation evaluation, project advisor, management of global activities in the period 1996-2016, with a focus on South Africa, India, Ethiopia, Tanzania, Rwanda, Zambia

Table 2. Authors' roles in activities concerning the three case vignettes and the HISP

Author 1 collected data for Vignette 2 from Uganda during eight field trips of 1-2 weeks' duration between 2012 and 2015. Primary data collection activities constituted 23 interviews, participant observation of three in-country trainings, seven field visits to health clinics, four software developer workshops in Uganda and neighbouring countries, email discussions and phone conferences. The study informants included users at different levels of the health service, such as staff at the ministry of health central office, staff with national or regional responsibility for specific health programs, national and international NGOs working with health programs and software development in Uganda and local health clinic staff, including doctors, midwives and voluntary health workers. Interviews were conducted in the capital area, and during two field visits to rural health facilities in Uganda. The interviews focused on software feature requirements to support patient follow-up and clinical work practice. In one workshop, developers and users acted as patients and clinicians and went through the workflow of the software patient modules to elicit software requirements.

The empirical data concerning the DHIS2 Academies—Vignette 3—was collected through participation in seven workshops each ranging from eight to ten days, as well as the authors' involvement in five other workshops through remote planning and material development and online pre-Academy team-building activities with registered participants. Each author has participated in at least one DHIS2 Academy. Collectively, the authors have attended academies in West Africa, East Africa, Asia, and at the University of Oslo. The academies have ranged from 'basic', for newcomers to the software, 'advanced' for those with prior experience, and 'expert', attended mostly by NGOs, implementation partners and international donors. In most DHIS2 Academies, either the software development roadmap has had a dedicated time slot in the program or discussions have emerged ad hoc during workshop sessions. The authors have had access to material and presentations from the academies in addition to the authors' own notes made during academy participation. Author 1 videotaped one three-hour software requirements discussion during a developer workshop in Mombasa, Kenya.

Finally, we have all contributed to the collection and co-creation of naturally occurring data in the form of documents concerning DHIS2 development and implementation. This material includes reports, meeting memos, project evaluations, implementation budgets, grant proposals, software documentation and training materials. We have also studied requirement specifications and email lists at some level of detail, including a quantitative analysis of the developer mailing list that helped indicate different stages of scale in the history of DHIS2 development. For many of the workshops carried out in countries, relevant data in relation to user participation exists in the form of emails reporting on software bugs or feature requests. The authors, in their roles as facilitators or coordinators of user trainings, have mediated users' requests to the core developers on numerous occasions.

3.4 Data analysis

Data analysis has constituted iterative reflections on our long-term engagement with HISP and DHIS2, thus benefiting from overlapping data collection and data analysis (Boland Jr 2005) with blurred stages (Langley 1999). The initial motivation for the study came from two of the authors' observations that an increasing number of DHIS2 stakeholders expressed discontent with the form and content of user involvement in DHIS2 development. For instance, an NGO working out of the first country to adopt DHIS, complained, during a plenary session at a DHIS2 Academy in Oslo, that submitted feature requests could "be in limbo for more than a year" without any feedback or notification from the core software developers. The authors found this and similar claims paradoxical in the sense that user involvement, through participatory design, had always been the hallmark of DHIS2 development. We considered the observable discontent regarding lack of feedback and transparency with the development process to stem, at least in part, from stretched coordination capacity. In addition, we felt that that stretched capacity had created a gap between global software development efforts and distributed situated implementation processes. Our emergent research interest was consequently to investigate this problem with PD on a large scale.

The empirical data analysed in this study stems from the authors' previous research in the HISP concerning both the scaling of DHIS2 implementations (e.g., Braa et al. 2004; Sahay et al. 2013; Sanner et al. 2012) and the application of PD across different country settings (e.g., Braa et al. 2004; Kossi et al. 2012). Hence, this study combines data sets, collected by the four authors at different places and different times, to explore the longitudinal scaling of PD. Initially, we drew on this larger composite data set to identify empirical dimensions of scale in relation to PD in HISP. We arrived at dimensions that informed our working definition of scale as the *number* and *heterogeneity* of distributed *users*, *uses* and implementation *settings*. Next, we combined qualitative and quantitative analysis to review a large volume of threads on the DHIS2 developer mailing lists. This allowed us to identify implementation settings and events in our larger data set that were well documented in terms of user involvement in design and that represented scaling of PD in the HISP venture along the aforementioned dimensions.

To explore the interrelationship between scale and PD further, we singled out three case vignettes based on two criteria. First, the authors had extensive experience with the particular empirical settings (i.e., by having participated in DHIS2 implementation or training). Second, the chosen vignettes illustrated an increase in scale of the overall PD venture over time. Data concerning the case vignettes was analysed with a focus on how participation took place, what kind of tensions emerged regarding design prioritisations, and what participatory techniques were employed. In more detail, this included going through field notes and interview transcripts, videos and audio recordings to identify who participated (developers and users), how and on what issues participation was carried out, modes of communication (face-to-face on-site, email, trainings, etc.), and how user requests were categorised and responded to.

We took Kensing and Blomberg's (1998) three dimensions of PD, the politics of design, the nature of participation and the employment of PD techniques, as a starting point for fleshing out and comparing different forms of PD identified in our empirical material through the exercises mentioned above. This provided us with a framework for reviewing and synthesising extant literature concerned with large scale PD. We went back and forth between literature and empirical findings to define the key characteristics of four types of PD. Individual analysis work in relation to specific case vignettes was guided by meetings where we discussed our analytical categories through figures and tables drawn on whiteboards. A significant theme that emerged from our collective data analysis was the shifting nature of participation in tandem with the shifting architecture of the software itself. The software had gradually adopted what has been described in the literature as a platform architecture (Baldwin and Woodard 2008; Tiwana 2013), both technically and in terms of the structure of the network of stakeholders that surrounded it. Specifically, DHIS2 was remodelled to separate core functionalities (the software core) from its extensions (called apps). This led us to inquire further to connect aspects of an emergent layered and modular platform architecture with the four types of PD. Finally, we observed that design and use had become conflated and interdependent due to the widespread adoption and local customisation of the software. We identified design and use flexibility as characteristics that could help us describe the role of a platform architecture in shaping different types of PD.

4 The case of participatory design in HISP

In this section, we present three vignettes from our longitudinal case study. The timeline in Figure 2 shows important milestones in relation to implementations, capacity building, and technical features throughout DHIS development. The vignettes concern version 2 of the software, called DHIS2, rather than the first version. Although the four types of PD coexist over time, each type has been more prominent in certain eras of DHIS2 development. The yellow colour in Figure 2 indicates the era of singular PD, green is serial PD, blue is parallel PD and pink signifies the emergence of community PD.

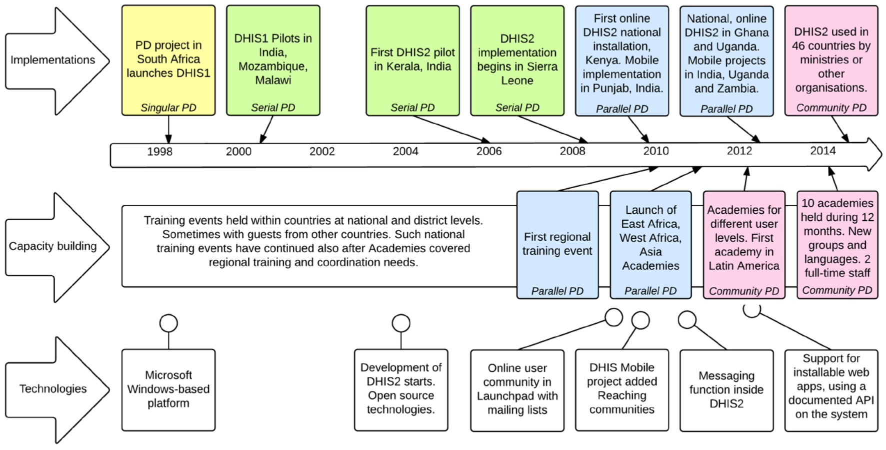
Implementations
PD project in South Africa launches DHIS1
Singular PD
DHIS1 Pilots in India, Mozambique, Malawi
Serial PD
First DHIS2 pilot in Kerala, India
Serial PD
DHIS2 implementation begins in Sierra Leone
Serial PD
First online DHIS2 national installation, Kenya. Mobile implementation in Punjab, India.
Parallel PD
National, online DHIS2 in Ghana and Uganda. Mobile projects in India, Uganda and Zambia.
Parallel PD
DHIS2 used in 46 countries by ministries or other organisations.
Community PD
1998
2000
2002
2004
2006
2008
2010
2012
2014
Capacity building
Training events held within countries at national and district levels. Sometimes with guests from other countries. Such national training events have continued also after Academies covered regional training and coordination needs.
First regional training event
Parallel PD
Launch of East Africa, West Africa, Asia Academies
Parallel PD
Academies for different user levels. First academy in Latin America
Community PD
10 academies held during 12 months. New groups and languages. 2 full-time staff
Community PD
Technologies
Microsoft Windows-based platform
Development of DHIS2 starts. Open source technologies.
Online user community in Launchpad with mailing lists
DHIS Mobile project added Reaching communities
Messaging function inside DHIS2
Support for installable web apps, using a documented API on the system

Figure 2. Summary of important milestones through 16 years of DHIS2 development

4.1 Vignette 1: Serial participation (DHIS2 in Sierra Leone, 2008-2011)

While the early period of DHIS development (see Braa and Hedberg 2002) can be classified as classical singular PD, DHIS version 2 was mainly developed through what we have labelled serial PD from around 2005. Every new PD effort thus builds on and extends previous PD efforts at a different time and place. We turn to one particular implementation to show how this was carried out. Our focus is on the interplay between situated PD in Sierra Leone and global core development. The core development team was initially heavily involved in Sierra Leone, but then shifted their attention to new country implementations. The strongest influence from PD in Sierra Leone on the 'global' development of DHIS2 was thus in the early implementation phase.

Sierra Leone was one of the first countries to implement DHIS2. At the time, the development team in Oslo mainly consisted of Master and PhD students, who did software development and implementation work as part of their degrees. The bulk of users were the monitoring and evaluation (M&E) officers at the district level, who were in charge of obtaining approximately 15 different paper reports from each health facility, and entering the data into DHIS2. In each district, there were two M&E officers, as well as the District Medical Officer (DMO), who was in charge of administering health services in the district. In addition, there were different health program officers responsible for different service areas. At the national level, a team of three people led the efforts to customise the national database and keep it synchronised by importing data from the offline district databases. Their roles included supportive supervision of the district staff and communication with DHIS2 implementers and developers. The project lead, with the World Health Organization (WHO) in Geneva, had no prior experience with DHIS2. However, while not an in-country user, the project lead was active in terms of accessing the Sierra Leone database, preparing information products, and conducting analysis. The distribution of various DHIS2 users, in relation to a single-country implementation, points to differences in user involvement even at an early stage of development.

For the most part, user involvement pertained to customisation of DHIS2. Only the more advanced requests made by the project lead at WHO were translated into direct changes in the software code. Inputs from district users mainly concerned local customisation of the database and user interfaces. The core DHIS2 development was, at this time, still based on redressing the functionality of DHIS version 1 into a web-based format. As DHIS2 had very few actual implementations, novel requirements from Sierra Leone received considerable attention. The Sierra Leone rollout started with a three-week intensive course for all district M&E officers and DMOs, many of whom had never used a computer before. Subsequently, core developers and implementers, including Author 3, visited four pilot districts. Each district needed on-site assistance in setting up a local copy of the software and database, as the extant infrastructure did not allow for an online implementation. During this period, user involvement was limited and feedback from users mainly concerned bugs in the software and hardware trouble. The initial focus was limited to data entry and the generation of simple reports.

Following the pilot implementation in four districts, further customisation of the system, such as defining metadata and custom report templates, was done in collaboration between developers and the national team in Freetown, mediated by the WHO project lead. Soon, however, district users provided feedback on the design of the custom reporting tools. In mid-2008, a core developer of DHIS2 visited Sierra Leone, and found that "health officers were happy with the improved reporting functionality, and many requested new reports and extensions to existing ones." The mentioned reports featured customisable templates, whereby the district users could potentially create new reports and alter report attributes. User involvement in the development of the DHIS2 software core, which required programming skills and deep knowledge of the existing code, was limited, but as the various users got more familiar with the system, ideas of improvement started to be voiced. The project lead, while based in Geneva, made frequent visits to Sierra Leone to work with the team in Freetown and communicated requirements to the core developers.

A major area of interest with the implementation was reporting completeness (i.e., whether the many health facilities in Sierra Leone had submitted their paper reports to districts and if these reports had then been entered into the database). To monitor this, new functionality had to be developed. This resulted in the design of a 'complete button'. By clicking on the button, the person responsible for data entry could mark a report as submitted (i.e., complete). Most of the time, a report would contain many empty data entry fields. Smaller clinics provide fewer notifiable health services, and some public health events are just rare. Hence, the label on the button read 'complete' to signal that despite an apparent lack of data, the report was complete. Completeness of reporting could then be calculated by tallying clicks on this button. However, many district users did not click on the complete button. Some forgot to click it, some had not received appropriate training, and some avoided the button because clicking it triggered form validation rules, which pointed out errors, and caused more work. An email exchange between the project lead and the DHIS2 lead developer from early 2009 touches upon the issue:

> An alternative would be for a DHIS2 expert to write some code that will automatically 'press the complete button' when one or more of the compulsory data elements in the form has a non-zero value. Please, please, please. (Project lead)
>
> We should develop completeness functionality in DHIS2 based on compulsory data elements as discussed before, will do this as soon as I get time (Lead developer)

Later, the possibility to define some fields in forms as compulsory was developed, which in turn allowed for another way of measuring data completeness. The vignette from Sierra Leone shows how user participation took place in relation to the local deployment and the global software development. Getting feedback from end users at the district level was feasible, with Author 3 and the local customisation team traveling from district to district to engage in support activities and requirement discussions. However, with new countries adapting DHIS2 in the years to follow, including whole states in India, user participation soon turned into streams of parallel activities. For a while, the development team in Oslo was able to be involved in and follow up on most implementations. Developers travelled extensively and engaged in requirements elicitation from countries, while long-term follow up was facilitated through communication via phone calls, emails and mailing lists.

4.2 Vignette 2: Parallel participation (DHIS2 Tracker in Uganda 2012-2015)

Vignette 2 concerns parallel participation in the development of the DHIS2 Tracker module. By 2012, several countries had implemented DHIS2. Kenya was the first country to deploy a fully online instance in 2011, where a central national office could manage all database and server maintenance. This was to become the dominant mode of DHIS2 deployment in subsequent implementations, testimony of the great mobile infrastructure advances across the developing world at the time. By 2012, a team of developers at HISP India had developed a local module to support clinical follow-up of pregnant women. This new module was a departure from the traditional aggregate statistics-only data model of DHIS2. It was also a departure from the norm where DHIS2 software developers were only situated in Oslo. At first, developers in Oslo considered this module as a hack, but they later re-engineered the highly demanded functionality of the module into the core of DHIS2, as the Tracker module. The DHIS2 Tracker was subsequently developed and implemented in several countries in parallel, and this vignette looks at how this process played out primarily from the point of view of an implementation in Uganda.

The Tracker was a new module in DHIS2 at the time of implementation in Uganda, and it still had design connotations to specific use in India. While a couple of countries started to use Tracker around this time, core designers from Oslo considered the Uganda implementation central in the design process:

> …we have to be part of the configuration of the system [in Uganda], or else we won't know which new features to implement and how to improve it. (Senior developer)

As major changes in the software were required to adapt to implementations in Uganda and other settings, several staff members from the core development team in Oslo were involved over a period of more than two years. The developers considered the implementation in Uganda as a template upon which one could forge generic features. Around this time, DHIS2 experienced massive adoption with the number of client countries growing quickly. Many of the new adopters considered implementing the new Tracker module to support health program follow-up. Consequently, while the developers engaged in situated PD in Uganda, there was an increasing pressure in terms of new requirements from other countries. Developers soon began to be involved in parallel development processes. This created tensions both among developers, and among local users and project stakeholders in Uganda.

The process of distilling generic requirements and developing corresponding software features was recognised as time consuming by local project staff in Uganda, who stated, "It is frustrating to have to wait for a new 'global release' every time a requirement or change has been suggested." The staff in Uganda also felt that their specific requirements were not met because the software had to maintain its generic features. This was brought up in a workshop with the Oslo-based developers, who believed they were largely accommodating local features, while the local staff felt their requirements were sacrificed on the altar of generality. An example of this tension was requirements regarding which start date to use when automatically scheduling events in a health program. The date of a person's enrolment in a health program and the automatic generation of schedules based on this date differ greatly from program to program. For a pregnancy, a commonly used start date is the often uncertain last menstrual period, leading to a roughly estimated date of delivery. For postnatal and vaccination programs, the start date is given as the birth of the baby, which is comparatively accurate. The logic of assigning dates is also different. Antenatal follow-up visits may be revised based on the condition of the pregnant woman, while vaccinations are predetermined by schedules at the health clinic. When the same generic software was used to manage many different types of programs, the scheduling mechanism, which was initially considered generic, turned out to restrict particular workflows. Over the entire project period in Uganda, no satisfactory solution was implemented to flexibly handle contradictory scheduling requirements.

For the core developers in Oslo, the emergent parallel development processes also had implications. The graphical user interface and features that had been carefully modelled in Uganda were subsequently picked up by design teams in other countries to fit other requirements. Because all countries used the same open source code base, developers working for one country could make adaptations that affected everyone else. One particular user interface changed back and forth several times, due to teams in different countries each adapting it to their needs. Some of the core central developers were caught in the middle, constantly changing the software and increasing complexity.

During this particular implementation period, DHIS2 went through a major architectural change and key features were revised to fit a new modular architecture. When a team of central developers rewrote the code base for the Tracker module to fit the new platform concept, a large part of the embedded local requirements from Uganda were lost. This led to heated discussions within the central development team, whereupon one of the central developers expressed his sentiments by saying, “You cannot expect your requirements from one case in one country to be leading the requirements for this generic software.” The comment illustrates how the overall DHIS2 development had moved from a serial PD-oriented process to a process that encompassed parallel cases, sometimes with conflicting requirements. It was no longer enough to engage in participation on the ground, since ‘the ground’ constituted many different realities.

The design experiences from Uganda also became central during the development of a new Android client meant to extend the DHIS2 Tracker module. A key focus with the Android development was to make a software development kit (SDK) that would allow different projects to make their own mobile applications with specific user interfaces, rather than a configurable but non-extensible product. The vignette concerning the Tracker in Uganda is representative of a transition in the conduct of PD in the HISP. First, it shows an emerging *parallel* mode of PD, with some developers active in one country while others were active in another. Tensions ensued and led to the adoption of a more pronounced strategy for the development of generic software. The motivation for PD would then shift towards feeding inputs to this software development process, while at the same time trying to customise generic software modules to meet local needs. Later, the mounting tensions between the parallel projects informed the development of a more platform-oriented software architecture. This phase was also characterised by a transition where the core team started using the platform’s capabilities themselves to make applications using open APIs, rather than the earlier mode of implementing functionality inside the core and just exposing APIs for others to use. While generic functionality remained hidden in the software core, a customisation layer and a support structure emerged to support the development of custom applications as extensions of the DHIS2 core. Vignette 3 in the following sub-section details the emergence of a suite of workshops, called DHIS2 Academies. The DHIS2 Academies have played a key role in balancing software development with growing community expectations and needs.

4.3 Vignette 3: Community participation (DHIS Academies, 2010-2015)

The third and last vignette illustrates Community PD and concerns the DHIS2 Academies. Since 2010, the DHIS2 community, with coordination from Oslo, has arranged a number of 1-2 week workshops targeting mainly national and regional implementation teams. Only a few academies were held the first few years. However, more recently, academies are held regularly in Africa, Asia and Latin America, and have diversified to cater for implementers and software customisers at different experience levels. The academies mainly consist of training, but they also provide an arena for local implementers to keep up to date with recent software developments, share use cases and experiences, build professional networks, and give feedback to developers directly.

In the period covered by this study, core developers would regularly participate at DHIS2 Academies. This ensured close communication between them and implementers, who represented local users. The academies included sessions where developers presented new and upcoming features of the software. Participants could then discuss this and suggest further requirements. The DHIS2 Academies thus offered a better alternative, logistically and financially, than distributed discussions confined to individual countries. In addition, participants expressed that they valued coming together to share experiences through mingling and plenary discussions about software use, feature requests, technical challenges, and the organising of the DHIS2 community itself. During a 12-month period from 2014-2015, ten DHIS2 Academies were held with the number of fee-paying attendees in each workshop exceeding 50, which illustrates the significance of these workshops.

With a community of users and user representatives from different countries, the Academies provide a venue for distilling requirements and avoiding the coordination hurdles encountered with parallel PD. During a DHIS2 Academy in Mombasa, Kenya in 2012, a group of 12 specialists from different countries in east Africa spent three hours discussing requirements for mobile phone-oriented features in DHIS2. During a discussion concerning a feature that had been difficult to implement across multiple contexts, one of the core DHIS2 developers shared his positive assessment of the experience, saying,

> In order for us to create new functionality, we really need a proper use case, a real one, so we can communicate. We cannot just [gesture showing something coming out of the air]… We have been thinking about this [feature] for the longest time, from the very beginning, but we never found good examples of what would be beneficial for the health worker as we've discussed today.

In particular, generic functions that have local variations in workflow and local improvisation have been difficult to develop without discussions concerning specific use cases, with one example being integrated disease surveillance and response (IDSR). IDSR is a process for discovering and managing disease outbreaks. IDSR has a number of global WHO guidelines, but still requires local adaptation to succeed in practice. During an Academy session, a discussion on usage of SMS alerts for IDSR opened the requirement scope to include many other non-IDSR functions that could benefit from SMS alerts, such as SMS reminders to health workers to improve data quality in routine health data reporting. One of the DHIS2 Academy coordinators thanked the group by saying,

> This is very good. We have been thinking about some of these cases, but very narrow. Now it's clear that it's a whole area with different […] it has more than one use case.

The academy discussions mentioned above helped clarify design features that despite international guidelines had local variations and idiosyncrasies. During the same session, one of the coordinators tried to uncover local variations by saying,

> I'm wondering about IDSR. When there is an outbreak, to which extent is the process based on these structured processes versus people picking up the phone, coordinating on the phone, and breaking out of the process…

A long discussion followed, where representatives shared their views on how IDSR was managed in their respective countries.

In addition to providing valuable requirement input to the core development team, participants stated after the session that it had been useful to discuss use across national boundaries. Since the Mombasa Academy, several of the attendees have been active in arranging new DHIS2 academies and have been teaching and implementing DHIS2 in various countries. The sessions were videotaped so that other community members such as developers and implementers who did not attend the workshop could learn from the discussions. The workshop was an attempt to bring developers from the other side of the globe closer to users, or at least their local representatives. Interestingly, despite the consensus during the session on a number of features for IDSR, these features were not implemented in the software due to resource limitations, until 2014 when a large international NGO paid for the development of the features as part of their DHIS2 adoption. This brings up another aspect with increased scale of PD projects: the prioritisation of design proposals.

At an Expert Academy in Oslo in 2015, with more than 80 participants from international NGOs, individual consultants, and some prominent country implementation representatives, a session was held on the software development process and how various stakeholders could interact with it. Although prioritisations and final decisions about standard releases are made by the core development team in Oslo, the development coordinator listed three different ways for community members to propose new features: 1) write on the developer mailing list, 2) participate at DHIS2 Academies, or 3) write and upload a well-defined specification. However, these modes of participation are not equally accessible to all users and organisations. Interestingly, the lead developer also presented a list of six qualitative evaluation criteria employed to rank functional requests. The list constituted the following criteria: perceived usefulness to the wider community of users, level of software development effort, perceived benefit beyond what is already possible with the software, interdependencies with other functions and modules, expected level of participation and feedback from the owner of the request, and availability of funds. Based on these criteria, it is not surprising that late adopters such as well-funded international

NGOs have become increasingly influential in shaping the DHIS2 software development agenda. The DHIS2 Academies also serve as venues to vent frustrations and discuss concerns, such as potential domination from resourceful international NGOs over early adopters of DHIS2 with limited technical capacity and funds, like ministries of health in less developed economies.

International NGOs with relatively deep pockets are able to travel and sit down with the core software developers in Oslo, and therefore have the opportunity to invest in more face time to make sure their requirements are well understood and recognised in future development of the software. These stakeholders are also in a position where they can fund or hire developers directly to implement their specific requirements, for instance as third party apps, and their wishes are more likely to make it into production at an earlier stage. A recent trend with DHIS2 development has been the development of third-party apps by international NGOs to meet the needs of users in their own organisations. Many of these apps could potentially be useful to other organisations such as ministries of health. However, most custom apps have not been shared with the broader software community. One possible explanation for this is that sharing generates additional responsibilities and costs associated with making solutions more generic and maintaining them over time.

The vignette from the DHIS2 Academies shows how community PD has emerged as an additional type of PD in the HISP over the last few years. While some developers continue to be engaged in specific projects and parallel PD, there are just too many activities for the global development team to be engaged everywhere. The DHIS2 Academies, which target users and community members at different levels, have emerged as occasions for community filtering of requirements and getting feedback from a variety of users.

5 Discussion: Architectures for participation

In section two, we reviewed extant literature and identified four types of PD with respect to scale, namely: singular, serial, parallel, and community PD. From our longitudinal case study, we find that all four types of PD have emerged along with the scale of the project. This can be seen as a consequence of key developers and project managers trying to uphold political and pragmatic ideals of user participation, while grappling with a rapid increase in the number and distribution of heterogeneous user needs and requirements (Braa and Sahay 2012; Shaw and Braa 2014). Rather than one type of PD replacing another entirely, we find that different types of PD coexist and interplay in the development of DHIS2. In parallel with the emergence of different types of PD, we also note the emergence of a generic software product, increasingly structured as a platform. We now turn to our discussion on how the emergent modular and layered architecture enables and constrains PD along three key dimensions (Kensing and Blomberg 1998): the nature of participation, the politics of design, and techniques for participation.

5.1 The nature of participation

One could argue that DHIS2 has been designed with a layered architecture from the very beginning, the backend with data models and core functional modules, and the frontend with customisable features and user interfaces. While the backend would be invisible to most users, the frontend would allow for some customisation of both the backend content and frontend views. This included configuration of what data to collect, process, and present to users. Early implementations of DHIS2 thus applied PD in frontend customisation, administered by implementers together with local users. As Vignette 1 from Sierra Leone illustrates, there were occasional PD processes whereby core developers made changes to the software code. This primarily took place when the scope for customisation was exhausted, and additional features needed to be developed.

Over time, DHIS2 was adopted by more diverse users and organisations, in distributed settings (Gumm 2006; Obendorf et al. 2009). We understand this as increasing scale. The modular development of generic functionality allowed for localisation and appropriation through the customisation of a frontend layer. In turn, this allowed for parallel PD processes across implementations. As the core developers were increasingly unable to be directly involved in implementations, they came to rely more on communication with DHIS2 *implementers*, who were skilled mediators of software requests and issues. This, however, necessitated the availability of qualified mediators who knew the software well and could translate requirements to core developers.

Tensions between design and use were frequent with the release of new features that were derived from particular use cases in a particular context. With new features derived from situated PD, the flexibility for use across a range of tasks may be low because alternative areas of use have not been explored. Even if parallel PD is employed to develop more generic solutions, increasingly distributed adoption and use of the feature places constraints on how well this can be achieved without compromising particular needs (Hansson et al. 2006). Vignette 2 concerning the Tracker implementation in Uganda exemplifies this tension between generic design and local use.

The emergent platform architecture of the software offered a partial remedy to tensions between the generic and the specific. In addition to the development of standard bundled apps, the division into a stable core and boundary resources, such as APIs and SDKs, allows for the development of custom apps to meet particular user needs. In short, a platform architecture allows innovation to take place locally without compromising any of the software components used more widely. The application of PD is then uncontroversial in terms of conflicting user needs and the logistics of PD. On the downside, useful functionality developed as custom apps by local developers often remains unknown beyond the local implementation. Hence, a software platform architecture alone is not a strong vehicle for very large scale PD, unless it receives support from its surrounding ecosystem. This can for instance be facilitated through inter-contextual workshops, such as DHIS2 Academies, and community circulation of power users' case studies (Obendorf et al. 2009), or video material from discussions and presentations as we have also seen in the case of DHIS2.

The organisation of capacity building and community management efforts in HISP has mirrored the growth and scale of DHIS2 software adoption. The DHIS2 Academies emerged

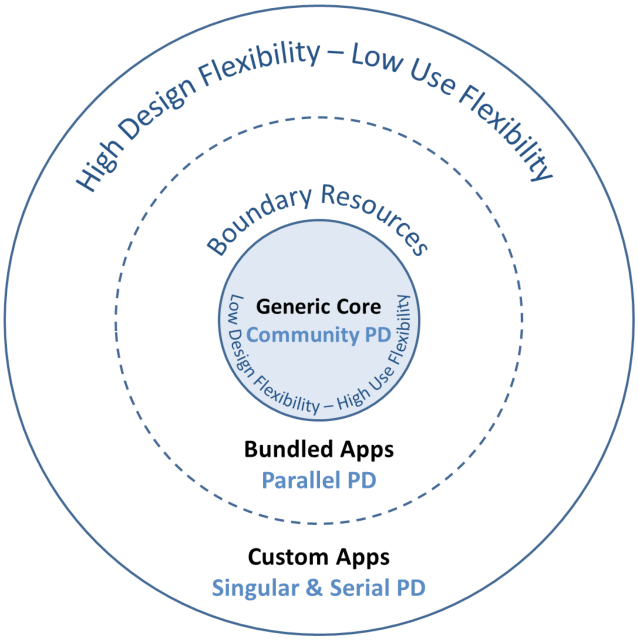

Figure 3. A platform architecture for Participatory Design

as a more cost-effective and efficient answer to the need for distributed training and experience sharing and could be seen as integral to the DHIS2 platform ecosystem. Hence, workshops play a key role in shaping the nature of participation in that they offer a venue for community PD, where participants negotiate design on behalf of different user groups. The latter is important in that it both smoothens out differences between various groups' design requirements, through community filtering (e.g., Pollock and Williams 2007), and that it allows core developers to identify potential resolutions to conflicting user needs by altering design at different levels of the software architecture.

In sum, we find that a *platform,* as one type of architecture, constitutes an enabler of large-scale PD (Figure 3). The concept of a platform, much like the notion of architecture, can denote both a model of the software itself and the ecosystem around it. The platformisation of both the software and the surrounding ecosystem allows PD to scale. With reference to our definition of scale, seen as the number and distribution of heterogeneous settings, developers, users and uses, we find that a platform supports singular PD, predominantly at the outer layer, as shown in Figure 3. The challenges related to logistics of participation and design priority setting has been mitigated by the platform architecture, both because more singular PD can be carried out on specific apps, and because community PD plays a harmonising and standardising role in managing the platform core. However, the technical architecture of a platform and its surrounding ecosystem are complementary. A modular and loosely coupled software architecture will itself not necessarily support scaling of *participation in design*. Likewise, structures that support the logistics of participation and negotiation of requirements will not necessarily succeed without a flexible software architecture that allows heterogeneous user inputs to be accommodated in design (Tiwana 2013; Wulf et al. 2008).

The configurable middle layer of the platform allows implementers to customise the software to meet end user needs within the scope of the pre-built apps. These apps are generally part of the standard software releases, labelled as 'bundled apps' in Figure 3. Recall that in section 2.2 we differentiated between flexibility for further development, which we refer to as design flexibility, and flexibility for use across a range of different purposes, which we dub use flexibility (Hanseth and Lyytinen 2004; Hanseth et al. 1996). In general, the bundled apps have a high degree of use flexibility through customisation, but low design flexibility due to software interdependencies between bundled modules and widespread adoption and use. As is currently the case with DHIS2, only senior developers, employed in full-time positions, work on the bundled apps, which provide the most widely used functionality of the platform. The dashed line in the outer layer in Figure 3 illustrates that although bundled apps need to interface with the platform core, through boundary resources, in the same way as custom apps, they differ in how they are developed and maintained. In particular, the core development team coordinated the development of bundled apps through standard software releases. The outer layer in Figure 3 represents custom or third party apps—often with low flexibility for use across tasks, due to specific design requirements, but high flexibility for further design improvements, due to fewer dependencies.

5.2 The politics of design

Figure 3 above illustrates how we consider a platform architecture to support different types of PD at different layers. This in turn has implications for the politics of design, in the sense that different power struggles may take place in relation to different platform layers and through involvement of different stakeholders. As we have pointed out, the core of a platform is relatively stable, and its developers seek generic solutions. As is the case with most software platforms, the core developers, such as the DHIS2 senior developers in Oslo, make final decisions regarding feature roadmaps and revisions of the software core. Generic core development steers the general movement of the software by allowing for new functional possibilities and opening up for use in new domains, based on assessments of community needs.

The development and refinement of a platform core is far removed from traditional politically motivated PD projects of empowering situated blue-collar workers. Instead, the political project of core development may be to focus on eliciting requirements from disempowered organisations, such as ministries of health, in low and middle-income countries. In particular, the political agenda could be to assist governments in their struggles with proprietary software vendors and large international organisations who seek to steer development for their own reasons and on their own terms. Through Community PD, the target for empowerment has shifted from the situated individual worker towards whole organisations and large cadres of users. Building capacity, both technical and organisational, is key to empowering organisations to design and configure the information systems they need, in the fashion they deem most relevant.

Classical situated PD is more pronounced at the outer layers, but will occasionally lead to changes in the core, especially at points of major revisions, such as with the initial implementation of DHIS version 2 in Sierra Leone, where many design decisions were still up for grabs.

Similarly, the Indian Tracker module, which was initially seen as a hack, was later incorporated as a core DHIS2 module. As the core software modules have become more stable, there is now an informal set of evaluation criteria applied by the core developers to determine how, when and for what purpose significant revisions should be made. For instance, core developers may get involved in the redesign of custom apps, developed through outer-layer innovation, to ensure that new functionality can be bundled with standard releases and adopted by more users in different contexts. Common requirements can potentially be accommodated in the platform core, but a main activity is to provide a flexible data model and stable boundary resources for the outer layers to leverage. New modules or apps can then be developed loosely coupled to the platform core, through the involvement of either core developers themselves or local third-party software developers. However, the appreciation for and employment of PD principles by third-party software developers at the fringes of a platform can only be encouraged, not ascertained, by the core developers and political agents of the platform.

Similar to the platformisation of the DHIS2 software, support structures, such as the DHIS2 Academies have been divided into basic, advanced and expert levels. In turn, these levels have been thematically modularised to cater for different user roles and interests. Increasingly, core developers are no longer present at DHIS2 Academies held around the world. The formation of an annual Expert Academy in Oslo, with decision makers, funders, and a few country experts, maintains at least one venue where core developers and a heterogeneous group of user representatives and other stakeholders meet. However, the attendants at these Academies are of course not the typical DHIS2 users, most of which are data entry clerks and health facility and district managers using basic functionality, but implementers and national core team members who act as intermediaries. The level of end-user involvement in domestic requirement development processes vary, and is largely unknown to the authors.

5.3 Participatory design techniques

An important architectural development of a platform is the stabilisation of boundary resources such as a web-API and an SDK, which in theory allows any third-party developer to make custom apps based on local requirements (Tiwana 2013). As such, platform boundary resources serve as potential means to facilitate situated PD at the fringes of the platform. To date, DHIS2 apps have mainly been developed in Oslo. However, the development of boundary resources has made it possible to stabilise the core and at the same time allow for customisation, prototyping, experimentation, and local development. This has two important implications for PD. First, it allows situated PD to be carried out directly in relation to apps. PD can take place either through custom configuration of the widely used, generic apps bundled with standard releases, such as data entry modules and general visualisation tools, or more specific apps developed with specific work practices in mind. Second, people other than the core developers can do such development, since software programmers will only communicate with the software core through boundary resources like the web-API and the SDK without the need to comprehend the internal dynamics of the core itself.

The organising of the Health Information System Program (HISP) has followed a similar trajectory towards platformisation. User involvement has become layered too. In addition to direct *in situ* involvement, the DHIS2 Academy and corresponding user community channels such as email lists, forums, and online development resources allow for distributed and layered user participation. The Academies were begun in response to a growing number of implementation sites, to be able to offer training in customisation of DHIS2 bundled apps to more than one country at a time. As stated above, the DHIS2 Academies have shaped the nature of participation, but they are also important techniques for large-scale PD. Academies or workshops bring various users and developers together. They serve as arenas for sharing new requirements and development agendas, eliciting feedback from users, and for various users to learn from each other. Since the vast majority of implementations are in the field of health, the various stakeholders have some shared understanding through common needs. Health program management has to a certain degree been standardised through the efforts of large international agencies, such as WHO, though there will generally be local variations. Requirements have been synthesised and filtered through community discussions and negotiations at DHIS2 Academies. Common values and a mutual understanding within the community informs the search for generic solutions similarly to the community-filtered distillation of generic requirements observed by Pollock and Williams (2007) in relation to global ERP development.

In addition to the Academies, the email lists, online development resources, and other online channels such as pre-Academy training programs and an emerging online learning platform offering standardised courses, constitute the platform ecosystem. These tools and structures offer distributed users and developers an arena for communication. In general, techniques associated with traditional singular PD tend to transition towards the outer layer in Figure 3 above. New software implementations, such as when a country adopts DHIS2 to be a national health information system, typically focus on developing the system within the frame of extant functionalities. A growing number of experts and implementers can facilitate this customisation in close collaboration with local stakeholders. Hence, features that were once developed through someone's situated PD effort, has become incorporated and generalised into someone else's off-the-shelf software package.

6 Conclusions

Based on a narrative literature review, we identified four types of PD along the dimension of scale. We then applied these four types to our empirical case in order to address our research question: What role does architecture play in large scale PD? We do this by discussing the role of architecture in relation to key aspects of PD, namely the nature of participation, the politics of design and techniques for PD. In this paper, we have exemplified the application of PD through more than one decade of development and use of an increasingly complex software product. The main characteristic of this journey has been the emergence of a platform for PD. Specifically, we have considered PD in relation to a platform architecture and how the four different types of PD may play different roles in relation to the emergence of different layers of a platform and its surrounding ecosystem.

We derive three key points from this story. First, our model of a layered architecture shows that PD may take place even with a large base of heterogeneous users and settings. Layering, both in the software development and in the conduct of PD, can reduce the logistical challenges of large-scale PD. This helps resolve, through design and use flexibility, some of the conflicting demands that a growing number of heterogeneous users and uses can create.

Second, user participation in design is more direct and open ended at the fringes, where standardisation and generic features are less of a concern and dependencies are fewer. For instance, PD takes place in parallel through the customisation of bundled apps. The room for classical PD is particularly elaborate in the early stages of design of new custom apps. In these processes, boundary resources such as APIs and SDKs are both design enablers and constraints. Community PD can play a role in the developments of the software core and the more mature apps, but not as pronounced as new development projects around custom apps. More broadly, community PD plays a new political role in giving voice to different user groups and organisations and in allowing for transparent negotiations of requirements, interests and values that set the general direction for further software development.

Third, a platform for PD, as we see it, depends on support from both a technical architecture and a surrounding ecosystem. The modularisation of DHIS2 and the growth of the academies and various other community mobilisation channels have developed in tandem. This is not coincidental. Both these measures have been responses to growing scale, and both have allowed users to participate in design. We thus see the software architecture and its surrounding ecosystem as constitutive of a platform for large scale PD.

References

Baldwin, C. Y., and Clark, K. B., (2006). The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science*, (10:7): 1116–1127.

Baldwin, C. Y., and Woodard, C. J., (2008). The architecture of platforms: a unified view. In: *Platforms, Markets and Innovation*, A. Gawer (ed.), Edward Elgar Publishing, Northampton, MA, USA, pp. 19–41.

Balka, E., (2006). Inside the belly of the beast: the challenges and successes of a reformist participatory agenda. In *Proceedings of the ninth conference on Participatory design: Expanding boundaries in design-Volume 1*, G. Jacucci, F. Kensing (eds.), ACM Press, New York, pp. 134–143.

Balka, E., (2010). Broadening discussion about participatory design: A reply to Kyng. *Scandinavian Journal of Information Systems,* (22:1): 77–84.

Bjerknes, G., and Bratteteig, T., (1995). User Participation and Democracy: A Discussion of Scandinavian Research. *Scandinavian Journal of Information Systems*, (7:1): 73–98.

Bjerknes, G., Ehn, P., and Kyng, M., eds., (1987). *Computers and Democracy - A Scandinavian Challenge*, Avebury, Aldershot.

Boland Jr, R. J., (2005). Interpretation and theory in qualitative research. In: *The art of science,* S. Tengblad, R. Solli, B. Czarniawska (eds.), Liber & Copenhagen Business School Press, Malmö, pp. 219–236.

Braa, J., Hanseth, O., Heywood, A., Mohammed, W., and Shaw, V., (2007). Developing Health Information Systems in Developing Countries: The Flexible Standards Strategy. *MIS Quarterly*, (31:2): 381-402.

Braa, J., and Hedberg, C., (2002). The struggle for district-based health information systems in South Africa. *The Information Society*, (*18:*2): 113–127.

Braa, J., Monteiro, E., and Sahay, S., (2004). Networks of action: Sustainable health information systems across developing countries. *MIS Quarterly*, (*28:*3): 337–362.

Braa, J., and Sahay, S., (2012). Participatory Design within the HISP network. In: *Routledge International Handbook of Participatory Design*, J. Simonsen, T. Robertson (eds.), Routledge, New York, pp.: 235-256.

Braa, J., Titlestad, O. H., and Sæbø, J., (2004). Participatory health information systems development in Cuba: the challenge of addressing multiple levels in a centralized setting. In: *Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices-Volume 1*, A. Clement, P. Besselaar (eds.), ACM Press, New York, pp. 53–64.

Braa, K., (1995). Priority workshops: springboard for user participation in redesign activities. In: *Proceedings of conference on Organizational computing systems*, N. Comstock, C. Ellis (eds.), ACM, New York, pp. 258–267.

Braa, K., and Vidgen, R., (1995). Action case: exploring the middle kingdom in information system research methods. In: *Proceedings of 3rd Decennial Conference Computers in context: Joining Forces in Design*, Århus, pp. 50-60.

Brigham, M., and Introna, L. D., (2007). Invoking politics and ethics in the design of information technology: undesigning the design. *Ethics and Information Technology*, (9:1): 1–10.

Bødker, S., Ehn, P., Kammersgaard, J., Kyng, M., and Sundblad, Y., (1987). A Utopian experience. In: *Computers and Democracy - A Scandinavian Challenge*, G. Bjerknes, P. Ehn, and M. Kyng, (eds.), Avebury, Aldershot, pp. 251–278.

Bødker, S., and Pekkola, S., (2010). A short review to the past and present of participatory design. *Scandinavian Journal of Information Systems*, (22:1): 45–48.

Checkland, P., (1991). From framework through experience to learning: the essential nature of action research. In: *Information Systems Research: Contemporary Approaches and Emergent Traditions*, H.E. Nissen, H.K. Klein, R.A. Hirschheim (eds). North-Holland, Amsterdam, pp 397–403.

Clement, A., and Van den Besselaar, P., (1993). A retrospective look at PD projects. *Communications of the ACM*, (36:6): 29–37.

DiSalvo, C., Clement, A., and Pipek, V., (2012). Participatory design for, with, and by communities. In: *Routledge International Handbook of Participatory Design*, J. Simonsen, T. Robertson (eds.), Routledge, New York, pp. 182–209.

Edwards, P. N., Jackson, S. J., Bowker, G. C., and Knobel, C., (2007). Understanding infrastructure: Dynamics, tensions, and design. In: *Workshop on History & Theory of Infrastructure: Lessons for New Scientific Cyberinfrastructur*, National Science Foundation, Alexandria, Virginia, USA.

Ehn, P., (1988). *Work-oriented design of computer artifacts*, Lawrence Erlbaum Associates, Hillsdale, N.J.

Ehn, P., (2008). Participation in design things. In: *PDC2008: Proceedings of the Tenth Anniversary Conference on Participatory Design,* J. Simonsen, T. Robertson and D. Hakken (eds.), ACM Press, New York, pp. 92–101.

Finck, M., Gumm, D. C., and Pape, B., (2004). Using groupware for mediated feedback. In: *PDC-04 Proceedings of the Participatory Design Conference Vol 2*, A. Bond, A. Clement, F. Cindio, D. Schuler, P. Besselaar (eds.), CPSR, Toronto, pp. 45–48.

Gawer, A., (2009). Platform dynamics and strategies: from products to services. In: *Platforms, Markets and Innovation*, A. Gawer (eds.), Edward Elgar Publishing Inc, Northampton, Massachusetts, USA, pp. 45-76.

Ghazawneh, A., and Henfridsson, O., (2013). Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal*, (23:2): 173–192.

Gumm, D.C. (2006). Distributed participatory design: An inherent paradoxon. In: *Proceedings of 29th Information Systems Research Seminars in Scandinavia*, Y. Dittrich, D.L. Strand, J. Nørbjerg, O.W. Bertelsen, W.G. Bleek (eds.), AIS, København.

Hanseth, O., and Lyytinen, K., (2004). Theorizing about the design of Information Infrastructures: design kernel theories and principles. *Sprouts: Working Papers on Information Environments, Systems and Organizations*, (4:4): 207–241.

Hanseth, O., Monteiro, E., and Hatling, M., (1996). Developing Information Infrastructure: The Tension Between Standardization and Flexibility. *Science, Technology and Human Values*, (21:4): 407–426.

Hansson, C., Dittrich, Y., and Randall, D., (2006). How to include users in the development of off-the-shelf software: a case for complementing participatory design with agile development. In: *HICSS '06 Proceedings of the 39th Annual Hawaii International Conference on System Sciences - Volume 08*, IEEE, pp. 175c-175c.

Hess, J., Offenberg, S., and Pipek, V., (2008). Community driven development as participation?: involving user communities in a software design process. In: *Proceedings of the Tenth Anniversary Conference on Participatory Design 2008*, Indiana University, pp. 31–40.

Karasti, H., (2010). Taking PD to Multiple Contexts: A reply to Kyng. *Scandinavian Journal of Information Systems*, (22:1): 85–92.

Karasti, H., (2014). Infrastructuring in participatory design. In: *Proceedings of the 13th Participatory Design Conference: Research Papers-Volume 1*, O.S. Iversen, H. Winschiers-Theophilus, V. D'Andrea, C. Bossen, M. Teli, K. Bødker (eds.), ACM Press, New York, pp. 141–150.

Karasti, H., and Syrjänen, A.-L., (2004). Artful infrastructuring in two cases of community PD. In: *Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices-Volume 1,* A. Clement, F. Cindio, A.-M. Oostveen, D. Schuler, P. Besselaar (eds.), ACM Press, New York, pp. 20–30.

Kensing, F., (1987). Generation of visions in systems development: a supplement to the tool box. In: *The IFIP TC 9/WG 9.1 Working Conference on system design for human development and productivity: participation and beyond on System design for human development and productivity: participation and beyond,* North-Holland Publishing Co, Amsterdam, pp. 285–301.

Kensing, F., and Blomberg, J., (1998). Participatory design: Issues and concerns. *Computer Supported Cooperative Work (CSCW)*, (7:3–4): 167–185.

Kossi, E. K., Sæbø, J. I., Braa, J., Jalloh, M. M., and Manya, A., (2012). Developing decentralised health information systems in developing countries – cases from Sierra Leone and Kenya. *The Journal of Community Informatics*, (9: 2).

Kyng, M., (2010). Bridging the Gap Between Politics and Techniques: On the next practices of participatory design. *Scandinavian Journal of Information Systems*, (22:1): 49–68.

Langley, A., (1999). Strategies for theorizing from process data. *Academy of Management Review*, (24:4): 691–710.

MacCormack, A., Rusnak, J., and Baldwin, C. Y., (2006). Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, (52:7): 1015–1030.

Markus, M. L., (2007). The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management & Governance*, (11:2): 151–163.

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

Muller, M. J., and Kuhn, S., (1993). Participatory design. *Communications of the ACM*, (36:6): 24–28.

Neumann, L. J., and Star, S. L., (1996). Making infrastructure: The dream of a common language. In *PDC'96 Proceedings of the Participatory Design Conference*, J. Blomberg, F. Kensing, and E.A. DykstraErickson (eds.). CPSR, Cambridge, MA USA, pp. 231–240.

Obendorf, H., Janneck, M., and Finck, M., (2009). Inter-contextual distributed participatory design. *Scandinavian Journal of Information Systems*, (21:1): 51-76.

Olleros, X., (2008). The lean core in digital platforms. *Technovation*, (28:5): 266–276.

Oostveen, A.-M., and Van den Besselaar, P., (2004). From small scale to large scale user participation: a case study of participatory design in e-government systems. In: *Proceedings of the eighth conference on Participatory design: Artful integration: interweaving media, materials and practices-Volume 1*, A. Clement, F. Cindio, A.-M. Oostveen, D. Schuler, P. Besselaar (eds.), ACM Press, New York, pp. 173–182.

Pilemalm, S., and Timpka, T., (2008). Third generation participatory design in health informatics—Making user participation applicable to large-scale information system projects. *Journal of Biomedical Informatics*, (41:2): 327–339.

Plantin, J.-C., Lagoze, C., Edwards, P. N., and Sandvig, C., (2016). Infrastructure studies meet platform studies in the age of Google and Facebook. *New Media & Society*.

Pollock, N., Williams, R., and D'Adderio, L., (2007). Global Software and Its Provenance: Generification Work in the Production of Organizational Software Packages. *Social Studies of Science,* (37:2): 254–80.

Sahay, S., Sæbø, J., and Braa, J., (2013). Scaling of HIS in a global context: Same, same, but different. *Information and Organization*, (23:4): 294–323.

Sandberg, Å. (ed.), (1979). *Computers dividing man and work*. Swedish Center for Working Life, Demos Project Report no 13, Utbildningsproduktion, Malmö.

Sanner, T. A., Roland, L. K., and Braa, K., (2012). From pilot to scale: Towards an mHealth typology for low-resource contexts. *Health Policy and Technology*, (1:3): 155–164.

Shapiro, D., (2005). Participatory design: the will to succeed. In: *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility,* O. W. Bertelsen, N. O. Bouvin, P. G. Krogh, M. Kyng (eds.), ACM Press, New York, pp. 29–38.

Shapiro, D., (2010). A Modernised Participatory Design? A reply to Kyng. *Scandinavian Journal of Information Systems*, (22:1):69–76.

Shaw, V., and Braa, J., (2014). Approaches to participatory design in Africa in the age of cloud computing. In: *Proceedings of the 13th Participatory Design Conference - Volume 2*, O.S. Iversen, H. Winschiers-Theophilus, V. D'Andrea, C. Bossen, M. Teli, K. Bødker (eds.), ACM Press, New York, pp. 45–48.

Titlestad, O. H., Staring, K., and Braa, J., (2009). Distributed development to enable user participation: Multilevel design in the HISP network. *Scandinavian Journal of Information Systems*, (21:1): 27-50.

Tiwana, A., (2013). *Platform ecosystems: aligning architecture, governance, and strategy,* Morgan Kaufmann, Waltham, MA.

Van Schewick, B., (2012). *Internet architecture and innovation*. MIT Press, Cambridge, MA.

Walsham, G., (1995). The emergence of interpretivism in IS research. *Information Systems Research*, (6:4): 376–394.

Walsham, G., (2006). Doing interpretive research. *European Journal of Information Systems*, (15:3): 320–330.

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

Yoo, Y., Henfridsson, O., and Lyytinen, K., (2010). Research commentary-The new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research*, (21:4): 724–735.