Development as a Free Software: Extending Commons Based Peer Production to the South

Knut Staring
University of Oslo, knutst@ifi.uio.no

Ola Hodne Titlestad
University of Oslo, olati@ifi.uio.no

DEVELOPMENT AS FREE SOFTWARE: EXTENDING COMMONS BASED PEER PRODUCTION TO THE SOUTH

Etendre la production communautaire par les pairs vers les pays du sud

Completed Research Paper

Knut Staring
Dept. of Informatics
University of Oslo
Norway

Postboks 1080 Blindern
0316 Oslo
Norway
knutst@ifi.uio.no

Ola Hodne Titlestad
Dept. of Informatics
University of Oslo
Norway

Postboks 1080 Blindern
0316 Oslo
Norway
olati@ifi.uio.no

Abstract

This paper examines the concept of commons-based peer production (CBPP) in the context of public health information systems in the South. Based on an analysis of the findings from a global network of software development and implementation, an approach to preserve the importance of local user participation in distributed development is presented. Through practical examples, we discuss the applicability of the CBPP model for software production aimed at improving the public health sector in the South, and propose the concept of a "snowflake topology".

Keywords: OSS/FLOSS, action research, participatory design, developing countries

Résumé

Cet article étudie le concept de production communautaire par les pairs (CBPP) dans le contexte des systèmes d'information en santé publique des pays du sud. Il présente, à partir de l'analyse d'un réseau mondial de développement et d'implantation de logiciels, une approche visant à préserver la participation des utilisateurs locaux lors du développement distribué de logiciels.

Introduction

The nascent shared global digital infrastructure is deemed by authors like Weber (2003) to become so pervasive that they warn against further marginalization of countries in the South that are unable to make use of it, either because of lack of local capacity, or through actively being shut out through processes of “property-right imperialism” (May 2006, Weber and Bussell 2005). Researchers have pointed to the potential of the emerging networked information infrastructure to not only lower transaction costs and increase the efficiency of existing economic processes, but to act transformatively on organizational principles (Avgerou 2007, Castells 1996), such as through renewed possibilities for user participation in innovation (Von Hippel 2001). Large scale change is taking place under labels such as global software outsourcing, virtual teams, and global information systems roll-out (Walsham 2008). However, as Walsham points out, the world is far from flat.

On this view, it is rather the case that access to the information infrastructure is fast becoming a crucial instrument to further human development, including education and health care. Sen (1999) makes a strong argument that access to such services are constitutive elements of development, essential for the kinds of capabilities intrinsic to a coherent and full concept of freedom. The terms Open Source and Free Software both designate "Software that comes with source code and a usage license that allows for modification and further redistribution of the source code by any user" (von Krogh et al 2003), the former emphasizing practical advantages of openness, and the latter the ethical aspects of software sharing: "From the perspective of so-called developing countries it is critical that we don’t lose sight of the freedom rationale behind free software. If we take Sen’s approach of development as freedom, then we are obliged to find ways of placing freedom at the centre of our development efforts, including those efforts which refer to software and other information and knowledge structures" (Jolliffe 2006). The expression Free and Open Source Software (FOSS) is meant to be inclusive of all these aspects.

Though not part of the definition, FOSS is associated with a model of organizing software projects as globally distributed, electronically networked communities (Lee and Cole 2003, Ljungberg 2000), with the product and source available for download by anyone on the internet, allowing the distinction between users and developers to be blurred (von Hippel 2001). Technical support is to a large extent provided by other users (peers) on open internet forums or mailing lists (Lakhani and von Hippel 2003). "The success of open source software projects demonstrates empirically that a large and complex system of code can be built, maintained, developed, and extended in a nonproprietary setting where many developers work in a highly parallel, relatively unstructured way and without direct monetary compensation." (Weber 2003, p.1) This mode of production offers an alternative organizational form to the classical dichotomy of firm vs. market, and Benkler (2002) argues that the much publicized characteristics of FOSS and Wikipedia are instances of an even wider phenomenon which he terms Commons Based Peer Production (CBPP), and proposes a model to explain why this phenomenon has been successful in a range of fields.

However, there are considerable obstacles to FOSS in such settings, including widespread copying of proprietary software, a general lack of awareness, and poor international links, making it hard to be part of “an active, global community of like-minded developers” (Heeks 2005). There are implicit infrastructural resources and knowledge required to fully take part in the practices (Lanzara and Morner 2005, Orlikowski 2002). Technological means of communication are crucial for FOSS development, and services offered by project platforms such as sourceforge.net are seen as the sine qua non (Fogel 2005). The underlying need to be familiar with these practices, coupled with requirements for adequate internet access, are factors which contribute to the fact that more than 80% of FOSS developers reside in the North (Gosh et al 2006).

Also, FOSS has so far been predominantly applied to what Fitzgerald (2006) terms horizontal infrastructure software, such as the Apache web server or the OpenOffice.org productivity suite, in contrast to organizational information systems catering to vertical domains like health care, where the knowing to a large extent is embedded in practices of professionals, whose participation in the development is therefore vital. The participatory design (PD) literature is rich with methods to facilitate communication between domain experts and programmers, leading to more context-sensitive information systems (Bjerknes and Bratteteig 1995, Bødker. and Grønbæk 1991). However, the challenge is to make such local efforts combine into more powerful and sustainable collaborative networks, where products and learning can be shared in CBPP fashion.

The objective of this paper is to investigate to what extent the advantages of FOSS approaches can be realized also when developing vertical information systems in the South. To do this, we apply the analytical lens of the CBPP model to examine the findings from an ongoing global action research network, and present an approach to preserve the importance of local user participation in distributed development in resource-constrained conditions.

In the next section, we outline conditions which have lead to the flourishing of CBPP in the North, followed by an outline of the action research methods applied. We then present a project which was explicitly attempted recast according to FOSS principles. The two last sections discuss how our findings lead us to extend the CBPP model where current conditions in marginalized areas .necessitate interventions and extract recommendations for further action both for this case and more generally.

Literature

Benkler and Nissenbaum (2006) characterize commons based peer production as "a socio-economic system of production that is emerging in the digitally networked environment. Facilitated by the technical infrastructure of the Internet, the hallmark of this socio-technical system is collaboration among large groups of individuals, sometimes in the order of tens or even hundreds of thousands, who cooperate effectively to provide information, knowledge or cultural goods without relying on either market pricing or managerial hierarchies to coordinate their common enterprise". Thus, the core characteristics of CBPP consist of I) decentralization, where individuals act as they see fit, without a central organizer, and II) "the use of social cues and motivations, rather than prices and commands".

A key premise of the model is the existence of excess capacity, resulting from a great number of potential contributors and a set of organizational structures: "CBPP succeeds when the peers' contribution is based on their having some excess capacity to contribute to the final artifact. It is this capacity that allows them to contribute without any direct monetary compensation" (Tsiavos 2008). In the case of FOSS projects, the excess capacity also stems from the fact that the source code is used to produce more source code, which in attracting further contributions can trigger a virtuous circle or snowballing effect.

The CBPP model applies when projects can be divided into modules which a) can be independently and incrementally produced, b) can be sufficiently fine-grained to allow the capture small contributions, and c) can be quality-checked and integrated with the overall system through reasonably low cost mechanisms (Benkler and Nissenbaum 2006). This enables the discrete efforts of a large number of differently motivated individuals to enhance an open-ended core provided by a dedicated few. With large and diverse numbers of participants and ample infrastructural tools, quality control can largely be carried out through redundancy and evolutionary selection processes. However, the model seldom works without effort: "CBPP is more the designed result of a carefully instrumented effort rather than the haphazard outcome of a natural selection process. In addition, it almost invariably co-exists or even depends on the classic models of the hierarchy and/or the market" (Tsiavos 2008).

It is evident from the above that the modular makeup of a system heavily influences the kinds of organizational form (division of labor) that are likely to work best. This is a variant of Conway's Law, which says that the (modular) structure of an information system will mirror the structure and communication patterns of the organization designing it (Conway 1968).

In relation to the focus of this paper, the question is to what extent the above characteristics are fulfilled when it comes to public sector information systems. While conditions vary significantly between regions and countries, there are enough commonalities to public administration to make efforts to share a common base sensible in many situations. Pollock et al (2007) describe how generic packages are "brought into being through an intricately managed process, involving the broader extension of a particularized software application and, at the same time, the management of the user community attached to that solution" by suppliers involved in "generification" work. Walsham (2008) asks which users get most to say in such processes and what effects standard technologies will have on diverse organizations. In contrast to the proprietary processes studied by Pollock et al, a CBPP approach would entail a fluid, open, boundary object with some infrastructural characteristics (Darking and Whitley 2007).

Thus, it is of interest to follow FOSS projects over the longer term to study how the implementation experience feeds back and shapes the further development over the life cycle.

One way of addressing negative implications of generification is to draw on the participatory design approach, but to make this sustainable and of sufficient impact, it must be extended beyond the focus on local situations, and emphasize the scope for mutual learning. The approach called Networks of Action (NoA) is targeted at precisely this, and is characterized by" *(1) abandoning singular, one-site (typically one organization) action research projects in favor of a network of sites, (2) generating local, self-sufficient learning processes together with working mechanisms for the distribution of appropriately formatted experiences across sites in the form of vertical and horizontal flows, (3) nurturing a robust, heterogeneous collection of actors likely to pursue distinct, yet sufficiently similar (Callon 1991; Latour 1986), agendas, and (4) aligning interventions with the surrounding configurations of existing institutions, competing projects, and efforts as well as everyday practices.*" (Braa et al. 2004, p.359)

The NoA approach addresses sustainability of information systems in poor countries through establishing a network of mutually supporting sites supportive of local learning processes, and aligning interventions with existing institutions. The basic tenets of the FOSS development model as laid out by Raymond (1998), would seem well suited to such an effort. The aim of this paper is to investigate the interaction of these principles in practice.

Methods

This article draws on experiences from a continuous software development process in a research initiative called the Health Information Systems Programme (HISP) over the last decade (Braa and Hedberg 2002, Braa et al. 2004, Braa et al 2007). HISP is an extensive collaborative network with a focus on information systems for public health care in the South. Initiated by universities in Cape Town and Oslo, it is present in a number of African and Asian countries. Within each country the projects are comprised of various actors in the health administration (community, sub-district, district, provincial, and national), universities, NGOs, and funding providers. At the global level, with the Norwegian and South African nodes as the major coordinating bodies, HISP has over the last decade been engaged in development and implementation of health information systems with emphasis of facilitating sharing of software and best-practices.

This research is based on Action Research, a form of participative research where the researcher takes part in the change processes in an organization, actively trying to improve some stated problem (Checkland and Holwell 1998; Avison, Lau et al. 1999). The case study describes an effort to establish a distributed software development process based on the principles of FOSS, in a network of countries in the South. The thick descriptions in the case are results of critical reflection on the change process which the two authors have been driving forward over the last four years. Application of the CBPP model as an analytical lens has helped us reflect more systematically on the value and appropriateness of the action taken, and thereby informed the proposed model and recommendations for further action.

Both authors have been engaged in the HISP action research project for about five years and been involved in software development, implementation and education activities. Since 2004, we have been the major driving force behind efforts to distribute global software development of DHIS v2, involved in daily coordination. More specifically, this work has involved managing programmers in the Norwegian node, facilitating collaboration between the different nodes, building development capacity in various countries, and making major architectural and design decisions concerning the software.

Given the focus on the distributed FOSS software development process, we emphasize the interactions with the developers including: notes from meetings, mailing list archives, the history of source code repository; collaborative web pages (wiki), and project management and issue tracking tools, as well as logs from extensive use of instant messaging between HISP actors concerning DHIS v2 development and coordination. These data were systematically read through and sorted into clusters around events which were deemed to be important.

Case

This case study builds on a long term action research project called HISP and more specifically describes how the DHIS software is developed and supported in a global collaborative network. First, we give a brief introduction to the DHIS software itself, and then the main part of the case presents the development process in various phases from the initial phase in South Africa in 1996 to the globally distributed process by 2008. In the final subsections we illustrate key challenges of the distributed process as well as interventions that addressed them.

The DHIS software 1996-2008

Table 1- Spread of DHIS use and development

	Use	**Development**
DHIS v1	South Africa (national scale), Botswana, Mozambique (pilot), Cuba (pilot in 2002), India (2000-2005), Zanzibar (national scale), Tanzania (pilot), Zambia (pilot), Malawi (national scale), Nigeria (some states), Myanmar (TB program).	South Africa
DHIS v2	India (4-5 states, since 2006), Vietnam (2 provinces, since 2007), Sierra Leone (4 pilot districts, since 2008, plans for all districts), Tajikistan (pilot, since 2007)	Norway, India, Ethiopia, Vietnam, Mali, Tajikistan

The DHIS is a data collection and visualization tool for statistical data, tailored to support integrated health information management activities. After three initial years of prototyping and pilot implementations, the software version 1 (v1) was adopted as a national standard for South Africa in 1999. This success triggered an expansion of the HISP network internationally (see Table 1). DHIS v1 is a stand-alone application based on MS Access and VB6. It has from the beginning been developed and maintained in a centralized manner by a small team in South Africa. The expansion to very different contexts created a strong push for more flexibility to meet new requirements.

In late 2003, key actors were forwarded an evaluation of the DHIS v1 that had been sent to the Ministry of Health in Mozambique by an unidentified source:

"The DHIS system appears to be a tool mostly intended for small scale scientific research. Its technology is outdated by a decade and design is poor from a conceptual point. As a data gathering system is barely usable by today standards. As a data warehousing system is inadequate. The system should be migrated to SQL server and tiers should be split. WEB enabled user interfaces should be provided. WEB Classes should be constructed to encapsulate the business tier."

This critique speeded up the work towards a complete overhaul and reimplementation, and by mid 2004, a group of Norwegian researchers and students had begun developing a new version, v2, with the following main objectives:

1) Redevelop v.1 on a new platform independent of operating system and database engine

2) Design a layered and modular architecture supporting distributed work

3) Provide an electronic forum with tools and documentation facilitating globally distributed software development, and the sharing of code, documentation and best practices across countries in the network

In line with advice from the main developer in South Africa and influenced by the popularity of Java at Norwegian universities, a set of up-and-coming lightweight FOSS Java frameworks were selected as the new development platform. The distributed development process was supported by set of tools commonly used in open source projects (Fogel 2005): source control management, a wiki for project documentation, an issue tracker for bug reporting, and mailing lists for communication among developers and users.

The first 1.5 years of v2 development was mainly confined to Norway, and focused on establishing a new core platform, with a fairly complex layered structure, based on "best practice" object oriented design patterns, pursued to perfection. Correspondingly, little attention was directed towards developing user interfaces and functionality for data visualization and analysis which were important features of v1 and highly cherished by health sector decision makers.

By the end of 2005, DHIS v2 still contained only the very basic functionality of v1, and had not been tested in the field. Then, the HISP India NGO negotiated a contract to implement DHIS in Kerala, and, unusually, state policies demanded the use an open source platform. This ruled out the use of v1, and provided the decisive push for a first release of v2. After a troublesome first year in 2006, v2 matured through input from use in the field, and was introduced in several other Indian states, Vietnam and Tajikistan in 2007, and in Sierra Leone in 2008. After 2.5 years of use and continuous development, it was finally on par with the functionality of v1 by the summer of 2008. Furthermore, the development team for v2 had expanded from a small Norwegian team in 2004-2006 to include developers and domain experts from a number of countries. Table 1 lists the countries that are or have been involved in DHIS use and development. Version 1 is licensed under a derivative of the GPL and v2 under the BSD license.

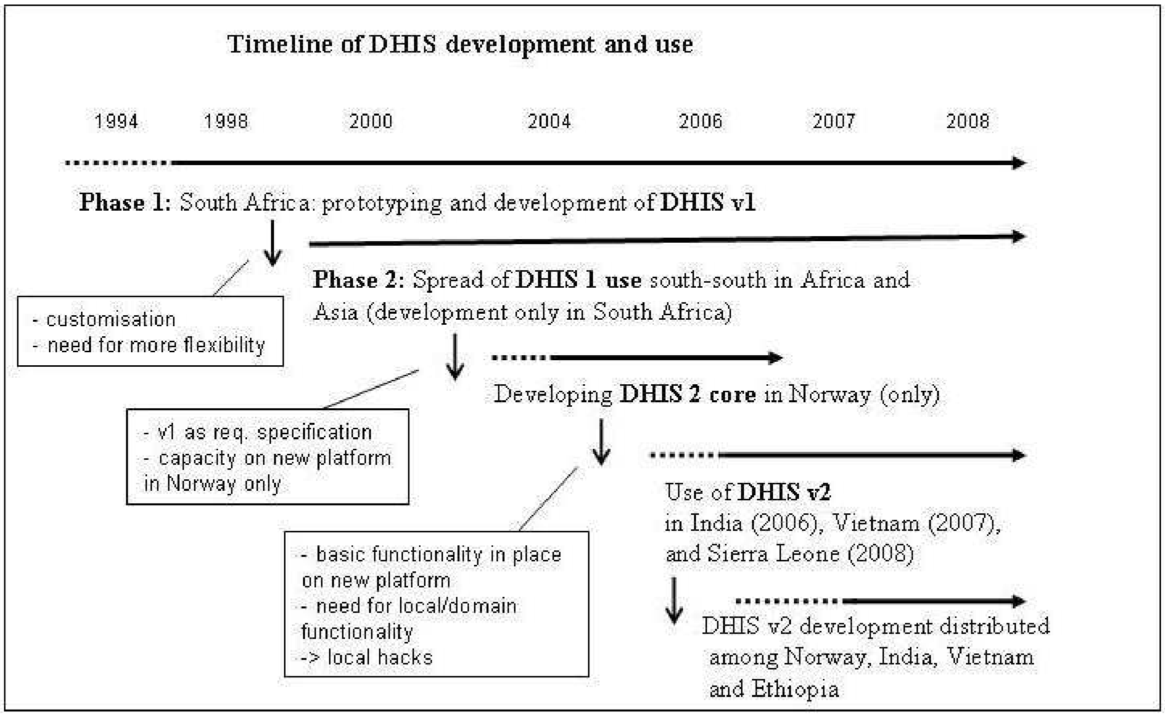
Timeline of DHIS development and use
1994
1998
2000
2004
2006
2007
2008
Phase 1: South Africa: prototyping and development of DHIS v1
- customisation
- need for more flexibility
Phase 2: Spread of DHIS 1 use south-south in Africa and Asia (development only in South Africa)
- v1 as req. specification
- capacity on new platform in Norway only
Developing DHIS 2 core in Norway (only)
- basic functionality in place on new platform
- need for local/domain functionality
-> local hacks
Use of DHIS v2 in India (2006), Vietnam (2007), and Sierra Leone (2008)
DHIS v2 development distributed among Norway, India, Vietnam and Ethiopia

Figure 1 Timeline of DHIS development and use

Centralized software development process

The software development process followed the timeline depicted in Figure 1. The process around DHIS v1 was influenced by core principles of the Scandinavian IS tradition such as user participation, local involvement, rapid prototyping, and bottom-up processes, and involved a small team of developers working closely with users in a few selected districts. Key design concepts were developed based on user interaction and an in-depth understanding of the domain, such as a flexible metadata model allowing for local additions to central standards and complete freedom for local users to modify and tailor the contents of the database, supported through readable tables with textual descriptions as identifiers as opposed to cryptic acronyms and numeric codes. Combined with the targeting of the district level as key, these principles slowly won their way up the health sector hierarchy in a bottom-up process until they became accepted at the national level.

While the use of v1 escalated to include many new countries, the development team remained centralized, and soon became a bottleneck in supporting the expanding network of users with context-specific requirements. The architecture and technology of v1 made it difficult to distribute the development, and no tools were used for sharing and synchronizing the code base. As a consequence, local developers in Ethiopia forked DHIS release 1.3.68 to include some local modules, making it difficult to incorporate improvements and fixes from later releases.

However, despite the challenges of distribution and lack of web support, v1 remained under continuous development in 2008, benefiting from a large user base, as exemplified through an active user mailing list with participation from information officers, health workers and other stakeholders at all levels of the health system. Implementation and customization efforts in a range of countries in the Sub-Saharan region are supported by a dedicated and experienced team of software developers and health information professionals making up the HISP South Africa NGO.

Designing for local involvement in global software development

From 2004, triggered by the critique of v1, the Norwegian team emphasized a more collaborative and distributed development process. The need for a modular architecture was clear, so a "core-periphery" architecture was designed, as illustrated in Figure 1. This model targeted the emerging needs of the global and increasingly heterogeneous HISP network by supporting customized configurations of core functionality that all countries would need, while allowing for a selection of peripheral modules to meet local requirements. Core elements of a HIS such as data collection sets and organizational structures could be set up generically in each context, and basic functionality for capturing and analyzing data was provided through standardized tools. The more advanced peripheral functionality such as tailored reporting, custom data capturing, and special analysis tools were kept as external modules that could be loosely coupled to the standard core to provide an integrated user interface. More specifically, a local team of implementers can use the shared functionality and then develop and plug local modules to assemble a package tailored to the local context, based on clearly defined interfaces.

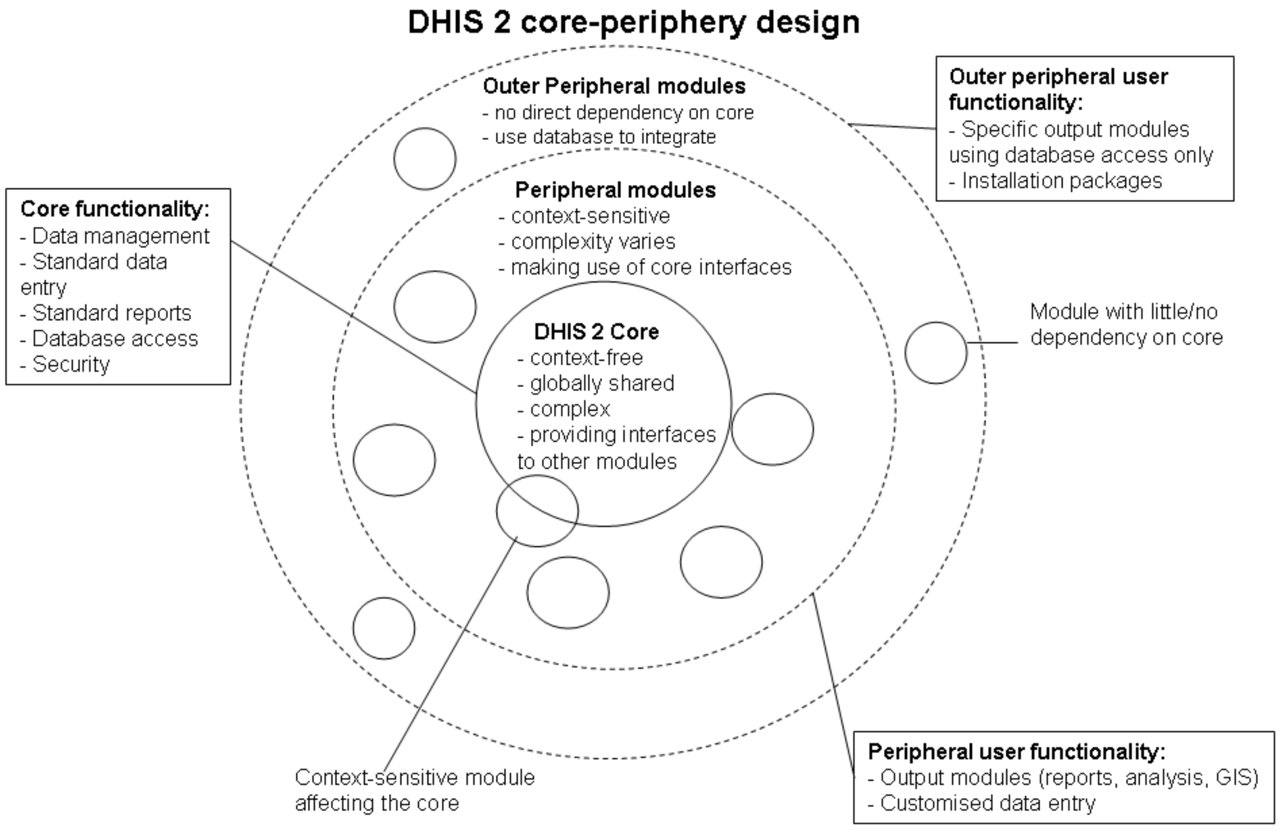
DHIS 2 core-periphery design
Outer Peripheral modules
- no direct dependency on core
- use database to integrate
Outer peripheral user functionality:
- Specific output modules using database access only
- Installation packages
Core functionality:
- Data management
- Standard data entry
- Standard reports
- Database access
- Security
Peripheral modules
- context-sensitive
- complexity varies
- making use of core interfaces
DHIS 2 Core
- context-free
- globally shared
- complex
- providing interfaces to other modules
Module with little/no dependency on core
Context-sensitive module affecting the core
Peripheral user functionality:
- Output modules (reports, analysis, GIS)
- Customised data entry

Figure 2 The modular DHIS v2 core-periphery architecture

Frameworks getting in the way of the data

In February 2006, when the first milestone of DHIS v2 was put to use, the HISP India team members, who had several years of experience with the previous version, were struggling to make v2 useable in the Kerala context. At that time, there were no report tools in place, and poor capacity for processing the large amounts of data collected and reported in the Kerala health system.

The platform independence requirement meant that the database was abstracted away using an Object Relational Mapping framework, and all interaction with the data was designed to take place through an object-oriented application programming interface (API), which itself was set up to use a state-of-the-art "light-weight container" framework (Tate and Gehtland 2004). This entailed extensive XML configuration and understanding of concepts such as "Inversion of Control" (Fowler 2004), but was meant to relieve developers of the burden of detailed database manipulation. However, the abstract conceptualization had the unintended side effect of alienating the Indian team from long established and effective habits of closely handling the data.

The Indian developers had worked with v1, where data are accessible through the means of a human readable database, as well as through a "datamart" in which a read-only data repository with easily accessible pre-processed data for reports and visualizations were generated and decoupled from the active database. This direct access to the database level had formed the basis for many creative reporting solutions tailored to the needs of local users.

The first problem reported from the Indian team was that they could not find their data, as the database model was difficult to understand. Although the meta-data model is quite similar in v1 and v2, the ORM auto-generated table names did not make sense to the local developers. Moreover, v2 did not a have a datamart similar to v1 in place until May 2006, and when it was at last implemented after requests from India, it worked nicely only with the small test datasets used by the core developers in Norway, but performed dismally with the much larger Indian databases:

"Instead of saying any thing about this module [datamart] I can say this entire module is incomplete and not working properly. First point is that till now we couldn't able to export data values even after waiting for one complete night after clicking the button export. Even the calculation of 10 indicators for a year it took more than 7 hours which is not acceptable."

The frustration of the HISP India team was evident, and the developers in Norway, at the time consisting of a couple of skilled master students, were struggling to keep up with the pressure from the field. The core team struggled for a long time to provide an efficient datamart solution, trying several approaches, but a breakthrough did not come until December 2007, when a much faster solution bypassing the whole Java API service was put in place by a Norwegian developer traveling to South Africa to work closely with the v1 developers.

From local hacks to globally shared functionality

The earlier releases of v2 did not have out-of-the-box data visualization modules for easy display of the data collected. Such modules must often be tailored to specific user needs and are difficult to generalize into globally shared modules. Furthermore, understanding the logic of how health care data are processed and presented in reports is more domain specific knowledge than most of core modules, which was more difficult for the Norwegian developers with little field experience to acquire, sitting far from the Indian users. Focusing on the importance of local contributions to the globally shared code base, the Norwegian developers encouraged the Indians to conform to the code standards of v2, to make use of the provided API to fetch data, and to use the chosen web frameworks and portal solution to develop user interfaces for the report modules.

However, the combination of all these foreign concepts and technologies made the learning curve very steep and did not allow the Indian team to deliver to the users within reasonable time frames. As a result, the Indian team managed to bypass the Java application and came up with several local web modules for displaying data that communicated directly with the database (in the outer peripheral category in Figure 1). In this way, they managed to keep credibility among the users. These local hacks were not appreciated by the Norwegian core developer who had implemented most of this complex platform:

"*Using the database directly is duplication of work (all access, aggregation and logic code for values is duplicated) and also means that the reporting tool is put outside of everything we do (and can do) in the DHIS2 in regards to optimisation, caching, security, and so on. [..] This is of course also a question of resources and available time, but keep in mind that we are starting to get quite a few of these hacks now, and someone will eventually have to clean it up...*"

However, the Dashboard module developed in India to graphically analyze data and indicators in charts, quickly became an important instrument for the HISP India coordinators when demonstrating the capabilities of the DHIS to local authorities. For quite some time this module was not part of the global code repository, and it was not compatible or able to plug into the globally shared core module.

The Norwegian coordinators were very eager to incorporate this module into the global code (from an "outer peripheral" to a "peripheral module" in the categorization of Figure 1), as it was seen to constitute an important extension of the functionality of DHIS v2. During a very constructive visit to India by one of the Norwegian developers, he sat spent long hours pair programming with the Indian developer behind the Dashboard module, and modified it to fit the standards of v2, and committed the code to the globally available repository. Shortly after this integration, the dashboard module was used to great effect by one of the authors at meetings with WHO and the Health Metrics Network (HMN) in Geneva, which might result in its utilization in many new countries.

In Ethiopia in 2007, there was one especially important requirement which put the local team in a make or break situation. The health system defined an intricate way of collecting fine grained information for about each disease, using the International Classification of Diseases (ICD), for which there was no support in the global software. A local developer was employed on a short term contract to add this functionality to DHIS v2, and without interaction from the global community managed to absorb the inner workings of the system purely by reading the code (with technical documentation being close to nonexistent). In relatively short time he successfully implemented a module which in many ways crossed into the core. Although successful in the sense that it solved the urgent need in Ethiopia, the solution was not complete, and in many ways a local hack, as it replaced key parts of the core data model. As result, the local implementation of v2 became a fork from the global code.

Still, this module served several purposes; it kept the local implementation process alive by solving an immediate need, it was a useful learning exercise for the developer, and it was the first step towards a very important contribution to the DHIS v2. A few months later, in September 2007, when the Ethiopian developer came to Norway to enroll in the PhD program, he worked closely with the core team to modify his solution to be compatible. At the time, implementation activities in Ethiopia were suspended and therefore the immediate need to go ahead with this integration was gone. The development still moved forward, albeit at a slower pace, and mostly at the design table where developers and coordinators were discussing the difficult issue of how this multidimensional solution could fit the core data model. In December, the Ethiopian developer went to Tajikistan where he was responsible for the customization of v2. Once there, he quickly saw the benefit of applying the multidimensional model for the detailed Tajik report requirements. Under severe time pressure, the multidimensional model was finalized and large parts of the core system re-factored to build on the new model, all in a pace impressing the Norwegian developers.

Then, in the critical phase of finalizing a release of the software to use as a demo for HMN, the Norwegian team found out that the recent re-factoring of the core had several negative unintended consequences. The solution committed by the Ethiopian developer was focused on the needs in Tajikistan and did not take into consideration all the other modules that depended on the parts of the core which he had modified. For a few critical weeks the system was not working properly and was in need of urgent housecleaning by the main Norwegian developer. However, after the system was brought back to a stable revision, the recent additions on the multidimensional model by the Ethiopian developer were utilized to their full potential in implementation also in Sierra Leone.

From developer to user – the value of understanding the field

In 2008, the Norwegian team finally got directly involved in DHIS v2 use as the Norwegian HISP group started to prepare a customized DHIS v2 for the above mentioned project with HMN in Sierra Leone (SL). At the time the Norwegian group was strengthened as a former master student had now graduated and become a full time employee in the project. The customization for SL was organized as a collaborative effort between a software company in Mali, and the team of coordinators and developers in Norway, as well a related open source project, the OpenMRS (Seebregts et al 2007). A lot of the preparations could be done in Norway, where one of the coordinators had extensive experience from v1 implementations. This process was very useful for the Norwegian team, as many issues with the software that had so far not been communicated, especially related to usability, were discovered and quickly fixed by developers located in the same office.

One of the specific requirements from SL was to design a serious of complex reports visualizing data. Through this work, a Norwegian developer put in place a generic reporting framework which enabled the new and faster datamart processing as well as a third party easy-to-use report designer. These developments were further improved as the developer traveled to SL and worked with the local implementation team, providing his first real exposure to the use of DHIS v2. This interaction was very dynamic, as can be discerned from weekly and even daily updates of the code base, quickly incorporating bug fixes based on feedback reports from the Mali team.

The report framework developed as part of this process was a major step forward for a sharable report solution, as most report design solutions so far had been local hacks that were difficult to generalize into a global module. During an international developer workshop in Vietnam in August 2008 developers from India, Vietnam, Ethiopia and Norway worked together for two weeks with the major objective to align programming efforts and to minimize forking and duplication of work. The new report framework was introduced and quickly got popular among the other teams that had been struggling with local report solutions that were more difficult to use and were considerably slower as they did not take advantage of the newly released and faster data processing solution. A major effort was made to extend the report framework to support all report requirements from India and Vietnam in addition to the SL requirements it had been based on which resulted in an improved global report solution that was immediately put in use in Vietnam and is planned to replace the existing report solutions in India. While extending this module to incorporate new requirements from the other countries the functionality tended to grow complex as too many use cases were squeezed into the same user interface and now there is some important work remaining on improving the usability aspects of the module.

Communication and use of collaborative platform in the South

Analyzing the use of the FOSS collaborative platform set up for v2 development, there was a strong dominance by the Norwegian team when it comes to contributing to the code repository, writing on the mailing list and on sharing documentation on the wiki (Staring and Titlestad 2006). It was very difficult to establish an active contribution to these public communication channels in the South, and e.g. in India with the far biggest user base, the implementation team has never contributed to the wiki, seldom written to the mailing list, and provided little regular feedback to the global team. This situation led to the following message from one of the coordinators in Norway in June 2006:

"*I certainly recognize the severely limited capacity on the part of coordinator X and myself vis a vis coordinating the expanding plethora of initiatives. Also, the beauty of a loosely coupled network should really be the avoidance of having coordinating bottlenecks. Still, communication is vital (as far as possible on public mailing lists); otherwise, there is in fact no network.*"

The following responses from the senior HISP coordinator illustrate a very difficult perception of what the collaborative network is, as well as the problematic consequences of imposing the technological platform for collaboration on the South:

"*i cant see how there is no network - only if you see the internet as the network..there is a thriving network with vietnam, india, ethiopia, and tanzania etc have seen it and want to leverage it and contribute to it...let us not see the world from one point of center and one kind of network - because it is not, and it never will be*"

"*the demands of implementation make it too difficult to use the internet for communication..for example, the ART guys are working day and night for making a presentation to the health minister...how can they do this, their course work, in addition to dealing with a damn slow internet...not saying they should not communicate = but in the scheme of things the priority changes as compared to the perspective from oslo*"

In an effort to establish v2 development as part of the coursework at the University of Dar es Salaam in 2008, student groups had to draw on the global v2 community, and were strongly encouraged to make use of the online collaboration tools. This proved difficult to achieve, and little communication took place apart from a few private e-mails sent from the students to the coordinators in Norway, as illustrated in the chat log below:

11:05 AM <coordinator in Norway>: how are the other groups doing? its very silent...and little use of the wiki...

11:07 AM <student in Tanzania>: yeah

but we are all trying to make things work

and we are not very much used in that direction of communication

I guess we have to adopt to it

Relying on online collaboration often proved very ineffective, as illustrated during Vietnam workshop in 2008, when the first day of presenting local work revealed a lot of duplication that had not been communicated on the mailing lists or to the coordinators in Norway. Among the more serious duplications were three different datamart solutions, meaning that both India and Vietnam were forking the global code, and were not able to benefit from the most recent improvements, which had taking place as a result of the prioritized Sierra Leone implementation. In fact, in Vietnam they were using a fork that had a nine-month old core module, meaning that a long range of improvements to the global code base was not being utilized. Demonstrating the latest version of the global system made it easier for the Norwegian coordinators to convince the Indian and Vietnamese team that moving local forks into the global code was a win-win for all parties.

Long term capacity building approach in HISP

Recruiting developers has been a major obstacle to distributing v2 development across the nodes in the network. The technology stack of the v2, included cutting edge frameworks, and when establishing local development teams in Vietnam and India, it proved difficult to find developers with relevant experience. Among the concrete efforts to strengthen the development teams were frequent field visits by Norwegian developers, and international workshops. Enrollment of developers and implementers from the field into international Master and PhD programs coordinated by the University of Oslo provided a more long term perspective. Face-to-face meetings and co-development among developers from different countries proved key to enhancing the globally shared software (the core), exemplified by the standardization of the dashboard and multidimensional modules.

In Norway, most of the core developers have been master students contributing their time to the project as part of their work for the master's thesis. Moreover, a practical master level course in open source software development has been conducted since 2005 which provides the students with an introduction to the relevant technologies, as well as practical experience from participating in the v2 development project. Although the course has been an important recruiting base for future v2 developers, the output from the student projects in the course have been of less value to the DHIS software, illustrating some of the challenges of mastering the technologies. The approach of recruiting local champions to long term study programs has created an inspiring and multinational environment in Oslo by 2008, with students from Mali, Tajikistan, Ethiopia, Nigeria and Mozambique all participating in v2 development.

Discussion

"Think of a snowflake. Under the right conditions, water will crystallize into a unique shape. Take that environment away and you have water or ice. Each flake/crystal is unique and beautiful"[1]

We proceed to analyze our case based on the conditions outlined in the CBPP model, and use the key characteristics of the DHIS trajectory to develop a new organizational structure.

Excess capacity

In horizontal FOSS projects, the developers are also users - anyone with the prerequisite technical skills can make contributions after a perusal of the code, because of the highly shared conceptualization of the purpose of software. With a large user base, it is likely that a critical mass will be able to contribute to the development effort. However, in the DHIS case, the health workers are not developers, and the base of potential contributors knowledgeable of the domain is therefore dramatically diminished. Furthermore, the number of developers skilled in advanced Java web programming is very limited, and even with a good technical background, programmers need extensive interaction with health specialists in the context of system use before they are able to be productive, as illustrated by the dramatic improvement of the code through the Norwegian team's hands-on work in Sierra Leone. This contrasts starkly to how FOSS developers often only have virtual contact (if any) with users.

The pressures of the local implementations and the slow internet meant that the tools and processes put in place by the Norwegian team were more of a burden than a help to the other teams, thus the communication costs were not lowered much at all. This severely hampered the creation of the short feedback cycles that both characterize many successful FOSS and projects and is the cornerstone of the PD approach to IS. However, the v1 mailing list does, represent a first step in the direction of user-to-user assistance.

[1]From a comment at the One Laptop Per Child News site: http://www.olpcnews.com/use_cases/education/an_answer_to_question_22.html

Local requirements in Ethiopia eventually lead to substantial changes in the core of the system to accommodate very generic multidimensionality. This initially caused major instability, but ended up enhancing the core significantly, expanding the overall capacity of the system to incorporate a range of new functionalities, which were immediately applied to deal with quite different requirements in Tajikistan and Sierra Leone. Thus the generification also proved generative, with the new product serving as raw material for a number of new adoptions. It furthermore facilitated collaboration with other projects, which again triggered further adoptions. In this way the network expanded in a snowballing manner.

Modularity

An explicit goal of the transition from v1 to v2 was to significantly increase the modularity of the system, and this was achieved. The creation of the core module and the API allowed developers to work on isolated parts of the system, and relieved them of creating tedious "plumbing code" for database interaction. In addition, the new approach separated the logic of the system from both the database and the web interface.

However, this separation introduced a steep initial learning curve into a more abstract world of advanced layered frameworks - thus the modular architecture paradoxically introduced more rigidity rather than flexibility, and became something the local hubs had to work around in order to respond to user needs. From this perspective, the Indian "hacks" on reporting and dashboard analysis can be seen as modular approaches to deal with urgent user demands, enabling the project to survive.

Heterogeneous granularity

The frameworks deployed in combination with the new web based collaborative tools and process amounted to a de facto barrier to less experienced developers. It was thus hard to make very small contributions. This was reflected in the fact that very few of the projects carried out by student groups were ever deemed good enough for inclusion in the core. The online collaborative tools in principle made it possible for users and developers of different skill sets and in various positions to participate with fine grained input, such as translations, bug reports, and suggestions for new functionality. However, inexperience with such tools and the generally poor internet access proved a hindrance to active participation. In practice, rather than a flat, peer-to-peer structure, the experience from India and Vietnam display a more hierarchical pattern, where user feedback and discussion mostly occurred with the local implementers who then gradually (if ever) relayed the information to the global network. In contrast, in the Sierra Leone case, the core developer team was in direct contact with the user setting, which shortened the time delay for user feedback to be incorporated in the code (even on a daily basis).

Integration/coordination

The limited use of the open collaborative platform meant that the flow of communication to a large extent was based on personal networks. This resulted in duplication of efforts and parallel solutions. Unrelenting local demands (“the ART guys working day and night for making a presentation to the Health Minister”), combined with both difficulties in learning about the abstract core API, and its immaturity and sluggish performance, meant that local modules grew into isolated “forks”. Also, the pressure to deliver meant that the self-selection of tasks that are suitable to individual skills found in the CBPP model was by and large not an option. This in turn meant that the need for training increased.

If a module is part of the global repository, the original author will no longer be solely responsible for keeping it up to date, and others may improve on it. Still, local developers did not trust the cost of integration to outweigh potential benefits, and thus, for a long time, integration only took place when people moved in the network, e.g. when developers visited other hubs for training as well as during and after global workshops. Orlikowski (2002) points out that even in distributed projects participants typically establish trust by interacting face-to-face. From a technical point of view, it was only after the performance of the datamart improved dramatically, and demonstrated through the Sierra Leone implementation that the integration costs were perceived to have reached an acceptable level where most of the hubs came aboard.

Several parallel solutions are a common phenomenon in large FOSS projects, in the spirit of letting a hundred flowers blossom with “natural selection” of the best solution based on what users appreciate. In the case of DHIS v2, this can be interpreted both as an affirmation of the local flexibility sought by the modular approach, as well as an iterative process of participatory prototyping taking place in different contexts in parallel, ending up in a robust generic solution. But this meant that too much of the limited available developer resources were tied up in the maintenance of forked systems. In the case of Vietnam, the users missed out on a large number of improvements to the global code base. So, while some local needs where met, each process was weak and the potential of a coordinated effort unrealized, with a high deferred cost of integration.

Cultivating snowflakes

The Networks of Action approach of building on existing networks and institutions provides a strategy of bootstrapping necessary conditions for CBPP which are otherwise absent in resource-constrained contexts such as the public sector in general, and in developing countries in particular. A bold strategy to deal with this situation must entail long term investment in capacity building by bringing developers into a network of mentors and collaborators, striving to communicate online, though this will seldom be effective without at least cursory face to face interaction. Therefore, the network of people traveling to collaborate will remain the most salient feature of South – South cooperation

Based on the DHIS experience, the most viable option seems to draw in advanced students or fresh graduates who are motivated by the prospect of working on a globally distributed long term project where they can have a lot of influence. This leads to a suggestion of tying up with and extending existing master and PhD programs. In addition to the theoretical content, the academic programs should entail learning from active participation in local implementations, and simultaneously in the global network. Experience from the field can also form the basis of academic curricula, as in the case of the new course in open source frameworks which was conceived simultaneously to fulfill a need for students to be conversant with modern Java frameworks as well as form a recruiting base for graduate thesis projects. Another important consideration is collaboration with related FOSS initiatives, such as the OpenMRS project in the case of DHIS v2, allowing the creation of a larger shared talent pool.

PD approaches and mutual learning remain crucial to interweave technical, domain specific and situated knowledge. Modular architecture, extensive communication, capacity-building, traveling and physical meetings are important instruments to make PD happen in a distributed environment. At the same time, both NoA and CBPP advocate sharing across contexts and this adds the complexity of variation in the local implementation sites. In contrast to the slow and theoretical start of the DHIS v2 effort, we recommend early emphasis on the production of a functional prototype to quickly get user interaction going (as was done with v1) This is very much in line with the FOSS admonition to "Release early. Release often. And listen to your customers" (Raymond 1998).

The case illustrates various mechanisms for generification as modules and developers move from the periphery (the specific) to the core (the generic); traveling and pair programming (core + peripheral developers), workshops, bring peripheral developers to Norway and sending the core team into the field to work with users. In line with Conway's law, the organizational structure behind such processes largely mirrors the development and trajectory of modules, and the roles of the nodes change over time. In contrast to the highly centralized "Star" topology seen in the development of the DHIS v1 and the radically distributed flat structure of the CBPP model, we propose that resource-constrained contexts call for more "chunkiness".

A "Snowflake" topology entails a number of strong regional hubs around a central repository and a core set of standards. The hubs collaborate, and in turn support smaller peripheral nodes. They filter the input from the field, taking pressure off from the center, and are in a better position to understand local concerns and interact in line with local languages and sensibilities. There is then less need for a dominant core, which can then rather focus on cultivating hubs and on building capacity towards more automated coordination and integration.

Such an organization seems appropriate at least in an initial phase in cases where the core functionality is relatively context independent, but technically challenging. It is important to facilitate the creation of user focused modules at an early stage and facilitate frequent two-way travel between the hub nodes. This structure is innovative in the sense that it provides a bridge between the focus on involvement in the local user context extolled by the participatory design tradition and Benkler's distributed peer production perspective, balancing global and local considerations, even in cases where developers are not end users.

Furthermore, the case presented here shows that modules can have dynamic and sometimes fuzzy boundaries – often following a trajectory from the periphery to the core, through a process of generification. Modules and developers move together from the periphery to the center – programmers bring modules with them as they move (or vice versa). It is very hard for a module created at the periphery to hit the bull's eye at first attempt. Many will stabilize as part of the inner peripheral circle, whereas a select few will gain access to the core, possibly delivering a major upset to the whole project, but also triggering new potentialities.

The full set of functionalities required by individual nodes will largely be overlapping, but often not at the same times or in the same sequence. Left to fend for themselves, they may thus end up with radically deviating solutions. Rather than leaving weak local nodes to develop their own solutions in isolation, a snowflake structure enables a (partial) distribution of the integration efforts needed to prevent extensive forking between strong hubs, thus enabling increased generification. Also, participation in the global network means that many more nodes have the potential to grow into influential hubs, increasing their capacity to serve local users.

Conclusion

Free and Open Source Software provides substantial opportunities for learning - but exploiting this potential in itself requires knowledge and resources. Therefore, sustainable FOSS projects in the South require a strong training components rooted in interaction with the system in active use, as elaborated by the Networks of Action approach.

The case presented in this paper underscores the need for long term organizational networks between institutions in the South to support the collaborations enabled by CBPP. Strong emphasis must be put on the circulation of both people and software, to achieve the right mix of generification and localization. The Snowflake topology outlined suggests a pragmatic and distributed approach to meeting the challenges of local development while drawing on a pool of shared resources.

The more of these local improvements that can be shared around the network, as generic solutions, the stronger the Networks of Action. Whereas the CBPP model seems to predict that given the right conditions, highly complex products will "emerge naturally", the analysis of the current case indicates that careful long-term cultivation of existing structures is crucial to achieve the critical mass of functionality and users to enable a self-sustaining snowflake structure.

References

Avgerou C. "Information Systems in Developing Countries: a Critical Research Review," London School of Economics and Political Sciences, Innovation Group, Working Papers Series, 2007.

Avison, D., F. Lau, et al. "Action Research." Communications of the ACM 42(1), 1999, pp. 94-97.

Benkler, Y. "Coase's penguin, or, Linux and The Nature of the Firm," The Yale Law Journal, 112(3) 2002.

Benkler, Y. and Nissenbaum, H. "Commons-based peer production and virtue," The Journal of Political Philosophy, 14(4) 2006, pp. 394–419.

Bjerknes G. and Bratteteig T. "User Participation and Democracy: A Discussion of Scandinavian Research on System Development" in "Scandinavian Journal of Information Systems" Vol.7 (1) 1995, pp. 73-98.

Braa, J., and Hedberg, C. "The Struggle for District-Based Health Information Systems in South Africa," The Information Society (18:2), 2002, pp. 113-127.

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

Braa J, Monteiro E, Sahay S, Staring K, and Titlestad, O. "Scaling Up Local Learning - Experiences From South-South-North Networks Of Shared Software Development," in Proceedings of the 9th International Conference on Social Implications of Computers in Developing Countries, São Paulo, Brazil, May 2007.

Bødker S. and Grønbæk K. "Design in Action: From Prototyping by Demonstration to Cooperative Prototyping" in J. Greenbaum and M. Kyng (eds.) "Design at work: Cooperative design of computer systems", Lawrence Erlbaum Associates 1991, pp. 1-24 .

Castells, M. "The Rise of the Network Society, The Information Age, Economy, Society and Culture," Blackwell Publishers, Oxford,1996

Checkland, P. and S. Holwell "Action Research: Its Nature and Validity." Systemic Practice and Action Research 11(1), 1998, pp. 9- 21.

Conway, M.E. "How Do Committees Invent?" Datamation, Vol. 14, NO. 4, 1968, pp. 28-31.

Darking and Whitley "Towards an Understanding of FLOSS: Infrastructures, Materiality and the Digital Business Ecosystem", Science Studies, Vol. 20 (2007) No. 2, pp. 13-33

Fitzgerald, B. "The transformation of Open Source Software," MIS Quarterly (30:4) 2006.

Fogel K. F. "Producing open source software," Sebastopol, CA: O'Reilly 2005.

Ghosh, R.A. et al. "The Impact of Free/Libre/Open Source Software on innovation and competitiveness of the European Union", 2006. Retrieved on August 12, 2007 from http://ec.europa.eu/enterprise/ict/policy/ doc/2006-11-20-flossimpact.pdf

Heeks, R. "Free and Open Source Software: A Blind Alley for Developing Countries?", *DIG eDevelopment Briefings*, 1, 2005. Manchester, Institute for Development Policy and Management.

Joliffe, B. "Aligning the ideals of free software and free knowledge with the South African Freedom Charter," First Monday, Volume 11, number 7 2006.

Lanzara GF, Morner M. "Artefacts rule! How organizing happens in open-sourcesoftware projects." In Actor-Network Theory and Organizing, Czarniawska B, Hernes T(eds). Copenhagen Business School Press: Copenhagen; 2005, pp. 67-90

Lakhani, K., von Hippel, E.. "How Open Source Software Works: 'Free' User-to-User Assistance." Research Policy. 32:6,.2003, pp. 923–43

Lee, G. K. and Cole, R. E. "From a firm-based to a community-based model of knowledge creation: the case of Linux kernel development", Organization Science, (14:6), 2003, pp. 633-649.

Ljungberg, J. "Open source movements as a model for organising," European Journal of Information Systems, 9 (4), 2000.

May, C. (2006) 'Escaping TRIPs Trap', Political Studies, 54 (1), pp. 123-146.

Orlikowski, W.U. "Knowing in practice: Enacting a Collective Capability in Distributed Organizing," Organization Science, Vol. 13, No. 3, May-June 2002, pp. 249-273.

Pollock, N., R. Williams and L. D'Adderio "Global software& its provenance: generification work in the production of organisational software packages." Social Studies of Science 37(2) 2007, pp. 254-280.

Raymond ES: "The cathedral and the bazaar,"*First Monday* 1998, 3(3)

Seebregts CJ, Mamlin BW, Biondich PG, Fraser H, Wolfe BA, et al. "The OpenMRS Implementers Network," Health Informatics in Africa (HELINA), January 2007.

Sen, A. "Development as Freedom," New York: Alfred A. Knopf. 1999.

Staring and Titlestad "Networks of open source health care action," Open Source Systems, IFIP Working Group 2.13 Foundation on Open Source Software, June 8-10, 2006, Como, Italy. IFIP 203 Springer, ISBN 0-387-34225-7, p.135-141.

Staring, K., Titlestad, O., Gailis, J. "Educational Transformation through Open Source Approaches," Proceedings of the 28th Information Systems Research Seminar in Scandinavia (IRIS 28) 2005, Agder University College, Kristiansand, Norway.

Tsiavos P, "Towards a model of Commons Based Peer Regulatory Production: the Creative Commons Case" COMMUNIA 2008, 1st International Conference on the Public Domain in the Digital Age, Louvain La Neuve, Belgium, 2008.

Tate, B., and Gehtland, J. "Better, Faster, Lighter Java", O'Reilly, 2004.

von Hippel E., "Innovation by user communities: learning from open-source software," Sloan Management Review 42 4 2001, pp. 82–86.

von Krogh, G., S. Spaeth and K.R. Lakhani "Community, joining, and specialization in open source software innovation: a case study," Research Policy 32 2003 (7), pp. 1217–1241.

Walsham, G. "ICTs and Global Working in a non-flat World," in IFIP International Federation for Information Processing, Volume 267, Information Technology in the Service Economy: Challenges and Possibilities for the 21st Century, eds. Barrett, M., Davidson, E., Middleton, C., DeGross, J. (Boston: Springer), 2008, pp. 13-25.

Weber, S. "Open source software in developing economies." 2003. Retrieved August 12 2008, from http://www.ssrc.org/programs/itic/publications/ITST_materials/webernote2.pdf

Weber, S., Bussell, J. "Will Information Technology Reshape the North-South Asymmetry of Power in the Global Political Economy?" Studies in Comparative International Development 40, No. 2, 2005, pp. 62–84.

Weerawarana, S. and Weeratunga, J. "Open Source in Developing Countries," Stockholm: Swedish International Development Cooperation Agency, 2004.