UNIVERSITY OF OSLO
Department of informatics

Global Software Development: The challenge of communication models

Master thesis
60 credits

Hans Størk Tømmerholt

6. August 2007

Abstract

This thesis looks at various communication models in global software development, how the affect the development process. Four models are explored, a hierarchical model, a network model, a community model inspired by Free/Libre Open Source Software and a composite model. The discussion is informed by a case study of one globally distributed development project, the District Health Information Software version 2. This project has development nodes in Norway, Vietnam, India and Ethiopia. The material is based on the analysis of data from a mailing list and commits to a source code repository used in the project and a series of interviews with participants. The author has also participated actively in the project. The analysis shows that participation is very skewed with the Norwegians dominating the communication and source code production. Some implications are suggest, mainly reducing "gaps of understanding" between the participants, including language issues, contextual gaps and problems in understanding the application and its tools and frameworks.

Foreword

The work with this thesis has been an interesting, engaging and at times frustrating journey. First, I'd like to thank my supervisor Knut Staring for valuable feedback, great discussions and an endless flow of literature. A big thanks to Jørn Braa, Ola Titlestad and Gianluca Miscione for truly last minute feedback and discussions.

My fellow master students and co-developers on DHIS 2 deserve a big thanks for constant input and making the last two years interesting and fun. I'll never forget the constant bantering and whiteboard usage in the office.

I'd also like to thank my informants for putting up with my questions and bringing interesting perspectives on the case. I can only hope I have done them justice.

Finally, I would like to thank Stine for her unyielding support and patience in a hectic period. I couldn't have done it without you.

Contents

Introduction

Software development is becoming more and more global with teams spread out across different sites or different countries (Hersleb & Moitra 2001:1). As a result of the distributed nature of such global software development (GSD), the work *"is done primarily in electronic spaces created through the use of information and communication technologies (ICTs) like videoconference and e-mail"* (Sahay 2003:4). This form of communication lines can be organized in different ways, for example in outsourcing, where one team or company defines the requirements, and another team codes the system. Other ways include having development teams in different locations. The teams can communicate through specific individuals, for example managers or coordinators, or directly between developers. Different tools can be used, like e-mail, mailing lists, video conferencing, and many others.

In this thesis I investigate how different ways of organizing the communication between participants in global software development projects affect the development process and the participation in the communication. I look for different kinds of communication models, which may give rise to advantages and challenges with regard to development. In the thesis I use the term communication model to describe the flow of communication among participants in a project, the norms guiding the communication and the electronic tools used to communicate. The concept is developed by examining the case project and then applied to global software development in general. This leads to the following research objective:

> **Research objective:** Explore how different communication models may be constructed and their effects on global software development projects.

One specialization of global software development is Free/Libre Open Source Software (FLOSS) development where participants take part in a community of developers. This kind of development takes many forms, but has many similarities with regard to their communication practices and use of communication tools. Code repositories, mailing lists, wikis and the like are in widespread use. FLOSS development usually arise from scratching and itch, user-developers coming together to solve their own problems in a loosely organized fashion. But what if FLOSS processes are applied to a more formalized project with distributed development teams? Which challenges and advantages arise from such an application? How does the FLOSS model compare to, or possibly conflict with other forms of GSD organizations? This leads to the following research question:

> **Primary research question:** How does a FLOSS communication model affect global software development projects and how does this model relate to other communication models?

In order to answer this question I draw on data from a global software development project aimed at producing health information software to developing countries. The communication practices of the host organization are explored, along with the experiences with a FLOSS-like development model. One part of the data comes from analyzing a developer mailing list and the logs from the source code repository of the project. These data are compared with qualitative data from an action research approach in the project, as well as from interviews with participants.

As communication in global software development primarily takes place in electronic spaces, we need to look at the tools used for communication. How tools are or are not used, and who are using them (or not), and why are important questions for understanding how well the FLOSS communication model works. Researchers point to at least three levels of challenges: The technical, the cultural and the social (Hersleb & Moitra 2001:1). This leads to my secondary research question:

> **Secondary research question:** How are electronic communication tools used in globally distributed development?

This includes who uses them, to which extent, why and why not and for what. In order to answer this question I analyze in detail how such tools as mailing lists, source code management systems, wikis and issue trackers are used in a case of GSD FLOSS development. The usage is analyzed both in quantitative terms and in terms of user perspectives gained from interviews.

I limit my research to development of source code, meaning the activity of producing software code and the communication and organizing which enable this production. Central to this definition of development is the idea of articulation work as discussed in an article by Schmidt and Bannon:

> *"Articulation consists of all the tasks involved in assembling, scheduling, monitoring and coordinating all of the steps necessary to complete a production task."* (Gerson & Star 1986:266, in Schmidth & Bannon 1992:13)

The process of writing source code is embedded in a myriad other activities such as eliciting requirements, coordinating who works on what, discussing the organization of the project and the use of tools.

1.1 The Case: District Health Information Software 2.0

In order to investigate the questions outline above, I do a case study on the development of the District Health Information Software (DHIS). DHIS is produced by the Health Information Systems Programme, known as HISP. HISP is a large scale action research project aimed at producing health information software for developing countries. This thesis focuses on the development of the second version of this software, called DHIS 2. DHIS 2 is developed as a fully open source web-based Java application, using open source tools, frameworks and libraries. Development is distributed among four countries, referred to as nodes, with participants in Norway, Vietnam, India and Ethiopia. In the thesis I concentrate on the first three nodes as I have limited data about Ethiopia.

The project was set up from the start with a set of communication tools commonly used in typical FLOSS development projects: A mailing list, code repository, a wiki and an issue tracker. FLOSS principles and practices have played a role since the start of the project. Even so, DHIS 2 is also developed in the context of the overall HISP network, which has a long history and its own practices with regard to communication and tool use. As such, a FLOSS-like project is done inside an organization, and not with primarily voluntary developers from around the world as is the case in many FLOSS projects. This means that the case stands out, both as an atypical global development project and as a FLOSS project.

1.2 Motivation for the thesis

It is my hope that this study can contribute to the understanding of how to facilitate electronic collaboration in a globally distributed project. Through the case, I aim to describe some problems that may occur and hopefully outline some ways of addressing them.

My motivation for writing this thesis stems from personal observations made over two years as a member of the DHIS 2 development team. During this time I noticed that there was a strong asymmetry with regard to using various tools for electronic collaboration, like mailing lists and the like. The asymmetry seemed to exist between developers from Norway and developers from the rest of the nodes in the project. This asymmetry has both intrigued and frustrated me. Was it because of language skills? Lack of competence? Time and workload issues? Cultural differences? Laziness?

At the same time I observed the internal workings of the HISP network which were also puzzling at times. People were communicating back and forth in a loose fashion. There appeared to be an extreme lack of coordination in the network itself. This would surely mean that it would break down? However, the network has persisted over a decade. What was also apparent was that the DHIS 2 project seemed to follow a different form of communication, heavily inspired by communication models in open source development projects, using public mailing lists and sharing as much information as possible. This seemed to sometimes put them at odds with the more ad hoc and person-to-person oriented communication methods of the HISP network.

1.3 Communication models

The idea of a communication model as an abstraction for how participants in a global software development project communication, developed originally as a set of drawings to illustrate the conflicting ways which seemed to exist in the case. From these drawings, I tried to develop a concept to help me analyze and abstract the material and point to some more general implications.

In this thesis a communication model consists of three things: First, it has a set of communication pathways, illustration who are talking to whom and in which direction. Second, there is a set of norms guiding the communication, for example "always keep discussions public" or "check outgoing communication with your boss". Third, the participants employ a set of electronic tools for their communication, for example video conferencing or mailing lists. I include the electronic tools because of their significance in globally distributed development. The norms can be embedded in the tools and pathways, for example using a mailing list with public archives in a FLOSS project.

First of all, I want to identify a baseline to help clarify the idea of a communication model. Classical software development projects undertaken by large organizations are typically organized in a hierarchical way. The development task is split up in recursively smaller and smaller tasks and divided out among developers, which reduces the need for direct communication (Weber 2004:60). Well specified code interfaces also help reduce the interdependency between developers. Customers define the requirements which are communicated to development teams through its managers. As an ideal type, this can be illustrated as the typical hierarchical bureaucracy as spelled out by Max Weber (Weber 1978). Orders flow down the chain of command, and questions, if present, flow upwards.

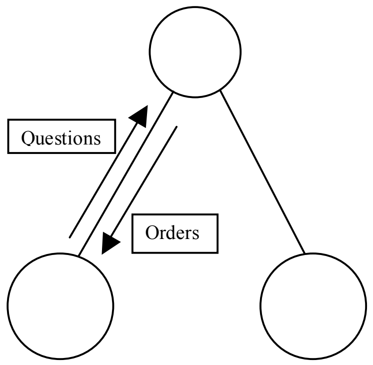


Figure 2.1: Hierarchical communication model

Developers or groups of developers do not talk to each other directly, but coordinate through the design of the system which has been defined by someone else. Effectively, developers communicate through managers or system designers. The tools involved are then the *documents* used to describe the system, but can also other tools such as e-mail, telephone and so on.

These models are ideal types. It’s unlikely that one will find projects where all communication follows patterns exactly like or only like the models described in this thesis. For example, in FLOSS communication, there will always be private communication in addition to public communication on mailing lists and the like. Similarly, there will be other lines of communication in a hierarchical organization besides system design specifications and communication with the boss. Rather, it is my hope that these models can be used to illustrate and highlight some of the principal differences in communication patterns and help point to implications for project organization and potential issues.

Based on the case, two additional communication models are introduced in this thesis: The FLOSS model emphasizes a public, all-to-all communication in a community of developers, while the network model point to some-to-some or few-to-few communication through e-mail.

1.4 Overview of the thesis

The thesis is organized in the following way: Chapter 2 supplies an overview of the current literature on Global Software Development, FLOSS development and some theoretical frameworks used in the thesis.

In Chapter 4 I discuss the methods I chose and make some observations on the data which was gathered throughout the work with the thesis. The thesis is based on a method triangulation approach, combining quantitative data from for example mailing list traffic and qualitative data from interviews. The research was done in an action research frame, as I have been an active participant in the project.

Chapter 4 gives some background on the HISP project and DHIS 2 development, focusing on the different nodes involved in development. A short introduction to the technologies used in development is also given and an overview of the tools used in the project is supplied.

Chapter 5 presents the empirical material used in the thesis, focusing on analyzing traffic on a developer mailing list and in a source code repository. I analyze the data from different angles, from grouping and counting, to thread and content analysis. I also present some of my own role as a participant in the project.

In Chapter 6 I discuss the findings in relation to the research questions.
The thesis ends with some concluding remarks based on the discussion and with some suggestions for further research.

2 Literature and theory

Two strains of literature stand out as particularly relevant to the topic of this thesis. First, the field of Global Software Development, which focuses on the benefits and challenges of highly geographically distributed software work. Second, the growing body of literature around FLOSS development, in particular with regard to how such projects are organized. In addition, I look at a theoretical framework used in the thesis: Communities of Practice. I use this to analyze and discuss my findings later in the thesis.

2.1 Global software development

Hersleb and Moitra (2001:1) point to several challenges with regard to GSD*, "from the technical to the social and cultural"*. They claim that multisite development takes longer time than similar collocated tasks and emphasize communication and coordination as important. Furthermore they point to developers having different cultural backgrounds as a complicating point. Different cultures have different views on for example "the need for structure, attitudes towards hierarchy, sense of time and communication needs" (ibid).

Global Software Development involves what Sahay (2003:2) defines as Global Software Work (GSW) : *"software work undertaken at geographically separated locations across national boundaries in a coordinated fashion involving real time and asynchronous interaction"*. GSD is conducted by Global Software Alliances (GSA), which are *"organizational forms established to support the conduct of GSW"*. Offshoring is perhaps the clearest example of this, where a company outsources a certain programming task or service to an organization in a different location.

GSD is driven by the process of globalization, where actors have to participate in an increasingly global market. GSD seems to be one strategy by which to survive in such global markets. Hersleb and Moitra (2001) point to lowering costs and gaining access to skills and resources as two important reasons for GSD. Furthermore they note that the trend is accelerated by the following factors:

- *the need to capitalize on the global resource pool to successfully and cost-competitively use scarce resources, wherever located;*
- *the business advantages of proximity to the market, including knowledge of customers and local conditions. as well as the good will engendered by local investment;*
- *the quick formation of virtual corporations and virtual teams to exploit market opportunities;*
- *severe pressure to improve time-to-market by using time zone differences in "round-the-clock" development; and*
- *the need for flexibility to capitalize on merger and acquisition opportunities wherever they present themselves (ibid :17)*

There has been little empirical study of the field of GSD (Sahay 2003:4). The focus on GSD research has been on elements such as coordination and standardization. Coordination refers to how one divides up work tasks among the various sites involved in development, and then how to integrate that work. Standardization is a common way of solving the problem of coordination (ibid). The organizations involved will create reporting standards, choose standardized development tools and standard measurements of developers' contributions and thus lower the need for explicit coordination. Several authors have noted how standardization is a difficult affair as it neglects local variations. The decision of what and how much to standardize must be balanced with the local contexts of those involved in the development.

Sahay & Sarker (2004) describe one way of looking at collaboration with regard to virtual teams, as a multi-faceted phase in the evolution of a team. They study how virtual teams of students from two different universities work together throughout work with a course project. They analyze the findings by employing a theoretical model which involves a micro level of communicative action, a macro level of participant structure and the interplay between these levels. The micro level communicative actions take the form of *turn-taking* and *dealing with trouble*. The macro level participant structures are either *production structures* or *social structures*. The production structure, the resources the team draws on to be able to complete the task at hand, manifests along two dimensions: *Task focus*, how much time the teams spends on substantive tasks, and *task ability*, which denotes to what degree the team has the expertise and skills to be able to complete the task. These skills are both technical and behavioral. The social structure is made up of four interrelated dimensions:

- Virtual presence: How the participants share consciousness about each other's presence online.
- Social responsiveness: Unidirectional, bilateral or mutual communication between team members.
- Shared goals: To what extent the team members perceive and agree on a common goal.
- Identity: How the team members perceive themselves in the relation to the team.

(ibid:6)

The team will move through phases like Initiation and Exploration, and possibly reach a phase of Collaboration before the activity ends in a Culmination phase. In each phase the type of communicative actions employed may change, as well as the structural properties of the teams. In the Initiation phase the participants are focused on individual goals, and have an individually based identity. Communication is infrequent and often unidirectional. In the Exploration phase, the goals reflect local concerns, communication is bidirectional (sides conversing past one another), both teams are more virtually present, but there is no agreement on how or how often to be present. The identities of the participants are focused, on their collocated teammates rather than on the virtual team itself. In the idealized Collaboration phase teams have gotten to know each other and communicate effectively. Virtual presence is common and there are common norms about it. An understanding has been reached and goals are shared as a result. The team has a congruent identity, identifying themselves with the virtual team. Some teams never reach the Collaboration phase and move straight to the Culmination phase when the team work is completed.

This communication model can be illustrated as two nodes. In the Initiation and Exploration phase, communication between the two nodes is limited, while communication inside the nodes may be frequent. The ideal result is to reach a Collaborative phase where the nodes see themselves as members of the *same* team. In the initial phases, the nodes talk to each other as two entities, and internally to the each individual participant. In the Collaboration they are still two sub-teams but communicate as entities doing some sub-task of a whole.

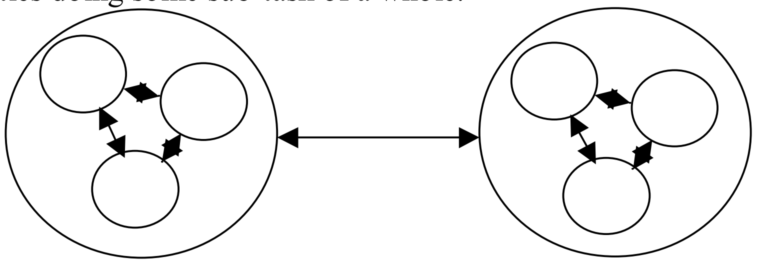


Figure 2.2: A sub team communication model

In a study of a Norwegian-Russian outsourcing project, Imsland and Sahay (2005) argue that the embeddedness of knowledge causes a problem when teams attempt to communicate. Three problems were encountered: First, there were severe language problems between the two teams.

Neither team had English as their first language. Second, the domain knowledge involved in making a salary system for Norwegian conditions was not easily understood in the context of Russia. At the same time, documentation on the existing system was in Norwegian, and for some reason, this was also chosen as the language of the new system. Third, project management issues prohibited effective communication. The Russian developers were not proficient in English and communicated their issues and questions to a project manager which then communicated these to the Norwegians in English. A similar process of steps occurred on the Norwegian side. This is summarized in the following figure:

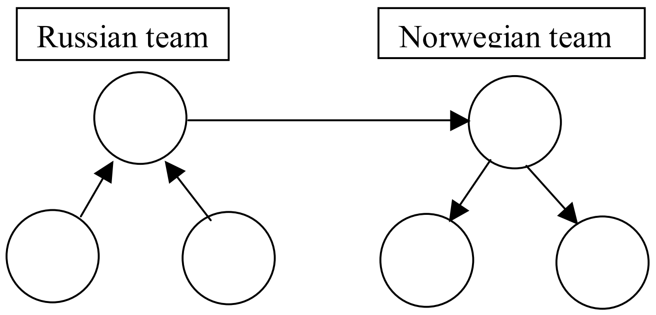


Figure 2.3: Communication in Imsland and Sahay's (2005) case

This communication process turned out to become slow. The Russians reported that they sometimes had to wait several days for a reply, including follow up questions and clarifications, which caused frustration and delays on their end. At the same time, lots of meaning related to the domain knowledge was lost in the translation process as only textual communication was used. Language issues had prohibited the effective use of video conferencing software for more direct communication.

A public mailing list was used to communicate the issues in order to document the process in archives. The project manager was also supposed to report project status, but this was problematic: The bosses of the project manager were also present on the list, and due to attitudes toward hierarchy, the project manager felt compelled to report that everything was OK. This was not checked by the Norwegian side.

An interesting aspect of this process is the use of the instant messaging network ICQ, which the developers themselves took the initiative to being using. With this tool, the Russian developers got a direct link to the Norwegian developers, especially one who spoke Russian. This happened without the knowledge of the managers on either side, and established an independent and direct channel between the developers, summarized in the following figure:

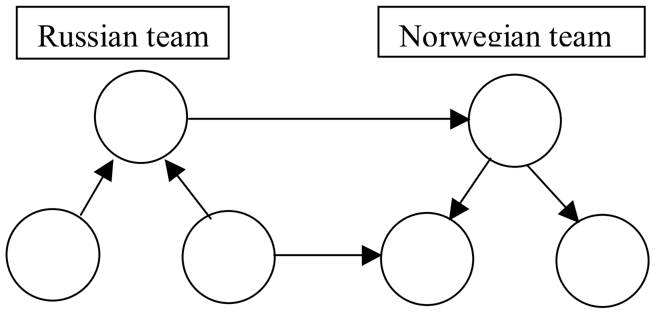


Figure 2.4: Communication with IM in Imsland and Sahay's (2005) case

This helped reduce the time for replies to queries to get through. Additionally, *"they were able to intersperse small bits of personal information with each other along with the technical. This interaction helped to develop both the technical and social side of the relationship."* (Imsland & Sahay 2005:26). But as ICQ was increasingly used, the Norwegian developer was overwhelmed by the number of questions. Furthermore, as the managers didn't know about it, he wasn't rewarded for his "invisible" work, rather it affected the time left to work on other things.

2.2 Free/Libre Open source development

Following the success of the development of the Linux operating system, there has been a considerable increase in the number of projects which are open source. This has also brought forward a special kind of project management, suited independent and geographically distributed developers collaborating a project using the web as the primary means of communication. Weber (2004:2) argues that *"[...] the open source software process is a real world example of a community and a knowledge production process that has been fundamentally changed, or created in significant ways, by Internet technology."*. In this section I first go through some general research about FLOSS development. I continue by looking at how FLOSS projects interact with hosting organizations. Finally, I look at how electronic collaboration takes place in FLOSS projects and how tools are used.

Feller & Fitzgerald noted in 2000 that reports on FLOSS have been somewhat evangelical, focusing on the success stories, and, since the term "Open Source" was coined in 1997, *"little rigorous academic research has been conducted on the topic"*. Since this statement was made, we've seen several studies made into OSS (Scacchi 2006). Many have been studies of well-known OSS projects like the Linux kernel (Raymond 2000), Fetchmail (ibid), GNOME (German 2004), Apache (Mockus et al 2002) and Mozilla (ibid). Some studies have attempted to draw conclusions on the basis of many small OSS projects, for example looking at projects hosted on SourceForge (Scacchi 2006). Many of these projects are seen as archetypical FLOSS projects, although they actually have varying practices. The Linux kernel represents an open, bazaar style approach to development, while Apache is founded on a meritocracy. The common denominator for these projects, however, and thus studies made on them, are voluntary participation and weak organizational ties. Volunteers come together to "scratch their own itch" (Raymond 2000), they are developers but also users of the Apache web server, the Mozilla Firefox browser or the GNOME desktop environment. The organization of the development process is somewhat ad hoc, governed loosely by for example a benevolent dictator, a council of experienced developers, or by the development community at large (Scacchi 2006, Weber 2004).

Furthermore, few studies have been made on OSS projects hosted by formal organizations, especially companies such as RedHat. At the same time, there is an increase in commercial FLOSS development (Open Source 2.0). There are organizations involved in the more known FLOSS projects, however, like the Mozilla Foundation and the Apache Foundation. In 2005, the Mozilla Foundation even announced a for-profit spin-off organization called the Mozilla Corporation. But these foundations do not represent a classic organization with a hierarchy and do not decide which features to implement in the corresponding software. A different model involves building a community around a commercial product, and allowing varying degrees of access to the development to outside developers. This is exemplified by Sun, IBM and other major companies which have invested in open source technologies or released their products as FLOSS. Examples are the Java Virtual Machine and the Eclipse IDE.

So there is a need to explore the ties between organizations and FLOSS development: How collaboration takes place in the organizational context, how development is affected by the history and practices of the organization, and in turn how practices from FLOSS projects affect the development process.

Eric S. Raymond uses the terms cathedral and bazaar to describe two different ways of doing software development (Raymond 2000). Many have wrongly interpreted the cathedral model to mean classical software projects, implying a top down process, and a central person or set of people define and design the system. The task of programming is recursively split up into components, which all fulfill specific tasks. This way, teams of developers can focus on different components. The need for coordination is reduced, as the components and interfaces between them are well defined in beforehand. However, Raymond's (2000) reference is to development of the GNU tools, where a professional group of developers develop something internally before releasing it to the public. The bazaar model, however, is more akin to the development of the Linux core, where literally everyone can participate. Patch submission is still tightly controlled, but is still open to the public.

Weber argues that the view of the open source process as a bazaar is an evocative image, but misleading as an ideal type (Weber 2004:113). Instead he argues that a central aspect of open source development is that there is no formal division of labor. Instead, developers choose individually which projects or parts of the code they want to focus on. As such labor is distributed, but not divided in the industrial sense of the word. But commercial FLOSS development can be used as a counter point where there is more "purposive planning", and the process becomes "less bazaar-like" with developers being paid (Fitzgerald 2006). Companies like RedHat, Novell and others definitively produce FLOSS software, but it is quite unthinkable that the employees of those organizations are completely free to choose which tasks they work on.

He then goes on to define different types of open source processes, which he claims are tied to the form of license chosen for the project. BSD-style projects, using the Berkeley Software Distribution license, are typically composed of a close knit team of developers which do most of the coding themselves. Although random external developers are free to submit code or suggestions, the team does not rely on this input. Those projects using the GNU Public License, GPL-style projects, instead have a low barrier for entry: Anyone can submit code directly into the project. Whether or not the changes are kept, is left up to a discussion on for example a mailing list. Yet another model is that of Apache which has a formalized system of members and well defined voting procedures for on most matters of importance.

The use of certain tools and processes seem to be common to most FLOSS projects. Such common tools include source code management, also known as software version control tools, like CVS and Subversion, used as *"both (a) a centralized mechanism for coordinating and synchronizing FOSS development, as well as (b) an online venue for mediating control over what software enhancements, extensions, or architectural revisions will be checked-in and made available for check-out throughout the decentralized project as part of the publicly released version"* (Scacchi 2006: 28).

The use of source code management is mediated by various communication tools, among them mailing lists, wikis and other asynchronous media which are *"persistent, searchable, traceable, public and globally accessible"* (Scacchi 2006:40).

Karl Fogel gives advice on how to start up and run a FLOSS project, based on his experience in the Subversion project and from observations of many other projects (Fogel 2005). He goes through setting up various communication tools like wikis, mailing lists, issue trackers, source code management and show what is common in FLOSS projects. He also elaborates on process issues, such as always keeping discussions public, which is a cornerstone in many FLOSS projects.

Furthermore, these tools are not only tools, but *"serve as venues for socializing, building relationships and trust, sharing and learning with others"* (Scacchi 2006:40). By participating in the use and sharing of values and beliefs the participants engage in a kind of organizational learning (ibid:38).

Lanzara and Morner (2003) argue that Open Source systems should be viewed as systems of interaction and not as organizations. The interaction of the participants is what actually structures the process. They argue for shifting focus from organizational routines to technical artifacts. The artifacts or tools used for communication inscribe organizational practices about how to do software development into the system.

All in all, this points to FLOSS projects having a certain communication model where a developer communicates with most other developers. But it's not a network, where each participant contacts a specific set of participants or maintains direct links. Rather, the FLOSS model is a *community* model. Communication doesn't go from one person to another person or from one set of people to another set of people. Communication goes into the community, represented by the technical infrastructure, the routing mechanisms, the archives. In its most extreme variant, the communication is all-to-all. Everyone receives the mail; everyone can read the wiki or see the issue in the issue tracker. Fogel makes a point about not keeping discussions private for several reasons. One is related to the motivation for participating: *"No serious volunteer would stick around for long in an environment where a secret council makes all the big decisions."* (Fogel 2005:33). Furthermore a public discussion represents a learning process where everyone involved learns about the project, how to relate to each other, and what is going on. Also, discussions in public can be archived and retrieved for future reference. Finally, a public discussion increases the chance of getting valuable input. All participants can make suggestions about technical solutions, project management and the like (ibid).

Specifically, the participants talk to certain sets of electronic tools which convey their communication to most, if not all participants in the project. The use of such tools results in what Scacchi calls "informalisms", which may include:

> *"(i) communications and messages within project Email, (ii) threaded message discussion forums, bulletin boards, or group blogs, (iii) news postings, (iv) project digests, and (v) instant messaging or Internet relay chat. They also include (vi) scenarios of usage as linked Web pages, (vii) how-to guides, (viii) to-do lists, (ix) FAQs, and other itemized lists, and (x) project Wikis, as well as (xi) traditional system documentation and (xii) external publications."*
> (Scacchi 2006:19)

These informalisms are an embodiment of the values and beliefs of the participants, and of the processes which participants are expected to follow (ibid:33). When in place they also constrain or facilitate the interface between participants. All these informalisms may have another type of function as well: As organizational memory, recording the activity and decisions made by the community over time.

I summarize this model in the figure 2.3: Here, the participants (the circles) communicate into the community itself, or specifically, the tools used by it, represented by the square. All the participants relate to others in some way through the tools. There will be other channels, of course, like private e-mails, but the vast majority of communication happens through the tools. The idea of openness and democracy pervades the technical set up of communication channels.

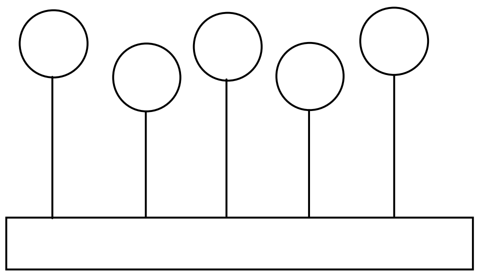


Figure 2.5: Community communication model

While this model seems to be prominent in FLOSS projects, being FLOSS is not a necessary or defining aspect of the model. It is possible to perceive proprietary development using a model similar to this. Communication may indeed be community like in a project inside a company with a hierarchy.

In the case of DHIS 2 development, we find a series of tools and processes which are common to FLOSS projects. At the same time, the use of these is somewhat of an oddity in the context of the organization that runs the project. I describe some of these aspects in my empirical material and return to them in the discussion.

2.3 Communities of practice

A global development project has a series of participants which still come together in a joint enterprise to produce software. In such settings, or due to the organizational connection of the project, there will be or grow forth some sort of culture or practices in the project. This is also true for FLOSS projects, which even has a strong history and ideological aspects. Lave and Wenger introduced the concept of a community of practice (CoP) (Wenger 1998), meaning a group of people who engage in a shared enterprise. These groups develop practices both to handle the pursuit of the enterprise and of handling the attendant social relations (ibid:45). A CoP can be a family, trying to organize their everyday life or a group of professionals, working to solve a problem. The group has a shared cultural history and artifacts. An important point is that the CoP is a result of an ongoing collective learning process. Learning is in itself a deepening process of participation in a CoP. Participants often move towards the center of the CoP, becoming more competent in the practices.

Core participants are those who have internalized the practices of a CoP, and perhaps act as central actors in it. Other participants are peripherally involved in the CoP, but become more and more involved as time passes. The peripheral participants internalize the practices and values of the community as they move inward. In a famous study, Jean Lave (Lave & Wenger 1991) looks at tailors in Africa, in a master-apprentice relationship. New practitioners begin in the periphery, but they are legitimate participants, for example as apprentices the master has taken on. This leads to the idea of legitimate peripheral participation, which is central to CoP. When starting in the periphery, the participant may be set to do menial tasks or observe others. As the participant learns more, he or she will gradually move from the periphery of the community and towards the center, gaining more respect by his or her peers and gets assigned to more challenging and important tasks. In the process the participant learns the trade, for example how to sow clothes, the formal skills, but also learns and adopts the views and culture of the community. Lave exemplifies this by how the tailor apprentices learn the culture the clothes are to be used in, the different kinds of social classes a customer belongs to, and so on.

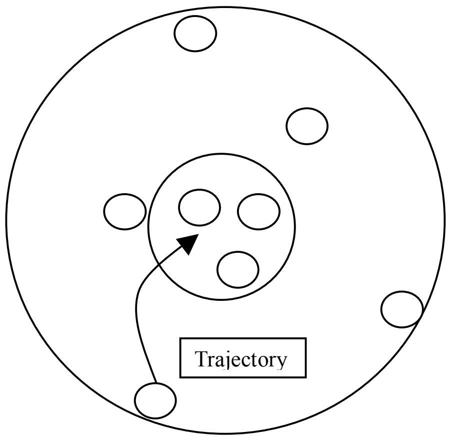


Figure 2.6: Model for a Community of Practice with participants in the center and periphery

The participant's movement and change in identity represents a trajectory in relation to the community. Such a trajectory may go through an education, imprinting the practices into the participant. Wenger, however, does not want to imply that there is a set trajectory people follow (Wenger 1998:154). Trajectories do not have to lead inward or to what Wenger calls full participation.

At the same time, becoming a part of a CoP does not mean simply internalizing everything indiscriminately or as a blank slate. Each participant brings with them their own values. By participating, they contribute to redefining the community itself. The CoP is not a frozen entity, but changes continuously.

Even though Wenger conceptualizes a CoP in very dynamic terms, I find a more static version to be fruitful for analytical purposes when discussing the case and view DHIS 2 development as a relatively stable CoP. There are participants who are central to the CoP and carry its values and practices. These are, as we shall see, mostly Norwegian developers. There are some participants who are more or less peripheral, including developers from other countries, non-technical participants, and so on. When applying this viewpoint, I look for and discuss various trajectories. I have chosen to view some trajectories as institutional rather than personal. I believe this viewpoint brings some interesting analytical results.

Wenger et al (2002) take the idea further, looking at how to develop distributed communities, where participants are spread out across the world, typically in different organizations. This can be different branches or business units of the same company, or a wider community where several companies participate. They claim that one shouldn't treat such a CoP as a monolithic structure, rather as a set of cells. They borrow the term of fractals to describe groups within groups, with different connection points. Each cell can be a local group. In their example, people from one cell can communicate with the rest of the community through a coordinator, have their own internal community and so on. They claim that by supporting this one can allow for a greater variation in culture and organizational structure (ibid: 126).

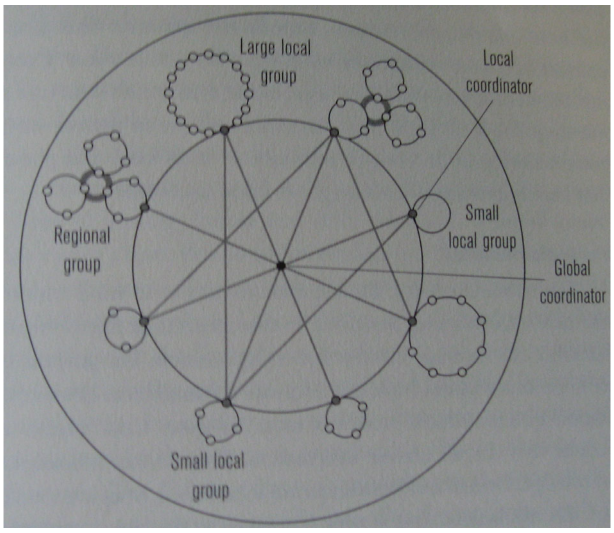


Figure 2.7: Fractal model for a distributed Community of Practice (Wenger et al 2002:127)

Wenger et al focus on the role of the coordinator as a person who nurtures a local community and tries to connect people in the global community. This person can point members to other members who may have useful information on a topic. The coordinator keeps tabs on the community, introducing questions and debates to stir up activity. The authors do not see this person as a manager, through which information must pass in a strict hierarchy, but more as a resource person and maintainer.

Wenger et al makes some points with regard to this:

1. Achieve stakeholder alignment. The stakeholders need to agree or at least spell out the goals of the community. Is it to get support, participate in development, get functionality, report bugs, etc? There is a need to reconcile multiple agendas.
2. Create a structure that promotes both local variations and global connections. For the authors this represents supporting local communities, and making connections to the global community.
3. Build a rhythm strong enough to maintain community visibility. The role of the coordinator is central here, trying to spark activity by introducing discussions, contacting various members and so on.
4. Develop the private space of the community more systematically. In this lies creating connections between the individual participants and creating a web of trust, often handled by the coordinator.

I shall return to these points in the discussion.

3 Method and data

In this chapter I give an overview of the different methods I used and the data I collected. In the thesis I use a method triangulation approach, combining quantitative and qualitative data to get a broader picture. As I have been an active participant in the project which serves as a case, I also draw on my own experiences from the project. The methods are described and some issues are discussed in relation to them.

3.1 Developing questions and evolution of the thesis

The work with this thesis started with websites. The original task was to design an online presence, a portal website for HISP, to help curb the fragmentation of information within the network and present a consistent face for the organization. My own interests changed during the course of the work to electronic collaboration in the network in general. Ultimately this was limited to a study of the specific case, as this was also what I had most knowledge about: Write what you know. A research project is not a linear movement from questions, via data and theory to a set of answers, but a constant movement between them (Kalleberg 1997). Questions are inspired by theory or data, findings along the way may change the direction of the research, and the answers you arrive at in the end may cause you to go back and revise the questions or revise the theoretical framework. This is illustrated as a series of interrelated aspects of research.

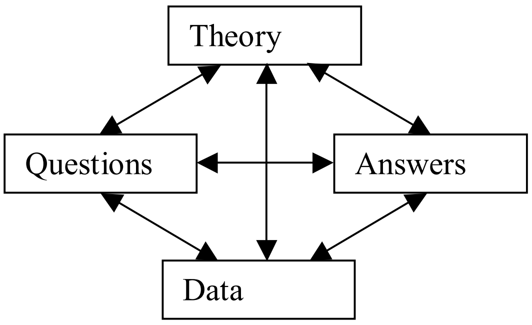


Figure 3.1: Kalleberg's (1997) diamond of interrelated aspects of research

During the process my questions changed as I worked with the data. Partly because interesting things came up, but also because I needed to consider more closely what questions I could answer given the data I had, what I could say something about. It was always clear that electronic collaboration and the use of tools would figure heavily, but it had to have relevance beyond just the case I was looking at. The idea of a communication model, an abstraction of how communication happens in different types of projects was introduced quite late in the process, although it had been present in the early stages as a set of drawings that I felt illustrated something. Fleshing it out as a useful and precise tool was more difficult. Ultimately, it's a simple concept, but something to use to compare projects based on certain characteristics.

3.2 Quantitative analysis: Grouping and counting

Quantitative analysis is often carried out by defining hypotheses about the relationship between one or more variables in the research, for example that the higher the age of a person the more likely is this person to be religious. Religion is here a variable which then depends on, or has a positive correlation with the variable age. One would then set out to select a manageable but sizable selection from an interesting population, say the population of Norway. Following this, one can collect data, for example numbers of memberships in religious communities and people's age, or by doing surveys. The data are then analyzed with regard to the hypothesis (more correctly, the null-hypothesis, stating that there is no relationship between the variables). Through various statistical calculations one can determine whether or not there is a correlation, and whether or not this is statistically significant, i.e. whether or not it can be generalized to apply to the entire population. If this is the case, then based on the selection one can say with statistical certainty that the entire population follows the same pattern.

I had limited experience with quantitative methods and how to explore statistical significance and generalization. I didn't feel I could carry out such a type of analysis in a precise way. Furthermore it would be difficult to argue that the selection of DHIS 2 developers would be generalizable to some sort of population of globally distributed developers. It is also hard to argue that country of origin would inherently effect participation in electronic collaboration, and even harder to prove it statistically. There would be too many interconnecting variables and spurious connections. As mentioned, it seemed to me and most of my Norwegian colleagues that there was an asymmetry in the use of the tools. I decided, however, to document this properly. I wanted to see the actual evidence, and what would possible moderate such findings. This is the reason for choosing the first of my methods: Grouping and counting, a simplified form of quantitative analysis. Simply put, I wanted to count how many mails and how many commits were being made by various participants. One could say I did have a hypothesis, based on anecdotal evidence, but I went into the analysis wanting to challenge that. Perhaps people were simply participating in different ways, focusing on different things.

I wanted not only to see who contributed how much, but also if that participation changed over time. As such I sorted the material by months. Months were chosen slightly randomly. It seemed like a sufficiently fine grained time span to use for the analysis, while being manageable.

3.2.1 Thread analysis: Who are initiating activity?

In addition to counting and grouping of individual mails, I wanted to get a feel for who were the initiators and who were the respondents on the mailing list, how long discussions lasted and the answer rate on mails. To this end I tried to break the mails down into threads of discussions, analyzing the from fields in the mails and counting the length and frequency of the threads.

3.2.2 Content analysis: What are people talking about?

In addition to counting and determining "who" and "how often", I wanted to know what was being discussed on the mailing list. To this end I created a tagging scheme. I read through a portion of the mails on the list and tried to determine what they were about. I then tagged the mails with a set of keywords, stating that the mail was about technical discussions, project administration, praise and the like. These tags were then also subject to a simple quantitative analysis of counting and grouping. I especially wanted to see if the pattern of what was being talked about changed over time.

This form of data represents a mix of the quantitative and the qualitative. The reading and interpretation of the material is definitely of a qualitative nature, and perhaps somewhat "fuzzy" and subjective. I bring that into the quantitative domain by grouping and counting it. By choosing a high enough amount of mails, I hope that the data is representative of the total. The population here is then all mails in the project. I have not, however, opted to do a hard calculation of statistical significance.

3.3 Qualitative methods: Making the descriptions thick

Interviews can be done in a structured or semi-structured fashion. The structured variant is closely related to quantitative research: Respondents answer a pre-defined set of questions. This is good for comparing across respondents and can function as basis for quantitative analysis. It requires that a lot work to get questions right, and must be geared toward a specific goal. It is usually not possible to change the questions when the process of doing interviews has been started. The open, semi-structured interview is more exploratory, often accompanied with a loose interview guide. The focus is on keeping the informant talking, and allows the interviewer to change focus if something interesting pops up. Consequently, the set of questions and subjects raised may vary between informants. However, as one of my lecturers in qualitative methods stated: "Semi-structured does not mean lack of structure". There should still be an interview guide with some question, however open-ended, and a defined subject. It is subsequently possible to analyze the resulting data by for example grouping the answers into categories or by looking for specific words used by the informants.

I had more experience with qualitative research, mainly interviews and observation. I'd taken courses in both qualitative and quantitative methods, but I'd only practiced small bits of qualitative research through my education. As such, qualitative methods felt natural to use. I knew that my form of quantitative analysis would be very limited. Furthermore it wouldn't give me anything about the feelings or motivations of the participants, only confirm that there was an asymmetry. So, to me the answer lay inside the heads of the various participants. The grouping and counting would be a starting point, for showing that there is or isn't a difference, and how that difference may change over time. This was furthermore strengthened by the following statement by Braa et al (2006:15):

> *"[M]easurements [...] provide only tentative indications on the level of participation as they do not cover the informal communication and many of the more ad-hoc local software development activities"*

.

While a lot of activity happens over the public list and in the repo, there is a wealth of communication going on directly between people. In my thesis, however, I have not focused as much on this, beyond people's usage of instant messaging clients.

I perceived two ways of finding out about people's motivations and views: Either to read through the data, the mails and logs, looking for comments about usage of the tools, or talk to the participants directly. I reasoned that interviews would be more appropriate, as it would allow the participants to talk more in depth about the subject. Furthermore, going through all the mails would be too time-consuming.

I opted for the semi-structured approach, as I didn't have a clear goal from the outset. The subjects were usually how they used the various tools, how they felt they worked, and they felt collaboration in the project worked in general. I also asked questions about the informants' relationship to FLOSS. I wanted to understand both how the individual informants and the nodes they worked in related to the collaboration. I had already done some interviews with regard to the early part of my thesis, and I wanted to avoid throwing them away. I realize there are some dangers in this: The informants I'd interviewed had not responded directly to questions related to DHIS 2 development, rather about website design. However, I had gotten some relevant information on the HISP network and on how the various nodes operated. India, for example, is strongly involved with DHIS 2, which means much of their processes and challenges were relevant. All in all I did 17 interviews. I interviewed five Norwegian developers, one Norwegian coordinator, one Norwegian facilitator, one Indian developer and three Vietnamese developers by mail about their usage of the tools and about collaboration. Additionally I interviewed two Norwegian coordinators, one South African coordinator, one South African developer and two Indian coordinators about websites, HISP and their local nodes. The South African developer and one of the Indian coordinators were interviewed during a conference in South Africa. The others were usually interviewed in their offices.

In my thesis I have not opted for a very structured analysis of the interview data. I felt the set of questions were too small and their variations too big. Instead I tried to draw out interesting quotes and views which exemplified some problem or advantage with the various electronic tools and how collaboration is carried out. I bring these views up as part of the discussion and look at ways to address them.

3.3.1 Document analysis: Wiki and websites

The DHIS 2 project has a wiki space with a lot of information. I've read through it, and also contributed a lot to how it is organized. Furthermore, many nodes in the HISP network have websites. Most prominent of them are www.hisp.org, the home of HISP South Africa and http://www.hispindia.org, home of HISP India.

While I have read through these, the data in them are not analyzed in detail in this thesis, rather they serve as background and reference material. I come back to the wiki when dealing with trajectories for learning in the project.

3.4 My own experiences: Participatory observation vs. Action research

In participatory observation the researcher takes part in an activity, seeing it for herself and experiencing the situations she is trying to understand. The researcher will usually make field notes along the way and spend some time analyzing the material afterwards. Action research differs from this in that the researcher is more active in constructing and re-constructing the organization he or she is working inside. The action research has four main phases: Diagnosing, action planning, action taking, evaluating and specifying learning (Baskerville & Wood Harper:238). In the diagnosis phase the researcher and representatives of the organization engage in identifying the problems of the organization that are valid to the research. After this a set of actions are planned which hopefully affect the problem. The actions are carried out and the impacts on the organization and the problem evaluated. Finally, based on the sequence, one tries to distill knowledge of the success or failure, and begin the cycle anew if necessary.

My original thesis problem was more action research oriented: Based on the feedback from the HISP network, I would construct a new website to address the fragmented online information in the network. A prototype would be used to improve information presentation, and based on its effects and feedback, this online presence would be changed. As my research interest changed to electronic communication in general and to DHIS 2 in particular, the research project took on a more descriptive form. During the course of the research I have been active in making suggestions with regard to communication, participated actively in discussions, and helped set up new tools. As the activity in the network is very action oriented, I did engage in action research like actions: One of which was reorganizing the documentation on the wiki, which was just done. The general feedback was positive, although we did not engage in a formal evaluation process regarding its effects.

However, I have participated as one of the most active developers in the project over the last two years, although not as active as the core developers. Consequently it is at times difficult to separate my role as researcher and my role as a "worker" in the project. Much of my research material is gained from just being a part of the project, observing the activity and making some notes of what I saw. In many cases I have raised the issues discussed in this thesis as discussion points on the mailing lists and in meetings. I go into more detail about my experiences under Chapter 5. As such, I cannot claim that my process constitutes a rigorous action research process. Rather it was a more liberal approach (ibid:241), where what I did was very much embedded in my work on the project. The evaluation and learning phases were not as disciplined as they could have been. In some cases the changes I was involved in came too late to allow any useful experiences to be gleamed.

3.5 Data analysis

In this section I describe more in detail how I analyzed my empirical material. The analyses were inspired by John Alexander Miller (2004), which performed a similar analysis of an education-related newsgroup. In his dissertation Miller looks at the content of messages by tagging them, and divides the messages into threads. He analyzes the threads with regard to their length and who participate in them.

3.5.1 Grouping and counting

In the following I go into detail on how I compiled my data into an analyzable form and the categories I chose for the analysis.

3.5.1.1 Mapping the data

In all of the questions it is interesting to note who the participants are, in order to be able to break participation down by country of origin and role in the network. I define a participant as a person who has either committed to the repository, sent a mail to the list or both. Committers are identified with a user name on the hisp.info server. Mailers are identified by the from field in the individual mails.

One particular problem is that several people use different mail accounts to send mail to the list, thus creating some irregularities in the data. I counted up to four different mail addresses for one participant. I needed a way of identifying people across both mails and commits, so I decided to create a database table which mapped between a user's 'id', in this case the hisp.info user name, a set of mail addresses (address1,address2,address3,address4) and finally to a country and a role. Some committers do not write on the mailing list and some people who send mail do not commit. I tried to solve this problem by creating made-up usernames (not present on hisp.info) based on a person's primary mail address.

Another reason to do this is to be able to report on these data without explicitly breaking a rule of anonymization. While the data are publicly available, it's not certain that the different participants would like to see their participation public in an analyzed form. The table structure allows me to display aggregated data using the country, role and designation of the person, rather than using their full name.

All in all, I identified 154 distinct participants. 3 of these were traced to administration accounts on the hisp.info server (used for sending test-messages to the list) and 2 were spam (wow, a mailing list actually bought a lottery ticket?!).

3.5.1.2 Country and role

I define country as the country of the node a person is primarily working for in the context of the material. For example, one developer was from Ethiopia, but did most of her work in Oslo. A certain facilitator is from Italy, but does his work in India. These two are labeled as 'Norway' and 'India' respectively.

The role is an attempt to approximate what sort of function the person primarily fulfills with regard to DHIS development. I defined the following roles:

- Core developer
- Developer
- Facilitator
- Coordinator
- Student
- External

Core developers are the developers who have been given the title, including one central developer before the title was instituted. Developers are those who have contributed consistently over a longer period of time. For example, the participants of the INF5750 course do develop on DHIS 2, one could say they at times represent the backbone of DHIS 2 development, but only a fraction of these continue to work on DHIS 2 after they have completed the course. Those who continue are usually master's students. In addition to these, HISP has several employees in Vietnam and India who are also primarily developers. A facilitator is a person who visits sites where the software is being used or introduced and helps with the implementation of the system in the organization and training of users. A coordinator represents someone within the higher echelon of HISP, a leader in a node or a leader for some activity like DHIS 2 development. Students represent the horde of people who participate in the INF5750 course, but do not contribute after the course has been completed. Finally, externals are those people who contact the mailing list for some reason but while not having any formal connection to the project, for example some Vietnamese teachers, some health care officials other's interested in the project.

These roles represent my approximation of the activity of the person. One could say that each person can, and indeed does, have many roles at the same time, or change roles over time. For example, several of the developers started as students, the facilitators and coordinators may occasionally do work as developers. I draw on my own experience from observing the development over time in order to classify people into roles. As such, there may be erroneous elements, but I believe the categories and assignments are sufficient to get interesting data.

In order to be able to separate people with the same role and country across the material, I designated them with a number. I only did this where I felt the results would be significant: The core developers, the Norwegian and Vietnamese developers and the coordinators from the same country were counted. I did not denote the different students.

Some participants could not be identified. In these cases I tried to tentatively identify a country of origin based on the names or e-mail addresses of the participants. I asked other members of the project to help identify particular users, but only when working on the users, not when working with aggregated data. Where identifying the role or country failed, they were set to Unknown.

3.5.2 Content analysis

I started by splitting the mails up by month. The reason for this was that I wanted to see if the content on the list changed in any way over time. One could hypothesize that the initial activity was dominated by requirements, technical discussions and project administration. As DHIS 2 gained a user base it would shift to support activity and the requirements discussion would be reduced as the system reached maturity.

In the selected period, there were a total of 3754 mails on the list, with over 200 mails for some months. Going through them all would be too time-consuming. I decided to go for a representative approach where I'd select a random set of mails from each month, inspired by the work done by Miller (2004:125). I tagged each mail with a specific tag, a category of what the content is about.

Examples include technical discussions, support, requirements, norm enforcement and others. I started out by selecting ten mails from each month and went through them to get a feel for the content and in informing the categories I chose. During the tagging process I collected mails which I felt were good examples of the categories and some that were difficult to classify.
A problem with this approach was the difference in the number of mails pr month. The initial ten mails from each month were thus in some cases far too many or far too few when compared with the total number of mails for that month.

Another issue with the mail data was that the SVN commit mails were sent directly to the dev list during the first three or four months of the development. This makes the number of real mails artificially high in this period. I chose to account for this by explicitly tagging mails as SVN commit mails so that their effect can be visible.

I tried to define a few rules to guide the tagging. First of all, I should primarily look at the mail in an isolated way, what is this particular mail about, rather than looking at a thread context. Thread analysis comes later in this text. This couldn't be practiced as an absolute rule, however. Sometimes it was necessary to read what the mail responded to, in order to understand what the answer was about. I did two runs in addition to the first 10 mail pr month run, in order to consolidate and reconsider the tags. In addition, I tried to retrieve examples while I did the run.

3.5.2.1 Categories

I ended up with thirteen different categories, described below. The examples next to them are fictional. See the section on examples below for actual examples from the data.

- Technical discussion, Tech: Discussions of which technical solutions to choose, organizing of code, etc. Not necessarily connected to specific system requirements. Examples: Should we choose this or that framework or organize code into this or that package structure.
- Enforcement of norms, Norms: Comments or criticism related to the norms around communicating on the mailing list or good development practices. Examples: Remember to answer below that you're responding too, please use the DHIS 2 code style.
- Requirements, Req: Discussions about system requirements. Overall considerations on how the system is supposed to functions. Feature requests from users.
- Support requests and answers, Support: Support requests based on error messages and bugs and questions from developers on how to use features in the software. And answers to the questions. Examples: I'm getting this error, I'd like to report the following bug, how do I use this feature?
- Bug reports or discussions, Bug: Reports that report a specific bug in the system. Differs from Support in the sense that the behavior is due to errors in the code or misunderstandings of requirements.
- Issue discussions, Issue: Discussions around concrete JIRA issues. Not necessarily bugs.
- Social "chatter", Social: Humorous comments and social discussions not directly related to development.
  Examples: Did you see the last X-Men movie, where to go for food in Hyderabad.
- Praise, Praise: Positive feedback on the software or some effort. Good work, amazing, I like this application
- Project administration, Project: Division of labor, conversations on how to organize the project or the support structure. Examples: When do we release the next version, can you work on this module
- Broadcast messages, Broadcast: Messages that are meant to be one way communication. Information on stuff that has happened, releases and the like. Examples: We have now released, this is the current status.
- Administrative notices, Admin: Examples: The wiki is down, the wiki is back up.

- License discussions, License: Discussions about which license to use. Not really a technical matter, and not really a project administrative matter.
- SVN: SVN commit mails. Affects the number and averages on the mailing list in the earlier periods of the project.

3.5.2.2 Examples

In the following I'll quote some examples of how some mails got tagged and why. I'll also quote a few of those that were difficult to tag for whatever reasons.

```
> A computer running dhis would normally be used by just one Org.Unit,
> right? But must not be locked to one. Wouldn't it be easier if it was possible
to
> specify the Org.Unit in the header of every page? And link the value to a
session
> and across sessions, making it unnecessary for an Org.Unit, that doesn't share
> it's computer with other Org.Units, to specify it's name each and every time
> the Org.Unit is to open a page in the system?
>

This is not correct, the typical data entry user of the system is a data
clerk at the district office that is responsible for data entry of 15-30
health facilities. So there will in most cases be data entry for many
different units at every installation of the application.


> Do all Org.Units have access to all forms? If not, when designing a custom
> form, who's to deside which Org.Units have access to the form and how to
> specify this in an easy way.

From a health domain perspective there should be no need to restrict
access to forms, but this feature might be linked to users and roles..
```

In the example above the participants are discussing use cases in order to understand how the GUI of the application should be arranged. There are technical aspects of the discussion, the arrangement of GUI components, but the mail is primarily about use cases and thus which requirements exist for the application. Thus the mail was tagged as being a requirements discussion.

```
> Btw, I still have an FAQ question regarding lazy initializations open.
> Could one of the core developers take a look at it?

I think it would be better to phase out the FAQ as a place to ask
questions. Using the mailing list is better. Then we can put actual
FAQs in the FAQ. We need more mailing lists...

As for the lazy initialization problem, I'm working on it...
Temporarily fix: Go into the *.hbm.xml files, and set the collections
to lazy="false". Most of the collections are set to lazy="true" (as
they should be), so when the sessions close it's no longer possible to
initialize the collections that where not loaded.
```

In this example two things are happening. The respondent suggests phasing out an interactive FAQ as an arena for support questions. In the past, the students of INF5750 would ask questions related to the system, their obligatory assignments and the technologies being used on a wiki page. In this mail the respondent argues that this should be done on the mailing list instead. This means that the respondent is commenting on how to organize the project and its support structure. It was thus tagged as Project. The second paragraph, however, is trickier. Initially it would seem like an answer to a support request. The original author has a question about lazy initialization. However, the response also has elements of a technical discussion (I'm working on a solution, here's a temporary fix.). This might indicate that the original question was actually some sort of bug report or become one. There is no discussion about which kinds of solution to choose here, however, so I ended up adding the tag Support.

```
> I also thought about retrieving the ValidFrom and ValidTo using the
> dataElementId parameter, but as I understand it, the ValidFrom and
> ValidTo in the DataElement and the SemiPermanentData tables aren't
> necessarily the same.

In DHIS2, all DataValues has an association to a Period. The startDate and
endDate of that Period will be used as "validFrom" and "validTo" for
semi-permanent data values.

In DHIS2 DataElements doesn't have a validFrom and validTo, is this
something that is used in 1.3? If so, when and why?
```

Similarly, this mail has different topics. From the entire mail it is clear that the original author is considering and suggesting different ways of doing a technical implementation. The respondent informs this discussion by explaining how the current system is set up. As such it deserves a Tech tag. The last paragraph here, however, is a question on how a previous version of the system uses a feature and a request for a clarification. The previous versions of DHIS were in a sense used as a requirements specification, so this also becomes a requirements discussion.

```
> When storing a DataValue a reference ID to the Period is required
> (naturally). Will these Periods always exist prior to storing of
> DataValues, or do every module that stores DataValues have to check if the
> Periods exist or not and create the required non-existent Periods before
> actually storing the DataValue?

The Periods will always exist, they will either be set up in a maintanance
section prior to registration of DataValues or already exist in the
initial database. The modules that store DataValues will choose a Period
for registration, so it will not be possible to pick one that doesn't
exist.
```

In this example there are two aspects. How does the current system work (how should I code against the system), and how will the system be used in the real world (Periods will have been set up by a maintenance section). The mail was thus tagged with Support and Req.

```
> I agree with the need for more documentation on internationalization. I
> suggest that the i18n (INT) module discuss this with Kristian shortly and
> provide some information as soon as possible.

Since several modules work with internationalized DataElements and so on, it
would be easier if there was an InternationalizedDataStore interface that
replaced all the names and such. Just a suggestion.
```

While the original text is about project administration, the answer, and thus this mail is a purely technical suggestion which has no direct bearing on specific requirements. It thus warrants the Tech tag.

```
We've been using your xwork-optional Spring/XWork integration package as
part of a university course. We're helping to develop an internationally
used heatlh information system.
See:
```

```
http://www.uio.no/studier/emner/matnat/ifi/INF5750/
http://www.hisp.info.

We're wondering if you guys could upload the xwork-optional package to the
IBIBLIO repository. This way, everyone can download the jar using Maven.
This would be very useful, at least to us.

See:
http://maven.apache.org/reference/repository-upload.html
for instructions.
```

This is an oddity in the sense that a developer has simply CCed the list when sending a mail to someone else. It is just meant to inform the list that something has happened. The author probably doesn't expect a reply. As such it is a Broadcast message.

```
> It is important to reduce noise (otherwise important things get
> drowned) while not fragmenting the community too much. I think your
> proposal sounds reasonable, but rather than that many mailing lists, I
> suggest we use the new phpBB forum, with possibly a forum for each
> module.

With an additional forum we need to keep track of changes to the wiki,
jira and the forum. This takes time... We also get another thing that we
need to administer and we already have problems administrating the server
/ mailing lists..
```

This is a pure discussion on how to organize the project, i.e. which tools to use for communication. There is no discussion with regard to the actual system or requirements around it. It warrants the Project tag.

```
> Valid points. What do other people think?

Hi,

I'm against using any kind of GPL license for DHIS2.0. We have seen the problems
that you are faced with when working with the GPL (eg. the report designer,
IReport). We are not sure if what we are doing is legal and we really don't want
to mix licenses in our project .
```

The licensing discussion crops up now and then. It's not a purely technical matter, although it has ramifications on how to organize the code, specifically whether or not the project can include GPL'ed libraries. It's not a purely project administrative matter, although it requires a decision from someone higher than the individual developers. It doesn't have to have consequences for how the project is organized as an entity. I chose to label it as its own category, License.

```
We're receiving several requests for help with the software on the mailing
list, which is good. We're all interested in helping out as much as we
can! However, short messages alà, "I get an error. Please help", aren't
useful. When stuff don't work, please supply the following, if possible:

[list cut]

Feel free to add to this list, but the summary is: Be as specific as
possible, and supply as much information as you can. This way people can
help you more easily and quickly.
```

On the one hand this is an attempt at shaping how the project's support structure should be, but on the other hand it is also a normative statement about being specific in bug reports. It gets the tags Project and Norms. This mail elicited a response where the respondent asked author to put the general information on the wiki for future reference.

```
> I down load the newest source of the dhis-report-generator in hisp
> website.
> But it doesn't work. Beacause it is not enough references. I don't know
> why and how to make dhis-report-generator work?

Do you get any error messages?
```

This is a classic example of a support request and answer. Some user or developer has a problem and reports it on the mailing list.

3.5.3 Thread analysis

A thread is here defined as a set of mails having the same subject and being relatively close to each other in time. Threads are difficult to track. They typically have the same subject, replies are usually prefixed with “Re: ”. Some threads change subject over time. The fact that there are many mailing lists in the project and because some threads start outside the mailing list, complicates matters. For example, mails to the dev list are prefixed with ”[Dev] ”, but if the mail is sent to multiple lists, it may receive a different prefix.

I wrote a script which tried to take into account these small variances and sort messages into threads. The script became quite complicated, but I believe it gives a correct enough representation. One particular problem is that some threads have the same subject, but are in fact different discussions at different periods in time. This was the case for a small amount of mails in the material. The number is quite small, so rather than engage in tedious and fuzzy date proximity analysis, I opted to simply omit the duplicates. I perused those mails that were subsequently not sorted into threads and found that they were often irrelevant test messages. I believe the resulting sorting is acceptable to get interesting data.

3.6 Limitations of the research approach

In this section I discuss some of the limitations of my approach and some issues with the application of the methods.

3.6.1 Issues with the interviews

The focus of the thesis changed a lot throughout the last year. In September of 2006, the subject shifted from wikis and website design for collaboration, to general electronic collaboration. Some of the initial interviews were about website topics. In all the interviews, however I talked with the informants about collaboration in some form and about the HISP network. I many cases I have chosen not to revisit informants, but use what was still interesting from the interviews. In retrospect, I could have been more systematic both in defining questions and in checking back to informants about the new issues.

Anonymity is also a different issue. While the various informants are probably totally anonymous to people outside the HISP network, I’m fairly certain that most people inside the network will recognize them. When I talk about a coordinator from India or a very active core developer from Norway, it’s difficult for those who know them not to see through the veil of anonymization. Still, I’ve usually chosen to include a person’s role and country when referring to them, because it makes it clear which perspective the person talking has.

3.6.2 Using a case study

In this thesis I analyze one particular case, DHIS 2 development. It is not possible to generalize from this case in a positivist sense (Walsham 1995:79), i.e. from the case to a population of projects. The case in itself is rather special as well, combining odd organizational properties and somewhat conflicting processes, complicating matters further. Instead, Walsham points to four ways of interpretative generalization: Development of concepts, generation of theory, drawing of specific implications and contribution of rich insight (ibid). But even in these cases Walsham warns that it's

3.6.3 Colored by participation

I've been an active member of the project throughout the entire work with the thesis. I've contributed code, taken part in project meetings and mail discussions and socialized with other members. Through this I've also been involved in shaping the project, including the collaboration and use of tools. Additionally, I've been socialized into the project and thus see a lot from the inside. So a valid question becomes: How objective can I be, how colored is the discussion by my views and experiences. On the one hand these grant me a lot of extra insights into the projects an external observer might have, on the other hand it may make me blind to its practices. This is somewhat alleviated by the use of quantitative data. Various counts are not affected by how I feel about them. Then again, the reason for counting the way I do and how I interpret the results are. Ultimately, it comes down to my own ability to distance myself from the material, that I am critical both of my own views and the views of others. The reader will have to determine whether or not I have been successful in this.

3.6.4 In retrospect

There were some weaknesses with how consistent I was when following methodology. For example, instead of taping interviews I usually made hand written notes. In retrospect this turned out to be a bad idea. The result was that some of the nuances of what the informants were saying disappeared, and some statements went missing entirely. It also means that my access to quotes is somewhat limited. This has both methodological and esthetical implications. On the one hand it reduces the precision of the material. On the other hand it makes the text somewhat heavy to read. Statements are embedded in the text, rather than lifted up as actual statements. It is also clear that I should have been more disciplined when formulating questions consistently across interviews. This would have increased the possibility of comparing statements directly across them. Then again, as mentioned, the focus of the thesis changed significantly during the work.

On a different note I should have been more proactive in getting feedback from developers in Vietnam. The e-mail interviews provoked short, concise answers, and I could have been more insistent on getting them to go more in depth into their views. On the other hand, the reports I got indicated that they were generally happy with the tools and the processes. There was no apparent "problem" to investigate or desires for changes to explore. This means that the thesis has a lacking in the empirical material: It has a stronger focus on the view from the Oslo node, where informants were readily available. The discussion would be more varied if it could also take the views from Vietnam into account. I try to balance this by using the many master theses written about the Vietnam node as material.

4 Background

In this chapter I present the case, HISP and DHIS 2 and some background information regarding electronic tools.

4.1 HISP – Health Information Systems Programme

What is HISP? Throughout the work with the thesis, I've attempted to find an answer to this question, thinking it would help to understand how and why collaboration takes place. It's been called an action research project, a network, a service organization, a loose group and several other things. First, some facts.

4.1.1 History

HISP was started in South Africa in 1994. It was part of a series of projects aimed at transforming South African society after Apartheid. The project was partially funded by NORAD, and supported by researchers from the University of Oslo. The aim of the larger project context was to organize and integrate a post-Apartheid health system, with several vertical lines of decision making and health services exclusive for different races. As part of the HISP project, a piece of software was developed called the District Health Information System (DHIS). This piece of software figured heavily in realizing the ideological goals of HISP: A district focus, empowering users, and improving the use of information for action in health care. The activities of HISP included not only developing the software, but training people in its use and in how one can use the information produced by the software to change health care delivery in the districts. This meant working closely with the authorities on different levels in South Africa.

From having a South African focus, the project extended to other African countries. Among them are Mozambique, Malawi, Tanzania, Ethiopia, Nigeria and Zanzibar. In addition to spreading the software and the practices, HISP has also built up a research network and a series of academic programs in systems development and health care in cooperation with various local universities. The history of action research and the strong academic connection in HISP has lead to various master courses and PHD programs in the universities around the network, among them the integrated masters programs in HIS in South Africa, Mozambique, Tanzania and Ethiopia and several PhD students in sustainable HIS development from Africa and Asia. In recent times, the project has extended to non-African countries, including Vietnam and India. Building a node in Cuba was also attempted, but failed.

In 2006, HISP approached the European Union and formed the BEANISH network, Building Europe-Africa Network for applying IST in the Health care sector. As the name suggests, the focus of the network is on the relationship between Europe and Africa, while HISP has a more global focus. Central to the proposal and demands of the EU are the use of various collaborative tools for communication in the network and for sharing research.

4.1.2 Structure

Following the metaphor of a network, HISP consists of nodes. Braa et al (2004:344) defines it as being on two levels, where the first is different countries and the second is a set of institutions in the various countries. Institutions are typically universities. Braa et al (ibid) continues to define South Africa, Norway, Mozambique, Tanzania and Ethiopia as the primary nodes in the network. Since this article was written, HISP has expanded further, with primary nodes in among other places Malawi, Vietnam and India. The two last of these are also centers of DHIS 2 usage and development, which is the focus of this thesis. In India and South Africa, the national HISP teams have taken the form of not-for-profit organizations with a board. In other countries the organization varies. Often the national organization is strongly tied to an academic institution, like in Norway and the University of Oslo.

The history of action research and the strong academic connection in HISP has lead to various master courses and PHD programs in the universities around the network, among them the integrated masters programs in HIS in South Africa, Mozambique, Tanzania and Ethiopia and several PhD students in sustainable HIS development from Africa and Asia.

4.1.3 What is HISP?

HISP started out as a large scale action research project aimed at transforming health care in South Africa. It was initiated by researchers from the University of Oslo, University of Western Cape and government officials from the two countries. As an extension of the research project, there has been cooperation between different institutions about PhD and master's programs, and most of the academics in HISP are involved in different University courses around Africa and in Oslo.

The DHIS project remains central to the strategy of HISP, and thus HISP also encompasses a development organization. This part of the organization is mainly focused around the two distinct version DHIS 1 and DHIS 2, both with their own development teams. In addition HISP has contributed to several other development projects in various countries, including AIDS monitoring systems and systems for electronic patient records. DHIS is sometimes adapted or tweaked to the requirements of individual nations and districts, usually by local teams.

With the software came a need for training of users, and furthermore working with health care workers to analyze and utilize the captured information. Thus HISP encompasses a training and advisory organization, with an army of facilitators on the local levels. With the spread of HISP and DHIS to other countries than South Africa, this part of the organization has been greatly increased, with groups of facilitators in the individual countries. It can be argued that this group also does the bulk of the organization's "work", if such can be defined, working closely with the local health care workers, training them in the use of the software and working to integrate the software into the practice of the local health care institutions.

The individual countries also form nodes in what has been called the HISP network. Instead of viewing all the employees and members of the various nodes as members of the overall HISP project, it seems that the nodes have a strong individual identity. You have, for example, HISP South Africa, HISP India and HISP Vietnam. They are separate legal entities with their own employees and set of local issues. This is especially strong in South Africa and India. Both have taken the form of not-for-profit organizations. Interestingly there is no HISP Norway or HISP Oslo because there is no similar infrastructure. DHIS is not used in the Norwegian health care system, and HISP is not directly involved with the health care authorities like in the other countries.

What's interesting to note here, is that there is no legally or organizationally strong bond between the different nodes. There is no formal leadership, no formal chains of command. Informally though, this is another matter.

4.1.3.1 Who are HISP?

HISP includes researchers, health care workers, trainers, facilitators, coordinators, IS managers, programmers, masters and PhD students, and many more. It is sometimes difficult to draw a clear line between who are and who are not members. It is clear however, that there is a leadership echelon, which still consists of researchers with a base in Oslo, one particular influential South

African developer/idealist, one seasoned South African doctor, plus a range of others which are also mostly academics. These leaders oversee the movement of people between the nodes, for example sending people from Oslo to work with the local team in Vietnam. Furthermore, these leaders work together to secure funds from international organizations like the EU and WHO.

The two developers of DHIS 1 are both hired by HISP South Africa. This is different for DHIS 2, however. Here the development team consists mainly of master's students from Oslo. In addition to these, there are four employees in Vietnam and an increasing number of employees in India. The Oslo team represents perhaps the biggest grey area, where large amount of students from the INF5750 course at UiO contribute code to the software over one semester. Are these "hispers"? Probably not, as few of them participate beyond their course work and perhaps even fewer become a part of the larger community around HISP.

Finally, it can be argued that many health workers themselves are a part of HISP seeing as they work closely with facilitators from the organization. The development of DHIS 1 and 2 has been strongly user oriented, and these health workers represent the primary source of feedback on how the systems work and how useful they are. In South Africa there is a very strong contact. "Everybody" knows the lead developer, and he is consulted on everything from system functionality and data analysis to local and national health care strategies.

4.2 Previous research on HISP and DHIS

In this section I look at some of the research done on the HISP network and DHIS development. It's my intention partly to build on this research and provide some critical reflections on it. For DHIS 2 development I focus on master theses by Nordal (2006) and Nguyen (2007, in press) and partly on Gjerull (2006) and Øverland (2006). In many ways these do not constitute theory in this thesis. Instead I treat them more like empirical material, data on periods and events I didn't participate in, which helps to thicken the descriptions and inform a discussion. Additionally the more senior researchers in HISP have made some points in Braa et al (2007, in press) and Staring and Titlestad (2006).

Braa et al (2004) is in many ways the foundation article for how HISP operates, describing the startup and the strategy of the project in South Africa. The central argument in the article is the need to organize Networks of Action, scaling and spreading the HIS approach into several nodes in order to attain sustainability. The network draws on University researchers and promotes South-North and South-South cooperation.

Braa et al (2007, in press) describes some of the problems with regard to capacity building in DHIS 2 development. They develop a concept of knowledge (in)congruence to describe the differences between the different nodes. While for example the Oslo node has considerable knowledge about Java frameworks, all the Norwegian developers lack field experience with regard to health work. Similarly, people in the Vietnamese node have close contact with the field, but less knowledge of Java frameworks. They also describe some problems with the usage of the communication tools used in the development, which is directly relevant to this thesis. In many ways, this thesis confirms and extends the conclusions from that article, with an increased empirical material. In this thesis I will continue to use the concept knowledge (in)congruence as it may highlight some of the reasons for why electronic collaboration happens the way it does in DHIS 2 development.

Kristian Nordal's thesis (2006) contributes two things. Most importantly, Nordal describes the start up of DHIS 2 development and his own involvement. Nordal was one of the central characters in the early period. He took part in discussions and decisions with regard to the project's status as open source, the development languages and frameworks to be used, and, most importantly for this thesis, the choice and setup of the different communication tools used in the development. Nordal situates this process heavily in experiences from other FLOSS projects, and is perhaps the clearest proponent of FLOSS processes and tools in the project. Nordal also participated heavily in development, serving as one of the four core developers until his departure from the project in the summer of 2006. The second contribution comes from Nordal's work in Vietnam, training students and employees in DHIS 2 technologies and tools. His thesis reports extensively from his experiences in Vietnam and thus gives access to the internal workings of the Vietnam node.

Lars Helge Øverland's (2006) master thesis goes more into detail regarding the development of the Vietnamese node. He describes the various DHIS implementation efforts in the country, which spanned three different attempts. Furthermore he describes his involvement in working with and training the Vietnamese team in the use of DHIS 2 technologies. Finally he recounts some of the more recent work with DHIS 2 implementation in the country.

Thanh Ngoc Nguyen's thesis (2007) describes Nguyen's work in Vietnam and Ethiopia, offering some critical remarks on the DHIS 2 development process. Nguyen's thesis also describes some alternatives to the Java and tools approach chosen and enforced mainly by the Oslo node. The Ethiopia team developed an ART system using the PHP language, with a different development process. Similarly to Nordal, Nguyen reports his observations on the training and work process in Vietnam which Nordal and Øverland took part in. Nguyen voices some reservations and issues made by various developers not so visible in the rest of the material.

4.3 DHIS – District Health Information System

The DHIS is software for capturing, analyzing and reporting on health data. The systems deals with aggregated data, meaning totals for one organization unit, a clinic, a hospital, a rowing health employee, and so on. Data is reported and aggregated to the unit on the level above, maybe a regional clinic or an administrative unit. The basic idea is to give health workers an overview of the situation in a district or region. This overview then serves as a tool in determining where to put in extra resources and efforts.

Workers at the clinic capture Data Values for a particular Data Element, for example the "Number of inoculations performed", for a particular Period, for example the last month or week. This is summarized in the following table:

Concept	Example
Organization unit	Ugga PHC
Data element	Number of inoculations performed
Period	May 2006
Data value	22

Data elements are organized in Data Sets, a collection of related Data Elements, for example different those related to a specific report. In the system the user is presented with all the Data Elements in one selected set and enter the collected data.

There are three kinds of data collected: Routine data, semi-permanent data and survey data. Routine data is featured in the example above and collected regularly, for example each month. Semi-permanent data are for example the number of beds in a facility, the population in a district, and so on. These values do not change as often and are collected at longer time intervals than routine data. Finally, survey data are data gathered in questionnaires, for example patient satisfaction surveys. Captured data are reported upwards, where they are aggregated in a series of reports. The reports are often electronic versions of existing paper reports.

The development of the software is based on the idea of user participation and user empowerment. From the beginning, health workers have been part of shaping the software. It is also designed in a flexible way, allowing users themselves to define metadata in the system. This includes the organization units and their hierarchy, which data elements data should be registered for, report design and which indicators should be used to measure progress. This allows districts and regions to tailor the information they want to analyze.

DHIS is developed in two major versions: DHIS 1.x and DHIS 2, under slightly different circumstances. DHIS 1.x, development is done by one team of developers, in close collaboration with users in South Africa. The project was extended to include other countries, but there are no developers in these countries. User contact is mediated by the facilitators which help with the implementation of the software into the organizations of the users. HISP coordinators manage the activity in specific nodes. The challenge has been compiling requirements from the different countries and developing software based on these requirements in one node, South Africa. This is illustrated in the following table:

Nodes/Roles	Implementation	Administration	Development
South Africa	Users, Facilitators	Coordinators	Developers
Malawi	Users, Facilitators	Coordinators	
Mozambique	Users, Facilitators	Coordinators	

A small note: As HISP is an action research project, one could say that health research and education should be in this table as well, but this is outside the scope of this thesis.

With the establishment of the DHIS 2 project, one of the goals was to distribute the development into the different nodes. Oslo took charge in establishing the teams and the development process, but the software is not used in this country. This is illustrated in the following table:

Nodes/Roles	Implementation	Administration	Development
Norway		Coordinators	Developers
Vietnam	Users, Facilitators	Coordinators	Developers
India	Users, Facilitators	Coordinators	Developers
Ethiopia	Users, Facilitators	Coordinators	Developers

Oslo as a driving force introduces one challenge: As the software is not used here, the node has no direct connection to the use of the software or the health domain. It must rely on the other nodes for information. The second challenge lies in the distributed development itself: As there are developers in different nodes, there is a stronger need to coordinate the activities between them. This last point is especially in focus in this thesis.

4.3.1 DHIS 1.3 and 1.4

DHIS 1 comes in two major versions: 1.3 and 1.4. Common for the 1.x versions is that they are based on MS Access. They are developed in a combination of the MS Access graphical interface and Microsoft Visual Basic. The basis for choosing this technology was that Microsoft products were already in widespread use in South Africa, which meant these were tools health workers were already familiar with.

Over time, DHIS 1.x has gotten a large amount of functionality, including a powerful mechanism for defining data elements, data sets and the like, aggregation capabilities, a flexible report tool and export functionalities. There has also been work on extending DHIS into the patient realm through a Patient module.

The special aspect of the development done in South Africa is the fact the there are only two developers. Development is done partly through designing user interfaces in the MS Access and VB applications, and partly writing snippets of code in VB or SQL for Access. The result is that there is no source code per se, only precompiled binary files in a proprietary format. The developers are based geographically close to each other in the city of Cape Town. Coding is sometimes done in concert, where they meet at one of the developer's house, or by working apart. In this latter case they often mail each other whole updated modules, and overwrite their local copies. A large amount of close coordination is necessary to make this work. They need to decide in advance who works on different modules. Otherwise they may end up overwriting each other's work. Coordination either happens through being physically present in the same room, by telephone or by e-mail.

DHIS 1.x has been described as an "onion" of interconnected layers, making maintenance and distributed development difficult. This was one of the reasons why a new version of the software was considered.

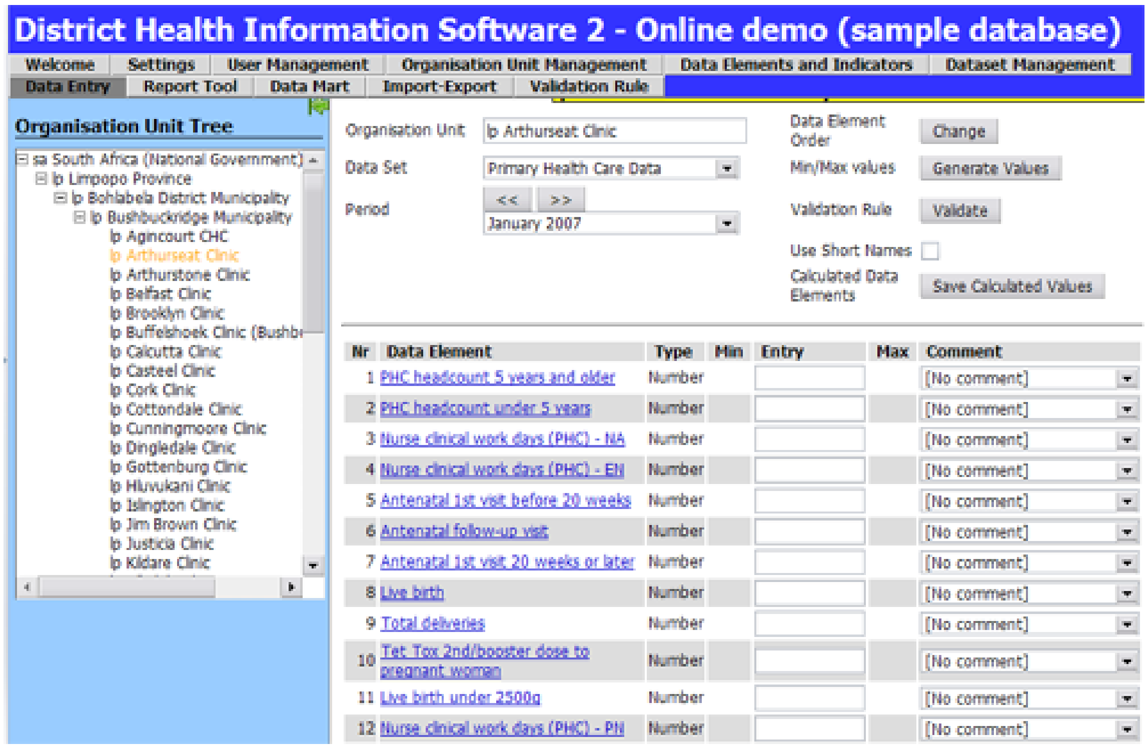
In the context of this thesis, the South African case serves as a sort of base case, a source of comparison with how things are done when developing DHIS 2.0. The extreme case would be one developer working alone, where no coordination is necessary. In this case there are two. The type of "source code" complicates how the coding itself can be coordinated and distributed.

4.3.2 DHIS 2.0

When a new version of the DHIS software was considered, a decision was made to base it entirely on Free and Open Source technologies. The Java programming language was selected for development, probably because the first DHIS 2 developers had extensive experience with this language. Furthermore the decision was to make the system web enabled, i.e. that it could be run on a web server and accessed from a web browser. Because of this, it would be possible to install and maintain the application in one place, and let users communicate with this instance over the web. Another early requirement was to make the system database independent so that the user organization could choose which underlying database to use.

The resulting system is based on a series of Java frameworks. The most important are Hibernate, Spring and Webwork. Hibernate handles persistence in a database, and abstracts away which database is being used. This allows the installer to for example select either PostgreSQL or MySQL for the job. Spring decouples classes and allows for quick and easy configuration and modularization of the system without the need to recompile the code. Webwork is the web platform of the system, used as a basis for creating a web based user interface.

The system is strongly modularized, allowing different actors to develop their own modules, tailored to their specific needs. Through the configuration possibilities, a local administrator may also pick and choose which modules to display to the users, for example hiding modules relating to maintenance.



Organisation Unit Tree

- sa South Africa (National Government)
  - lp Limpopo Province
    - lp Bohlabela District Municipality
      - lp Bushbuckridge Municipality
        - lp Agincourt CHC
        - lp Arthurseat Clinic
        - lp Arthurstone Clinic
        - lp Belfast Clinic
        - lp Brooklyn Clinic
        - lp Buffelshoek Clinic (Bushbı
        - lp Calcutta Clinic
        - lp Casteel Clinic
        - lp Cork Clinic
        - lp Cottondale Clinic
        - lp Cunningmoore Clinic
        - lp Dingledale Clinic
        - lp Gottenburg Clinic
        - lp Hluvukani Clinic
        - lp Islington Clinic
        - lp Jim Brown Clinic
        - lp Justicia Clinic
        - lp Kildare Clinic

Organisation Unit	lp Arthurseat Clinic
Data Set	Primary Health Care Data
Period	January 2007
Data Element Order	
Min/Max values	
Validation Rule	
Use Short Names	☐
Calculated Data Elements	

Nr	Data Element	Type	Min	Entry	Max	Comment
1	PHC headcount 5 years and older	Number				[No comment]
2	PHC headcount under 5 years	Number				[No comment]
3	Nurse clinical work days (PHC) - NA	Number				[No comment]
4	Nurse clinical work days (PHC) - EN	Number				[No comment]
5	Antenatal 1st visit before 20 weeks	Number				[No comment]
6	Antenatal follow-up visit	Number				[No comment]
7	Antenatal 1st visit 20 weeks or later	Number				[No comment]
8	Live birth	Number				[No comment]
9	Total deliveries	Number				[No comment]
10	Tet Tox 2nd/booster dose to pregnant women	Number				[No comment]
11	Live birth under 2500g	Number				[No comment]
12	Nurse clinical work days (PHC) - PN	Number				[No comment]

Figure 4.1: Screenshot of DHIS 2.0

4.3.3 Organization

DHIS 2 development is organized as a distributed software development project. There are four nodes or teams involved: Norway (usually referred to as 'Oslo'), Vietnam, India and Ethiopia. In thesis I limit my focus to the three nodes which have been most involved in the development: Norway, Vietnam and India. The project is lead by a project manager who is situated in Oslo. The lead developer on DHIS 1.x functioned as an advisor in the early phase of the project, helping to explain requirements and describe the functionality in the previous version. DHIS 1.x is treated as the requirements specification for 2.0, having been developed in close collaboration with users in the field since its inception.

HISP has a limited budget, so communication is mostly done through electronic means. The most important coordination mechanism is the developers' mailing list, which all of the developers and most of the coordinators involved in the project subscribe to. The project's wiki is used for documentation and for presenting the system to outsiders. Instant messaging is used for meetings and between individual developers and coordinators. However, there has also been a significant amount of travel involved. Developers from Oslo have traveled to Vietnam, India and Ethiopia, developers from Vietnam have assisted in Ethiopia and representatives from Ethiopia and Vietnam have taken courses in Oslo.

The majority of development work has been done by a team of about 7–9 developers in Oslo, whereof 4–5 are most active. These are all or have all been master students at the University of Oslo. The INF5750 master's course on Open source development using Java at the University has also been a source of concrete work on the software. Students in the course have effectively designed and implemented modules for the DHIS 2 software as assignments in the course. The course is at the time of writing running in its third iteration. Furthermore, the course is used as the primary recruiting ground in Norway. The project managers try to recruit students to do further work on DHIS 2, for example as part of their master's thesis.

Additionally, there are development teams in Vietnam and India especially. In both these nodes, the developers are usually hired as employees. In Vietnam, HISP has worked closely with a local university and recruited from a pool of students. Vietnam in particular has contributed to the core modules in the system, albeit not at the same rate as developers from the Oslo team. I will return to this issue later in the thesis. These two teams have, however, worked actively on specific modules in order to satisfy specific requirements in their respective countries. Both teams have worked on report modules which emulate paper reports in their countries' health organizations. This seems to be a response to the Oslo team failing to supply a general report module which satisfies their requirements in short enough time.

The project is managed through a roadmap, a document which gathers feature requests from different sources. These are divided among milestones, which are stable releases of the software. At the time of writing the current release is M7. At irregular intervals the project managers will call together a meeting which decides the future course of the development. In the beginning these meetings were face-to-face in Oslo, as most of the developers were working here, and because the focus was on supplying the same functionality as in the previous releases of DHIS. As the software has seen more use in India and Vietnam, this practice has changed to electronic meetings, using instant messaging clients like Skype and MSN. Coordinators and other interested parties from these countries have been invited and participated in the meetings.

The developers are assigned tasks as a result of a negotiation process. The administrators will suggest that a developer takes on a specific set of functionality, and the developer will respond with whether or not he or she has time. This has also resulted in a sort of quasi-responsibility for modules. One module of the software is often written by one, or sometimes two developers. This person thus gets a kind of guru status on that piece of code. This has never been formalized in any documents, however.

4.3.4 The case of Oslo

In this section I describe the history and activities of the Oslo node in relation to DHIS 2 development.

Oslo has been the primary seat for DHIS 2 development. The coordinators of the project come from Norway. The initial development of DHIS 2 was done by Norwegian master students. Those active in Norway can be divided into three main groups: Core developers, developers and students.

In 2006 the project leaders defined a group of core developers, consisting of the most active developers. All of them had been a part of the project from the start and had already contributed large amounts of code on the core modules. Consequently, these developers are also the ones who are most knowledgeable of the core structure and the different frameworks being used. They are entrusted with important modules and with the overall maintenance of the system. As Nordal notes (2006) some technical decision-making power was also delegated to this group. The reasoning was that the developers would have more competence to make decisions regarding purely technical matters. Subsequently, how these developers solve their tasks, which concrete programming solutions they choose, is rarely called into question by non-technical participants.

Around these, there is a more peripheral group of developers. They participate regularly and often have responsibility for modules, but not on as regular a basis as the core developers. They have a fairly good knowledge of the frameworks and lean on the core devs for guidance. I count myself as being a member of this group. At the time of writing there are six such active developers.

The more peripheral group consists of those students who participate in DHIS 2 development during the INF5750 course at the University of Oslo. They do their work on DHIS while taking the course, but often have no interest in or time to contribute further to the project. By definition, these terminate their activity within DHIS and HISP once they complete the course. If not, they would typically move into the developer group.

Finally, Oslo also hosts much of the academic work in the HISP network through a range of professors and PhD students. The Oslo node has a long tradition for this stretching back to the initial startup of HISP together with South Africa. The most prominent leaders in the HISP network also have their offices at the University of Oslo.

4.3.4.1 The INF570 course

The DHIS 2 activity in Oslo is closely tied to the course INF5750 – Open Source Java Frameworks in Global networks at the University of Oslo. The course focuses, as the title implies, on various Java frameworks, and situates them within global software development. The course has had three main parts: The first part focuses on using various development tools and Java frameworks. These are the same as those being used in DHIS 2 development: Hibernate and Spring, and tools like Subversion, Maven and Eclipse. The students are guided through a series of tough obligatory assignments. The second part has been a general introduction to the workings and motivations of Open Source, focusing on licenses, business models and various case study projects. The third part, taking up half the course is practical work using the frameworks in a global software development project: None other than DHIS 2.

The first run of the project occurred after the startup of the project, while one of the core developers did his stint in Vietnam. One of the Norwegian coordinators of the project ran the course. The students were divided into groups, charged with developing the initial modules for DHIS 2. Here they were exposed to and told to use various communication tools like the developer mailing list, the wiki and the JIRA issue tracker. The work also entailed eliciting requirements from the older versions of DHIS. This often meant contacting developers in South Africa and various users for input. Furthermore, the Norwegian master student was working on the core part of the system. The endeavor was only partially successful: Most of the module never reached a usable state. Following the first run of the course, the most active students were recruited as the initial developers to continue work on the system. Out of these, two developers later became core developers along with the master student in Vietnam and one of the teaching assistants of the course.

In the second iteration, two core developers and I returned as teaching assistants and the second Norwegian coordinator took the reins. The technical focus of the course remained the same, and the FLOSS aspect was raised more clearly as a subject. As the last time, students were divided into groups working on modules of DHS 2 or various supporting activities. The projects had varying degrees of success. Once again, the students were required to use the various communication tools. Out of these students, an additional three developers were recruited.

The third iteration reformed the technical focus somewhat, but didn't change the selection of frameworks or tools. One core developer and I reprised our roles as teaching assistants, joined by another active developer. The group work now resulted in at least two modules which have since been included in the main release of DHIS 2, with at least another two modules awaiting implementation.

As we have seen, the course represents the primary recruiting ground for the project. It is also a source of a considerable development effort, even though relatively little of the code has actually made it into the software. What is perhaps more interesting is the normative focus on the use of the communication tools, which may help explain why the Norwegians are so active in using them. More on this later.

4.3.5 The case of Vietnam

In this section I give a quick overview of the history and activities of the Vietnam node. As I have not had the chance to visit the country and do much first hand data gathering, I rely mainly on reports from the capacity building efforts detailed more closely in contributions by Øverland (2006), Nordal (2006) and Ngyuen (2007).

The Vietnam node was the first to get involved in DHIS 2 development. As noted by Øverland (2006) developing the system solely with developers situated in Oslo would be difficult due to the distance between users in Vietnam and developers in Norway. This lead to a desire to establish a more permanent development team in Vietnam. The establishment of the Vietnamese team was closely tied to various efforts at implementing DHIS 1.3, 1.4 and 2.0 in the country. In July 2004 the Vietnamese government approached HISP as an extension of their Master Plan developed in March 2004: Applying and Developing Open Source Software in Vietnam for the 2004–2008 period.

Øverland (ibid) notes that the implementation effort was spread over three main periods. In the period from the initial contact and into the fall the same year, Vietnam and HISP worked out the formalities for an implementation plan. In this period, HISP forged ties with a Vietnamese IT company, referred to as OutSoft. The first implementation process involved piloting version 1.3 in two districts in Ho Chi Minh City (HCMC). During this period the project was aided by three interns from OutSoft and lead by a Norwegian master student. This process failed for several reasons, among them lack of support personnel, lacks in the DHIS 1.3 software and problems with the Vietnamese infrastructure (Øverland 2006:49). Communication with OutSoft eventually broke down.

The second effort began in November 2005 and involved attempting to use DHIS 1.4 instead. This was a two step process, first involving merging DHIS and a local system, and then replacing these installations with an upcoming DHIS 2.0 release. The pilot effort was extended to include districts in Hue and Hanoi. Again, support personnel were lacking and the interns, favoring FLOSS technologies, had qualms with working DHIS 1.4's Microsoft based technologies. The development of DHIS 2.0 and a "light" version of the software also accelerated, resulting in the abandonment of the 1.4 pilot in December 2005. From the second effort, HISP attempted to secure a partnership with a local university, using local students as developers on DHIS 2. Following an initial student project, HISP employed four students in May 2006, which have since formed the backbone of the development team.

The third phase began in July of 2006, using a build of DHIS 2.0. This phase is currently ongoing. It has focused on implementation, overseeing a wide-scale spreading of the software to some 30 districts. This process was overseen by the development group of students, and assisted by three Norwegian master students.

As we have seen, Vietnam has been the target of the biggest capacity building effort among the nodes. A total of nine master students have traveled to the country and assisted in training local developers and assisting the implementation efforts. Three of the master students are core developers on the project. Another three have been active developers on the project.

Øverland (ibid) notes that since the release of DHIS 2, the development have increased their activity, but focusing on addressing local requirements and adjusting the software to the Vietnamese context, rather than global development. However, as we shall see, the Vietnamese team has also been involved in work on the core. I shall return to this point in the analysis of my empirical material in Chapter 5.

During the spring of 2007 the members of the development team quit following a long term conflict regarding their contracts. Two of these have expressed interest in contributing to the project, but this effectively sets the team back to square one. Following this, two new developers were hired, and have begun training. These have been assisted by one Norwegian master student which has remained in Vietnam. Time will tell how this new development team is involved in the global development activities.

All the contributions point to various problems with the capacity building. One problem is that Vietnamese students seem to be somewhat passive, as a result of the workings of the Vietnamese school system. Øverland (ibid:81) recounts that the students "expressed that they were not used to solving problems on their own without strict guidelines and requirement specifications". Øverland connects this to how the education system functions:

> *"A traditional educational scheme based on students passively receiving information from their teacher is dominating. Teamwork and practical tasks are rarely included in the curriculum. This gives the students poor conditions for building up abilities in individual problem solving, reflection and independent learning"* (ibid:87f).

4.3.6 The case of India

In this section I give a quick overview of the history and the activity of the India node. As opposed to Vietnam, there is not much written about the work of this node. I visited the central Delhi office of HISP India during the summer of 2007. HISP India is organized as a not-for-profit organization, supplying health care software and training to various states. It has a president and a national coordinator, in addition to several state coordinators.

HISP India bases its activity on the various states. At the time of writing, HISP India is present in 6 states: Jharkand, Karnataka, Andra Pradesh, Madhya Pradesh, and Chattisgarh. The presence varies from piloting the DHIS 2 software in a few districts to full state wide coverage. There was also a presence in Gujarat state, but the organization was thrown out when political circumstances changed. The activity started in Chittoor, Andra Pradesh in December 2000. The first rollouts were done with DHIS 1.3. The software was used and scaled up in that state all through 2004. 1.4 was chosen for a new project in Kerala in 2005. DHIS 2 later replaced this installation and was slowly scaled up into the current state presence.

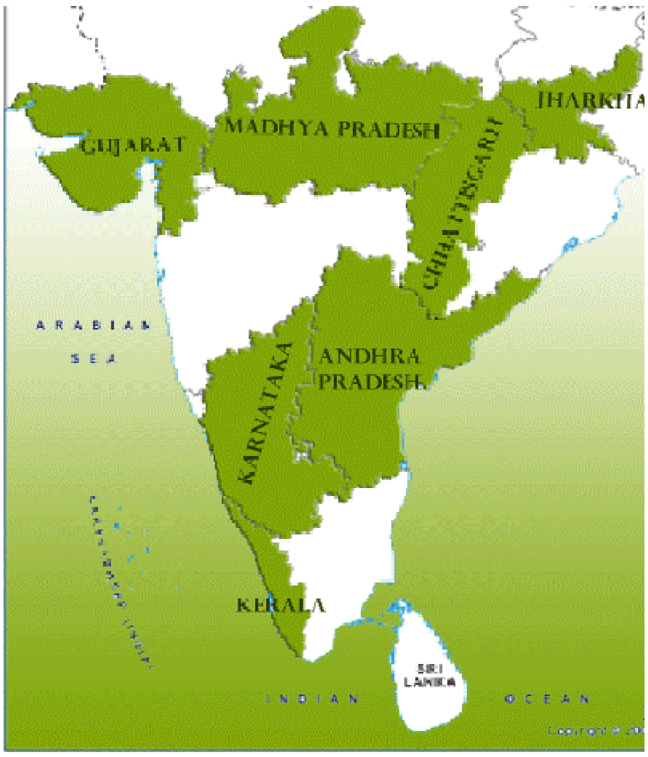


Figure 4.2: Picture of Indian states with a HISP presence. From hispindia.org

The India node is made up of several facilitators which are on the ground to assist implementation. They arrange courses, help with data entry and validation and give advice on how to use ICT for improving health care. Additionally, India has a development team which develops mostly local functionality: A Graphical Analyzer module and local reports for India. Each state should have a technical coordinator and a state programmer.

Today HISP India is the primary source of feature requests and bug reports, as this is also the location where DHIS 2 is in most widespread use. This has introduced some challenges in balancing requirements for India, which have ongoing and critical needs, with developing DHIS 2 for use in other countries.

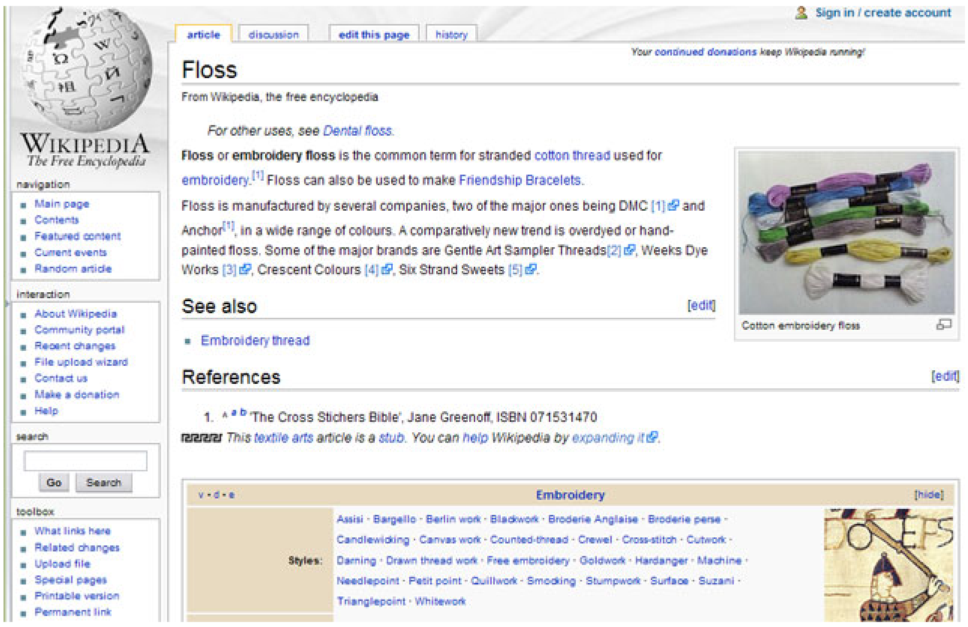
As we shall see, India has developed a strong internal communication infrastructure. They have a website with lots of information, http://www.hispindia.org, an internal mailing list, and recently an internal issue tracking system and project management tool. HISP India is also involved in the development of several other systems besides DHIS 2. At the same time, the technologies used in DHIS 2 are not common in either higher education or the commercial market, meaning it's difficult to hire people to work on DHIS 2 locally.

4.4 Tools for collaboration

In this section I'll outline how some technologies for collaboration used in DHIS 2 work. Note that there are many other tools and variants.

4.4.1 Wikis: Web pages editable by anyone

A wiki is a collection of web pages which may be edited by the reader. Normal web pages are usually authored by an author, and then transferred to a web server. Once there, it remains a static document until the author chooses to change it and upload a new version. Web pages and related technologies have driven the development of the web, making everyone their own publisher with little or no expenses. Wikis take this development further by allowing anyone to co-author the same document, using nothing more than a web browser. Each wiki page has an "Edit"-link, which will present the user with a text area containing the text on the page. The user may edit the text and save the changes, making them available to everyone else.



Floss

From Wikipedia, the free encyclopedia

For other uses, see Dental floss.

Floss or **embroidery floss** is the common term for stranded cotton thread used for embroidery.[1] Floss can also be used to make Friendship Bracelets.

Floss is manufactured by several companies, two of the major ones being DMC [1] and Anchor[1], in a wide range of colours. A comparatively new trend is overdyed or hand-painted floss. Some of the major brands are Gentle Art Sampler Threads[2], Weeks Dye Works [3], Crescent Colours [4], Six Strand Sweets [5].

See also [edit]

- Embroidery thread

References [edit]

1. ^ a b 'The Cross Stichers Bible', Jane Greenoff, ISBN 071531470

This textile arts article is a stub. You can help Wikipedia by expanding it.

Embroidery

Styles: Assisi · Bargello · Berlin work · Blackwork · Broderie Anglaise · Broderie perse · Candlewicking · Canvas work · Counted-thread · Crewel · Cross-stitch · Cutwork · Darning · Drawn thread work · Free embroidery · Goldwork · Hardanger · Machine · Needlepoint · Petit point · Quillwork · Smocking · Stumpwork · Surface · Suzani · Trianglepoint · Whitework

Figure 4.3: A collaboratively authored article from Wikipedia

Web pages are usually written in HTML – the HyperText Markup Language. This markup language is used to mark parts of the page as headers, paragraphs, lists, tables and the like, using the markup to describe semantically how the page is structured. This HTML is then interpreted by a special program, a web browser, which will render the content in a visual way. An example is to render a semantically marked up list of elements, as a set of bullet points on the screen.

```
<ul>
  <li><a href="index.html">Main page</a></li>
  <li><a href="download.html">Downloads</a></li>
  <li><a href="docs.html">Documentation</a></li>
</ul>
```

Figure 4.4: A navigation menu in HTML. An unordered list (ul) of list elements (li), containing links (a).

There are a range of supporting technologies for writing web pages, editors like Dreamweaver and WYSIWYG (What You See Is What You Get) editors like FrontPage. Additionally there are Content Management Systems like Joomla and Plone, which are used to build websites rather than repositories of information. And even with these tools, writing web pages by hand remains relatively tedious. Wikis attempt to simplify this by introducing a special syntax which is more natural to casual authors, called APT or Almost Plain Text. Instead of markup up a list with several ul and li tags around the list elements, the user may write an actual list of bullets, using the asterix '*' character.

```
* [index.html|Main page]
* [download.html|Downloads]
* [docs.html|Documentation]
```

Figure 4.5: A navigation menu in Textile, an APT format used in the Confluence wiki. The list is created using asterixs and the square brackets delimit a link.

The idea behind this simplified language is to lower the barriers use: Non-technical or casual users are invited to participate in adding content to the wiki pages.

4.4.2 Mailing lists – The bread and butter of FLOSS development

A mailing list is software which spreads a mail to a list of recipients. Rather than send a mail with the individual addresses of the recipients in the To-field, the user can instead type in the address of the mailing list. Sending a mail to this address will resend the mail to all the participants on the list. In this way the list can be used for large scale announcement or for discussions between several people. The last part of the title is a reference to a statement by Karl Fogel (2005), emphasizing mailing lists as the most important medium of communication in FLOSS development projects.

4.4.3 Source code management

Source code management (SCM) is a system where all the source code in a project is stored on a central server, called a repository or repo. Individual developers check out the code from the repository and save a working copy on their local computers. A developer would then change the code in some way, adding or modifying features, and then check in, or commit the changes made back into the repository. Other developers may then update their working copies with the changes made, and commit changes of their own. The source code management tool handles this by transmitting and storing file differences, or diffs, on the server. Each committed change is stored as a separate revision, eventually a graph of revisions to different files. This enables developers to retrieve older revisions, and rollback changes made to the code.

Examples include Subversion (SVN), used in DHIS 2 development, and the older Concurrent Versions System (CVS).

4.4.4 Instant messaging clients

A last, but also useful technology to mention is the use of instant messaging (IM) clients and, in some cases audio and video conferencing tools. An instance messaging client, like MSN or Yahoo! Messenger maintains a list of the user's contacts. Contacts are displayed as online, offline, busy or similar. When online, the user can type quick messages and engage in a chat with one or more of the contacts. This allows for synchronous communication, as opposed to for example e-mail and wikis. Audio and video conferencing tools like Skype allows users to talk contacts using a microphone and see contacts using a web camera.

4.4.5 Issue trackers

An issue tracker allows developers or users to register bugs, make feature requests or define tasks to be done, in relation to the development. Such tasks or requests are called issues, or in some cases tickets. The issue tracker is a project administration tool to ensure different tasks get done, as well as contributing to planning. Each issue can be assigned to a specific person and given a priority. In many issue trackers, you can also define components, milestones, release dates and so on. Issues may be opened, assigned, reassigned, closed, etc. In some issue trackers you can also report progress on specific issues. All issue trackers offer some sort of reports on which issues are pending, which are completed, the percentage and so on.

4.5 Collaborative tools in DHIS 2

In this section I quickly describe the various collaborative tools being used in DHIS 2 development. In Chapter 6 I continue to describe their use more closely.

4.5.1 DHIS 2-related mailing lists

First and foremost the project has a series of mailing lists, the most important being a developers mailing list, dhis-dev@hisp.info. In the rest of this thesis I shall refer to this list as the “dev-list”. Here developers from the various nodes come together and discuss technical solutions, ask questions, coordinate work and more. The list has also served a dual role as the primary channel for support. Developers and administrators with questions ask them on this list. During the work with the thesis the list system was migrated and new lists were created to replace old ones. The new ones are currently completely equivalent. The dev-list is one of the main sources of the data in this thesis. Note that it’s the old list, dev@dhis.hisp.info and its archives that are subject to analysis.

All revisions in the project’s source code repository are reported onto an SCM-list, dhis-scm@hisp.info. This allows developers to track changes to the code without consulting the repository itself. Those who have been subscribed to the dev-list are also subscribed to this list. It is also a source of some coordination as we shall see.

Similarly, all changes to the project’s JIRA issue tracker are sent onto a list dhis-jira@hisp.info. At the start of the project the list was active, but recently some changes on the server stopped mails from going through.

The users-list at dhis-users@hisp.info was originally created to build up a user community around DHIS 2. Another idea was to channel support activity onto the list. The list has basically not seen any activity since its inception. It was inspired by a similar list in South Africa where health professionals come together not only to discuss DHIS, but general health policy issues.

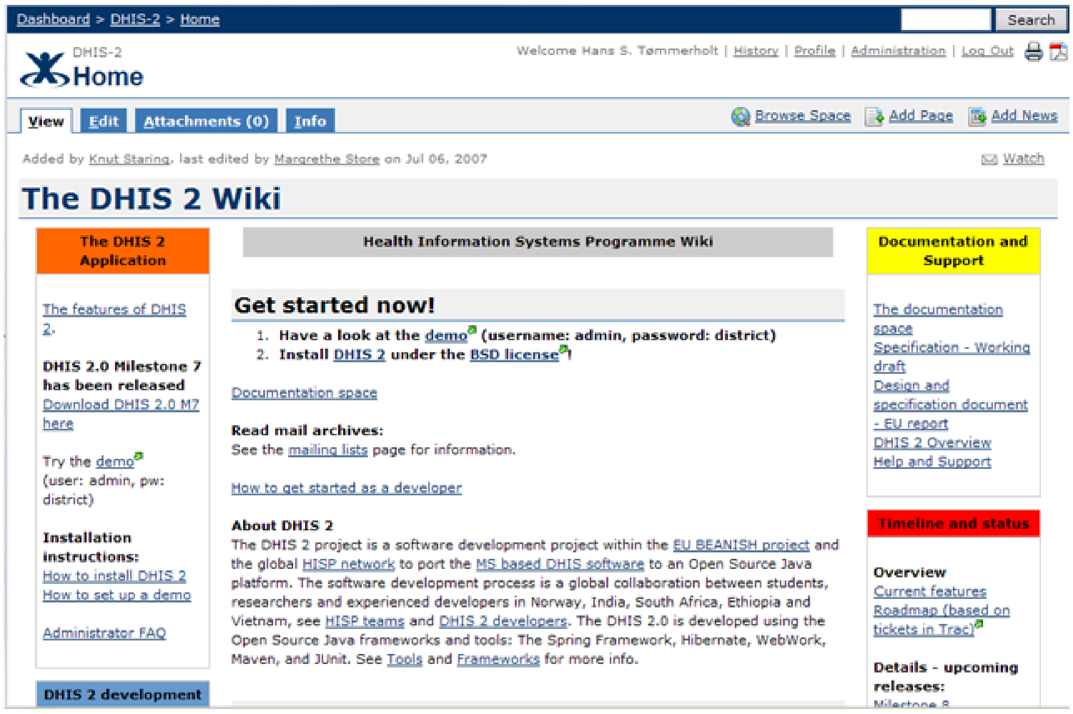
The HISP-list, hisp@hisp.info is supposed to function as a communication channel for the wider HISP network. The list has seen some sporadic activity, but mainly from Norwegian users. Instead it seems that the network coordinates through mails sent to selected users, rather than to a list.

There are other, more peripheral lists which should be mentioned. HISP India has its own mailing list for coordination internally. After a milestone meeting between the developers, it was decided to create a coordinators list, for more high level discussions than the dev-list. At the time of writing it has not seen any activity.

All these lists are archived online, which should make it easier to find information, and definitely makes them easier to analyze.

4.5.2 The HISP wiki

The HISP network has a Confluence wiki which resides on http://www.hisp.info. The software is created by Atlassian and acquired under an Open Source license, meaning that it’s free for projects that are Open Source. It started mainly as a site for DHIS 2 development, but has since grown to become the defacto website for the HISP network itself. Today it houses information on the BEANISH project, HISP’s EU connection. The wiki has often been criticized for being badly structured, especially with regard to all the information on DHIS 2 it contains. An attempt at improving on this was to create a separate space for documentation.


Dashboard > DHIS-2 > Home

Search

DHIS-2

Home

Welcome Hans S. Tømmerholt | History | Profile | Administration | Log Out

View | Edit | Attachments (0) | Info

Browse Space | Add Page | Add News

Added by Knut Staring, last edited by Margrethe Store on Jul 06, 2007

Watch

The DHIS 2 Wiki

The DHIS 2 Application

The features of DHIS 2.

DHIS 2.0 Milestone 7 has been released
Download DHIS 2.0 M7 here

Try the demo
(user: admin, pw: district)

Installation instructions:
How to install DHIS 2
How to set up a demo

Administrator FAQ

DHIS 2 development

Health Information Systems Programme Wiki

Get started now!

1. **Have a look at the demo (username: admin, password: district)**
2. **Install DHIS 2 under the BSD license!**

Documentation space

Read mail archives:
See the mailing lists page for information.

How to get started as a developer

About DHIS 2
The DHIS 2 project is a software development project within the EU BEANISH project and the global HISP network to port the MS based DHIS software to an Open Source Java platform. The software development process is a global collaboration between students, researchers and experienced developers in Norway, India, South Africa, Ethiopia and Vietnam, see HISP teams and DHIS 2 developers. The DHIS 2.0 is developed using the Open Source Java frameworks and tools: The Spring Framework, Hibernate, WebWork, Maven, and JUnit. See Tools and Frameworks for more info.

Documentation and Support

The documentation space
Specification - Working draft
Design and specification document - EU report
DHIS 2 Overview
Help and Support

Timeline and status

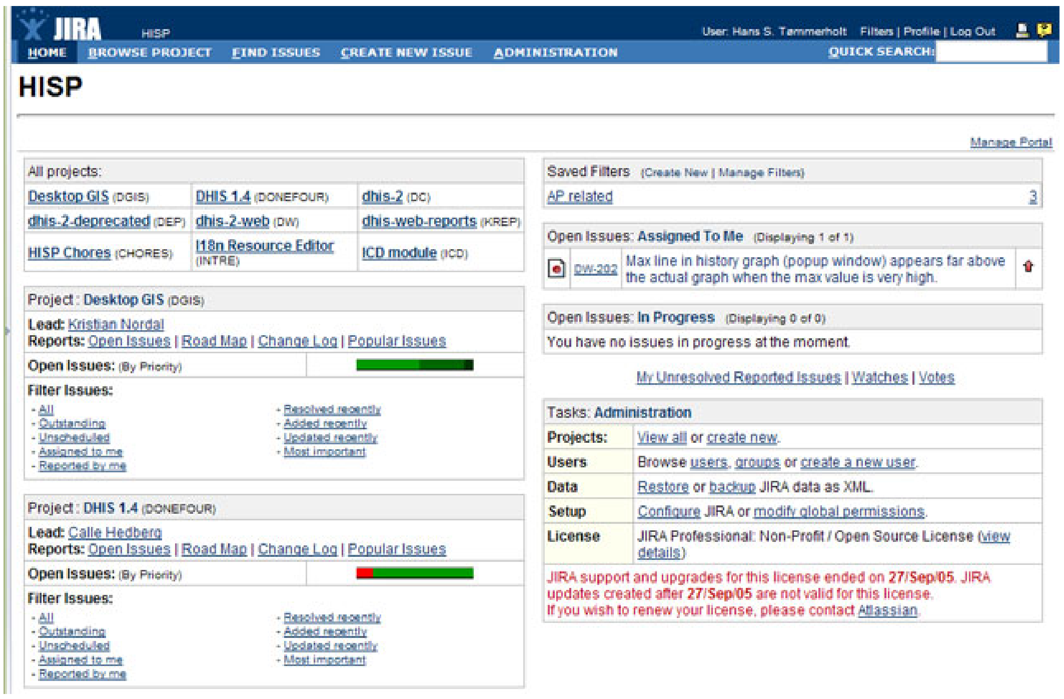
Overview
Current features
Roadmap (based on tickets in Trac)

Details - upcoming releases:

Figure 4.6: Screenshot of the hisp.info Confluence wiki

4.5.3 The JIRA issue tracker

Hisp.info runs an issue tracker for the DHIS 2 project. JIRA is also created by Atlassian and offers integration with the Confluence wiki. The issue tracker is used to register bugs and requests for functionality. It was in extensive use by the core developers in the early days of the project, and during the first run of the INF5750 course, each project received its own area in the tracker. Today, the tracker is not being used at all, probably because mail integration has recently failed.


JIRA HISP

User: Hans S. Tømmerholt Filters | Profile | Log Out

HOME | BROWSE PROJECT | FIND ISSUES | CREATE NEW ISSUE | ADMINISTRATION | QUICK SEARCH:

HISP

Manage Portal

All projects:		
Desktop GIS (DGIS)	DHIS 1.4 (DONEFOUR)	dhis-2 (DC)
dhis-2-deprecated (DEP)	dhis-2-web (DW)	dhis-web-reports (KREP)
HISP Chores (CHORES)	I18n Resource Editor (INTRE)	ICD module (ICD)

Project : Desktop GIS (DGIS)
Lead: Kristian Nordal
Reports: Open Issues | Road Map | Change Log | Popular Issues
Open Issues: (By Priority)
Filter Issues:
- All
- Outstanding
- Unscheduled
- Assigned to me
- Reported by me
- Resolved recently
- Added recently
- Updated recently
- Most important

Project : DHIS 1.4 (DONEFOUR)
Lead: Calle Hedberg
Reports: Open Issues | Road Map | Change Log | Popular Issues
Open Issues: (By Priority)
Filter Issues:
- All
- Outstanding
- Unscheduled
- Assigned to me
- Reported by me
- Resolved recently
- Added recently
- Updated recently
- Most important

Saved Filters (Create New | Manage Filters)
AP related 3

Open Issues: Assigned To Me (Displaying 1 of 1)
DW-202 Max line in history graph (popup window) appears far above the actual graph when the max value is very high.

Open Issues: In Progress (Displaying 0 of 0)
You have no issues in progress at the moment.

My Unresolved Reported Issues | Watches | Votes

Tasks: Administration	
Projects:	View all or create new.
Users	Browse users, groups or create a new user.
Data	Restore or backup JIRA data as XML.
Setup	Configure JIRA or modify global permissions.
License	JIRA Professional: Non-Profit / Open Source License (view details)

JIRA support and upgrades for this license ended on 27/Sep/05. JIRA updates created after 27/Sep/05 are not valid for this license.
If you wish to renew your license, please contact Atlassian.

Figure 4.7: Screenshot of the hisp.info JIRA issue tracker

4.5.4 The Trac issue tracker

During the fall of 2006, a project group looked at setting up a different issue tracker: Trac. The new tracker was introduced in the early summer of 2007, intended to completely replace the JIRA issue tracker. Since the installation, some 120 tickets have been registered in the system.

4.5.5 Instant messaging in the project

Instant messaging is widely used, and most of the developers and coordinators have IM accounts. The problem is that different IM clients are in use in the different countries: Yahoo! Messenger, MSN, GTalk and Skype. During the summer of 2007, an IRC channel was also set up to support a more stable place for developers to meet.

5 Empirical material

This chapter is devoted to my empirical material. I begin by describing my participation in the project. I continue to present data from the developer mailing list and the source code repository for DHIS 2. This date is projected along various dimensions such as country, role and time. First I look at quantitative analysis of the mailing list and the repository. The next section analyzes discussions on the mailing list. I then look at what is being discussed, putting the mails into categories by tagging them with keywords. The next section presents my interview data and observations with regard to the different tools and practices of the countries involved in development. Finally, I summarize the most important findings from the data.

5.1 My participation

Part of my knowledge about the case stems from the fact that I've been an active participant on the project for the last two years. I joined the project during the first iteration of the INF5750 course in January 2005. At this point the basic communication structures for the project had already been created: The mailing list, the Subversion repository, the Confluence wiki and the JIRA issue tracker. I was not involved in choosing them or setting them up. I did, however, contribute heavily in discussions regarding how to use the tools.

Unlike many of my Norwegian colleagues, I have never done field work with the project. It was common for master students to visit one of the other nodes and do work with the local teams. This way they'd also get some experience with how the software was being used in the field. During the early stages of the work with the thesis, I felt that doing field work with regard to HIS was not important to my thesis. And by the time it did come up as important, it was too late in the process. I did do some traveling, however, two weeks to a developers' workshop and conference in South Africa in the fall of 2006 and a two week visit to India during the summer of 2007.

5.1.1 Prototyping Data Entry

As a participant in the INF5750 course, I was assigned to a project group with two other students. Our task was to create a prototype for Data Entry. This involved looking at the Data Entry screen in DHIS 1.4 and trying to make a close a replica as possible. This was challenging, as 1.4 itself was quite new. We did not communicate directly with the 1.4 team, however. This communication was primarily mediated by one of the Norwegian coordinators however. During this period, we participated in discussions on the mailing list and documenting our progress on the wiki. The wiki was also used as a teaching tool in the INF5750. Among the more interesting features was a wiki page where students could ask questions and the teaching assistants could answer them. The page functioned as a sort of FAQ, where everyone could see answers. The questions varied wildly between course and technology oriented courses. Students were also invited to participate in answering questions, and some of those most comfortable with the technologies actually did. In many ways these students represented an actual DHIS 2 developer community. There was much communication going back and forth, probably helped by the fact that some of us could meet face-to-face as well.

The group set up its own code repository, which was later merged in with the main DHIS 2 repository. The reason for this was that, at the time, the repository was reserved for completed core modules in the system. Today the repository has a section where experimental modules can be developed. Since we prototyped this code, the entire module has been rewritten by other developers.

5.1.2 Teaching assistant in the INF5750 course

I also served as a teaching assistant in the second and third iterations of INF5750 the course. This meant I'd hold lab session, correct assignments and guide project groups. What is interesting to note here is that I was also responsible for enforcing development and communication standards, which basically meant I'd make sure the students were coding to the DHIS 2 code style, using the repository, documenting their work on the wiki and communicating their issues on the mailing list. From being someone who had received and internalized these practices, I now became someone who enforced them in others. They usually picked up using tools like Eclipse, Subversion and Maven well, with close instruction. More abstract aspects, such as using spring, inversion of control and other programming principles used in DHIS 2, were more difficult to understand.

During my work I observed a few interesting things. Only one or two of the members of each group would code, while the others would document or do other non-technical work. The students also rarely used the mailing list. Much of their questions were answered by their teaching assistants (who, incidentally, were the most active people on the mailing list) in the lab sessions. Some students even commented that there was too much mail going on the lists, and often mail which they didn't feel was relevant to what they were working on.

Since the first iteration of the course, the coordinators expressed a wish to separate the course activity more from the development activity. This involved moving all the course information and activities away from the HISP wiki. This definitely contributed to people being less integrated in the general development activities.

Students often developed modules in the periphery of the normal development cycle. They would make branches of functionality in the trunk of the code and work on that until they finished the course. The functionality in these branches was later merged into the trunk by regular developers. I, for example, did this for most of the export extensions. As I was the mentor for those projects, I indirectly got the responsibility for these modules later on.

5.1.3 Establishing the DOC space

In late 2006 I took the initiative and participated in cleaning up the wiki and pulling documentation together. I created a separate Documentation space on the wiki, to house all technical and user documentation on DHIS 2 and 1.4. The goal was to make the documentation easier to find and better organized. One of the other Norwegian developers spearheaded the actual work, however, refactoring and rewriting the documentation. It is still lacking, but that's mainly because few developers document functionality actively and extensively on the wiki.

5.1.4 South Africa workshop and conference

During the fall of 2006 HISP organized a conference on District Health Information Software, bringing together both people from the various HISP teams around the world and health officials from different countries. The conference was hosted by HISP South Africa and took place at a resort called Mpekweni, near Port Elizabeth in South Africa. Before the conference, DHIS developers got together for a pre conference workshop in Cape Town. Over a period of one week representatives from South Africa, Tanzania, Ethiopia, India and Norway were present in the same room, working on code, giving presentations to each other and having discussions around the direction of development. I was fortunate enough to accompany two of the core developers and two coordinators from Norway on the journey. We met the development team from Ethiopia, which we had met in Oslo a year before. We also got to meet a coordinator and facilitator from India. This was the first time the developers from Oslo had actually met the Indians face-to-face. We sat down and had some productive discussions around requirements and the needs from the field in India.

After the workshop, most of the participants traveled to Mpekweni for the developer's workshop there and the conference itself. At this point, the two core developers left for home. The workshop aimed to plan the activities of the conference more in detail and serve as a wider technical workshop then the one in Cape Town. Among the interesting discussion was one about information sharing in the HISP network. Participants had showcased several locally built applications. Many of these were surprising to the audience in the sense that they had not heard anything about them prior to attending the workshop, even though they had been in development over the last year. It was highlighted that part of the point of the network was to share experiences, which in this case hadn't happened. The participants asked for ways of making sure information about new projects were spread to the network as a whole. One participant noted that HISP lacked what he referred to as "protocols of sharing". Electronic communication was discussed, but one of the problems was the district and regional levels in the network, where many don't have internet access. One participant suggested a solution for addressing this: A CD with information about what had been happening in the network and demo applications developed elsewhere could be distributed along with the DHIS application itself. An extended discussion about electronic collaboration revealed that one technology that had not been explored was newsfeeds: These are snippets of news which are "pushed" to the client's web browser. Instead of the user having to go out and look for information and stay updated, news is delivered to the user which can then click a link for more information.

The conference itself was targeted at facilitators, health professionals and people working with health information systems in different countries. There were several plenary sessions involving panel debates and informative lectures from the different countries involved in HISP. The DHIS 1.4 and 2.0 applications were demonstrated, along with applications developed in Zanzibar and Ethiopia. Additionally, there were workshops on more specific topics such as help manuals, data set definitions. I hosted one such workshop on electronic collaboration tools, showcasing the hisp.info wiki especially.

5.1.5 Re-implementing the export module

During the fall of 2006, I was assigned to updating the Import-Export module along with one other developer. We split the task between us, and I focused on Export. We ended up replacing the entire module, as the old code was too hard coded and tightly coupled. I introduced a new set of concepts and interfaces to handle extensions to the export functionality. This work started while I was in South Africa and at the same time my groups in the INF5750 course were scheduled to implement such extensions. I would check code in through Subversion and explain the code on the dev-list. Each group would develop one type of export each, for example to the statistics program SPSS and to some nation-specific software, using my interfaces.

I had to coordinate closely with my colleague, which worked in import functionality. This was a Norwegian developer, stationed in Vietnam at the time. We communicated primarily through mail on the mailing list, as we both had problems we needed help from others to solve. In addition to that, we talked over MSN. The five hour time difference between Norway and Vietnam did not make itself so felt, but mostly because my colleague worked late and often into the night.

5.1.6 My visit to India

I traveled to India during the summer of 2007. I spent two weeks in the central office of HISP India, just outside Delhi. While I was there I worked closely with a coordinator, two facilitators and one central developer in the organization. I was mainly sent to help with capacity building, especially working with the developer. The aim was to get him started on working directly on the core of the system, as well as solving some of the problems India faced while I was there. Both the coordinator and the developer also requested that I work on a few things or give some introductions to various technologies.

It quickly became apparent that their problems were related to a part of the codebase I knew little about: The datamart module. This module will aggregate data in the system to be used in reports and pivot tables. The module ran very slow, taking up to a day to complete calculations for a common set of parameters. It would also run out of memory and fail on the Indian, low-end machines. Furthermore, it would become clear that for some reason, the aggregation result was wrong or lacking. They then set me to work actually running aggregation for them, as my laptop was faster and had more memory. The data my computer produced would then be used in presentations and the like. The erroneous results proved very problematic however. They sparked heated mails between people higher up in the network.

As I couldn't solve the problems myself, I first tried to get an overview of how the aggregation and datamart functionality worked. This involved discussing with the local team about how it's supposed to work, and with the person in Oslo who was responsible for the code. In many ways I took up the mediator role, as described by some of my informants. I would discuss problems locally and get an understanding for them, then try to explain them to my Norwegian colleagues. This was probably good in the sense that I could formulate mails in English better than my local colleagues. I tried to note down the feedback they were giving, describe the problems they were having, note system specifications, and so on. I then tried to put as much of that information onto the mailing list as possible, to keep people informed and possibly spark some discussion. This was only moderately successful, as there were few active people on the Norwegian side at the time. But we did manage to demonstrate and clear up some different views on how the aggregation algorithm was supposed to work. The Indian team was used to 1.3 and 1.4, where aggregation is done for each organization unit in the system. The Oslo team had implemented a different algorithm, however, which also attempted to take into account if organization units had been moved during the aggregation period. This resulted in an algorithm which did calculations differently.

Furthermore, I would physically sit close to the others while they worked. The developer would ask me questions directly and we had a chance to discuss solutions to the problems. I fear the Indian team may have expected a "guru", but there were many questions I just couldn't answer. In such cases I tried to help by doing web searches. In many cases I had to rely on my colleagues from Oslo for answers, especially the core developers. This required direct communication, which was made difficult by the 3 1/2 hours time difference.

While I was there, I also had a chance to do a walkthrough of the system. One of the facilitators went through each screen and commented on problems and what was needed. We had a chance to talk about problems with the user interface face-to-face. This resulted in a myriad of tickets into Trac. While I was there I reported over thirty tickets in the system, making up roughly one fourth of the current tickets in the system at the time.

I tried to be a bit manipulative as well. After certain discussions, or when they made suggestions, I would try to nudge them in the direction of the ticket system, or encourage them to write a mail to the list. My motivation for this was to make sure that the team wouldn’t come to rely on me for communicating issues to the Oslo team. I couldn’t always be around to relay information or answer questions. I also thought that if I was there and helped out when they registered tickets or sent mails, their general comfort level would increase as a result.

During my visit I felt I grew closer to them on a personal level. I'd met a few of them already, but only for a short time period. We shared meals, went out together, went on trips and talked about other things than coding. After the visit, the effect of meeting face-to-face became clear immediately. The Indian developer would now contact me directly through GTalk and ask me technical questions he had. This is the same effect as has been reported by the Norwegians which have gone to Vietnam. After their stay, they would often continue to communicate with each other on IMs, and there seems to be a lower barrier to ask questions directly. It has not, however, seemingly increased any participation on the mailing list or in any of the other tools. From a personal standpoint, this hasn't been all positive. I have enjoyed conversations with an Indian developer, but questions often came at a time when I was busy with other things, for example during work or while writing my thesis. I wanted to help, but felt I had to rebuke the questions due to time constraints. That gave me a guilty conscience: When I finally had a good dialogue with a developer from another part of the network, I found I couldn't spend time assisting him.

At the same time, one of the informants' point about how problems are not solved via mail was confirmed. While I had written mails and registered tickets, it wasn't until I got back to Oslo that I felt what I'd reported was understood properly. We needed a dialog to clear things up.

5.1.7 Implementing Calculated Data Elements

During the summer of 2007 I was charged with implementing Calculated Data Elements, as a result of a heated mail discussion between the India and Oslo teams. The assignment seemed to be a sort of fire extinguishing effort from Oslo's, supplying some critical functionality quickly. The functionality had been requested by the Indian early the same year, but it was never picked up as critical functionality in any later priority discussions. The requirement had been introduced in a mail on the mailing list, but had since fallen through the cracks. Interestingly, it had not been raised as a priority by the Indian representatives until the discussion broke out.

I began the development while being in India, talking directly to people there for requirements. I finished the implementation in Norway, using IMs to ask for feedback and clarifications. I mainly talked to one Indian coordinator and one facilitator, as well as to a Norwegian coordinator in Oslo. An interesting thing to note here are some conflicting requirements. Or more correctly, the Indians requested less functionality then was currently present in DHIS 1.4.

5.2 Quantitative analysis

I wanted to get an understanding for how the mailing lists and repository are being used and who are participating. The mails are stored in a list archive, available online. The repository keeps a record of all the revisions, i.e. commits performed, in a log. This gives an easy access to a lot of data. I tried to formulate a few questions to guide how to structure and analyze the data.

1. How many commits have been made to the SVN repository?
2. How many have committed to the repository?
3. How many mails have been sent to the mailing list?
4. How many have participated on the mailing list?
5. How many mails and commits can be attributed to specific participants?
6. How many people are participating in discussions?
7. How many people initiate and how many people respond to discussions?
8. How do these numbers fan out across dimensions such as role, country and time?

I go into detail about the choice of roles in a later section. I choose these dimensions to get some more data on who the different participants are. It's interesting to know, for example, whether or not students account for much or most of the communication, whether or not other than the core developers commit code and how much the coordinators get involved in the development. The time dimension allows us to track changes in the pattern of participation.

5.2.1 Country and role

I quickly repeat the various roles that I defined in the Data Analysis section:

- Core developer: The four developers with the assigned title.
- Developer: Other active developers on the project.
- Facilitator: A person working in the field, implementing the system into organizations.
- Coordinator: A leader of a node or activity in HISP.
- Student: Students who participate in the project in a limited time span.
- External: Other people who contact the mailing list.

5.2.2 Commit log analysis

I reviewed the logs from the Subversion repository, using data from the first commit in January 2005 to early January 2007. In that period there had been 2647 commits (Question 1). Quick analysis shows that there had been 104 distinct committers in that period (Question 2). The following results are based on averages and member counts for only those who have committed to the repository, i.e. not those who have only sent mails.

If we break down the commits down by country, we get the following results:

Commits	Country	Members	Average
2405	Norway	90	26.7
199	Vietnam	7	28.4
35	India	5	7.0
5	Ethiopia	1	5.0
3	Tanzania	1	3.0

In this table we see that Norwegians represent the vast majority of commits, almost 91% of the total. However, as we can also see, there are a lot more Norwegians than others, 86% of the total number of committers. In this data, Norwegians and the Vietnamese contributors have almost the same average number of commits.

If we break the data down by country and role, the results become a bit more nuanced:

Commits	Country	Role	Members	Average
1154	Norway	Core developer	4	288.5
861	Norway	Student	78	11.0
358	Norway	Developer	6	59.7
198	Vietnam	Developer	6	33.0
32	Norway	Coordinator	2	16.0
22	India	Developer	4	5.5
13	India	Coordinator	1	13.0
5	Ethiopia	Developer	1	5.0
3	Tanzania	Facilitator	1	3.0
1	Vietnam	External	1	1.0

The core developers are in the lead with 1154 commits, which accounts for roughly 43% of the total number of commits. Then follow the Norwegian students which have 861 commits, another 32%. Norwegian developers have 358 which is 13%. The Vietnamese team accounts for 198 commits, around 7% of the total. What we also see here is that of the 90 Norwegians, 78 are students which have a temporary relationship to the project. At this point each student does on average 11 commits, while the core developers each do 288 commits on average. We see that the work on the project is clustered around the core developers.

We also see that some coordinators from Norway and India have contributed to the project, in addition to four Indian developers.

If we break the results down by role we see few changes. The Norwegian and Vietnamese developers are summed, but still not reaching the level of the students. The students are all Norwegians, so the result for this group stays the same.

Commits	Role	Members	Average
1154	Core developer	4	288.5
861	Student	78	11.0
583	Developer	17	34.3
45	Coordinator	3	15.0
3	Facilitator	1	3.0
1	External	1	1.0

If we move down to the individual level, we see the following results:

Commits	Person
598	Norway Core developer 1
275	Norway Core developer 2
259	Norway Core developer 3
116	Norway Developer 1
87	Vietnam Developer 1
71	Norway Developer 2
65	Norway Developer 3
57	Norway Developer 4
48	Norway Student
46	Norway Student
43	Norway Developer 5
37	Vietnam Developer 2
35	Vietnam Developer 3
33	Norway Student
31	Norway Coordinator 1
30	Norway Student
27	Norway Student
27	Norway Student
26	Vietnam Developer 4
26	Norway Student
25	Norway Student

In the figure above the results are cut on 25 commits. The rest of the data show a steady decline from 24 and down to 1. The majority of the people below the list are Norwegian students.

As seen, the core developers are in the lead, with the most active developer on a solid first place with 598 commits, a full 22% of the total. Then it drops to a Norwegian developer, which is me, and from then on to include most of the Norwegian and Vietnamese developers. Many students rank high here and all of them have a higher number than the average for their group. In the full data one can see that many students have one or a few commits. Students are organized in groups, and from tutoring them in the INF5750 course it seems that one or two from each group does most of the coding.

The five leading committers have contributed 50% of the commits done to the project. In FLOSS development there is often an 80–20 rule, stating that 20% of the developers have contributed 80% of the code (Weber 2004:71), a variant of the Pareto Principle (Wikipedia 2007). With a 104 committers, that would mean that the top 21 committers should represent this. Incidentally, the list above is cut at exactly 21 committers, the last having 25. There are 2647 commits, so 80% would mean roughly 2118 commits. The sum of the commits above is 1962, which accounts for 74% of the total number of commits. We could say that we're close to the 80–20 rule, but the number of commits decreases steadily down the list.

So, what can we say based on these results? It is quite clear that the majority of commits are done by Norwegians. The Core developers again represent most of the commits. Furthermore, one particular core developer as the most active contributor.

5.2.3 Mailing list analysis

I reviewed archives of the mailing list between September 2005 and January 2006. In that period 3574 mails has been sent (Question 3). Quick analysis shows there were 110 distinct mailers in that period (Question 4).

If we break down the mails by country, we get the following results:

Mails	Country	Members	Average
2821	Norway	65	43.4
340	Vietnam	15	22.7
295	India	12	24.6
78	South Africa	3	26.0
19	Ethiopia	5	3.8
15	None	3	5.0
3	Unknown	3	1.0
2	Spam	2	1.0
1	Tanzania	1	1.0

Again the Norwegians dominate the activity with 2821 mails, roughly 79% of the total. Vietnam follows with 340, which is about 9,5%. India has 295, around 8% of the total.

And broken down by country and role, the results become thus:

Mails	Country	Role	Members	Average
1213	Norway	Core developer	4	303.3
838	Norway	Developer	7	119.7
565	Norway	Coordinator	2	282.5
268	Vietnam	Developer	6	44.7
191	Norway	Student	51	3.7
160	India	Developer	7	22.9
121	India	Coordinator	1	121.0
77	South Africa	Coordinator	2	38.5
50	Vietnam	Student	6	8.3
19	Vietnam	External	2	9.5
15	None	None	3	5.0
14	Norway	Facilitator	1	14.0
13	Ethiopia	Developer	2	6.5
12	India	Facilitator	3	4.0
3	Ethiopia	Coordinator	1	3.0
3	Unknown	Unknown	3	1.0
3	Vietnam	Unknown	1	3.0
2	Ethiopia	Facilitator	1	2.0
2	India	Unknown	1	2.0
2	Spam	Spam	2	1.0
1	Ethiopia	Unknown	1	1.0
1	South Africa	Unknown	1	1.0
1	Tanzania	Coordinator	1	1.0

In these results the core developers have 1213 of the mails which amounts to 34% of the total. Other Norwegian developers have 838 mails, 23.5%. Then follow the Norwegian coordinators, with

565 or 16% of the total. Vietnamese developers contribute 268 mails or 7% of the total. Norwegian students, Indian Developers and Indian coordinators have 191, 160 and 121, which amounts to 5%, 4.5% and 3% respectively.

First of all, contributions to the mailing list are more diverse than contributions to the repository. This is to be expected. Not all participants in DHIS 2 developments are coders. At the same time, the core developers and other Norwegian developers still dominate. The coordinators are more active here. Interestingly student participation drops considerably when compared with how active they are in coding. This may be because they are more interested in completing their project task more than communicate with the rest of the developers. They also receive information and guidance directly from their teaching assistants in the INF5750 course.

Broken down by role:

Mails	**Role**	**Members**	**Average**
1279	Developer	22	58.1
1213	Core developer	4	303.3
767	Coordinator	7	109.6
241	Student	57	4.2
28	Facilitator	5	5.6
19	External	2	9.5
15	None	3	5.0
10	Unknown	7	1.4
2	Spam	2	1.0

What's interesting to note here is that while Norwegian developers have produced the highest number of mails, they are still effectively beaten by the core developers, which while being fewer in number produce more mails. The coordinators appear on a solid third place, with a higher average than the developers.

The mails break down by person in the following way:

Mails	**Person**
772	Norway Core developer 1
563	Norway Developer 1
396	Norway Coordinator 2
322	Norway Core developer 3
169	Norway Coordinator 1
121	India Coordinator 1
118	Norway Developer 5
104	Norway Core developer 2
89	Vietnam Developer 6
78	Norway Developer 2
76	South Africa Coordinator 1
75	India Developer 1
73	Vietnam Developer 1
45	Vietnam Developer 3
41	India Developer
34	Norway Developer 4
34	India Developer
28	Vietnam Developer 2
28	Norway Developer 3
26	Norway Student
25	Vietnam Developer 4

In the figure above, the results are cut on 25 mails. Once again the number of mails drops steadily down the list.

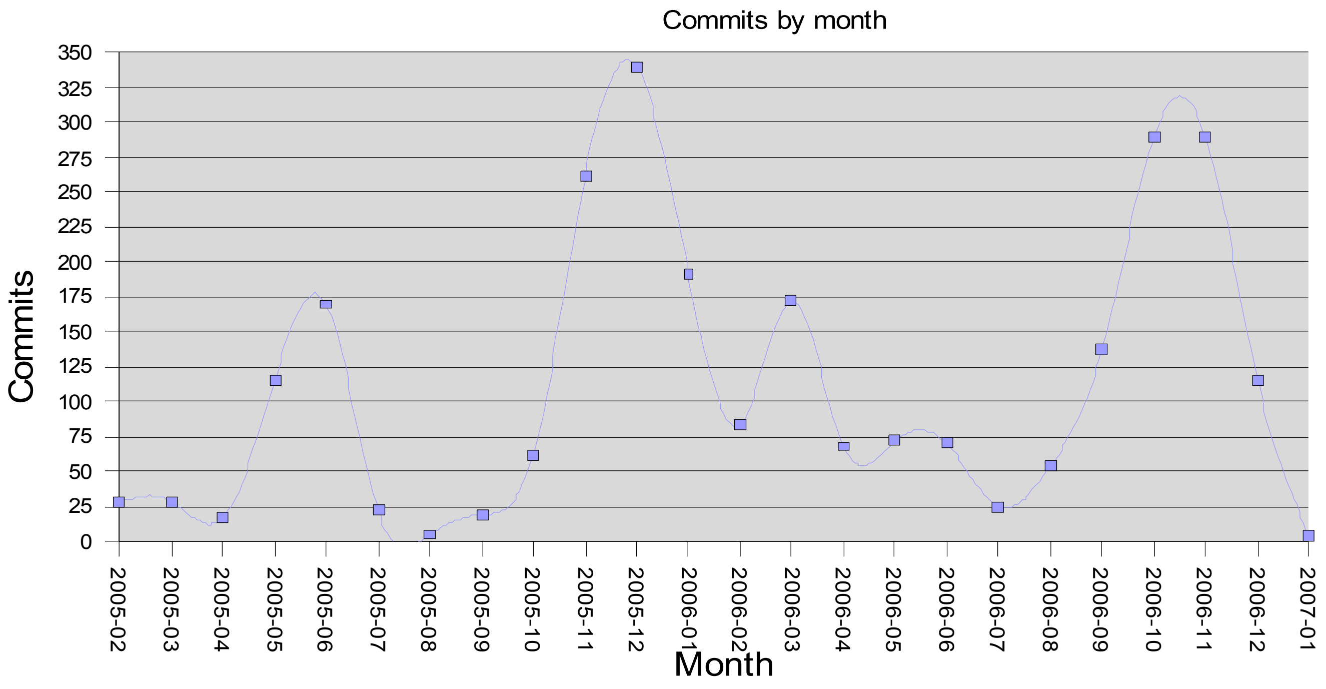
One particular core developer is once again in the definitive lead with 772 mails, a full 21.5% of the total. I come in second place with 563 (I pester the list quite a lot, you see…) which is roughly 16% of the total. There is only one Norwegian student among the most active, which shows that the bulk of the communication is done by those internal to the project. An Indian coordinator ranks quite high. He is the primarily contact person between the Oslo and India teams.

5.2.4 Changes over time

5.2.4.1 Repository activity

It is interesting to see if the participation has changed over time, if the relative participation of the different nodes changes over time or during seasons, plus what has happened to the development in general. One could hypothesize that the activity of nodes outside Norway have gone up over time, following capacity building in for example Vietnam and other efforts to spread the development activity.

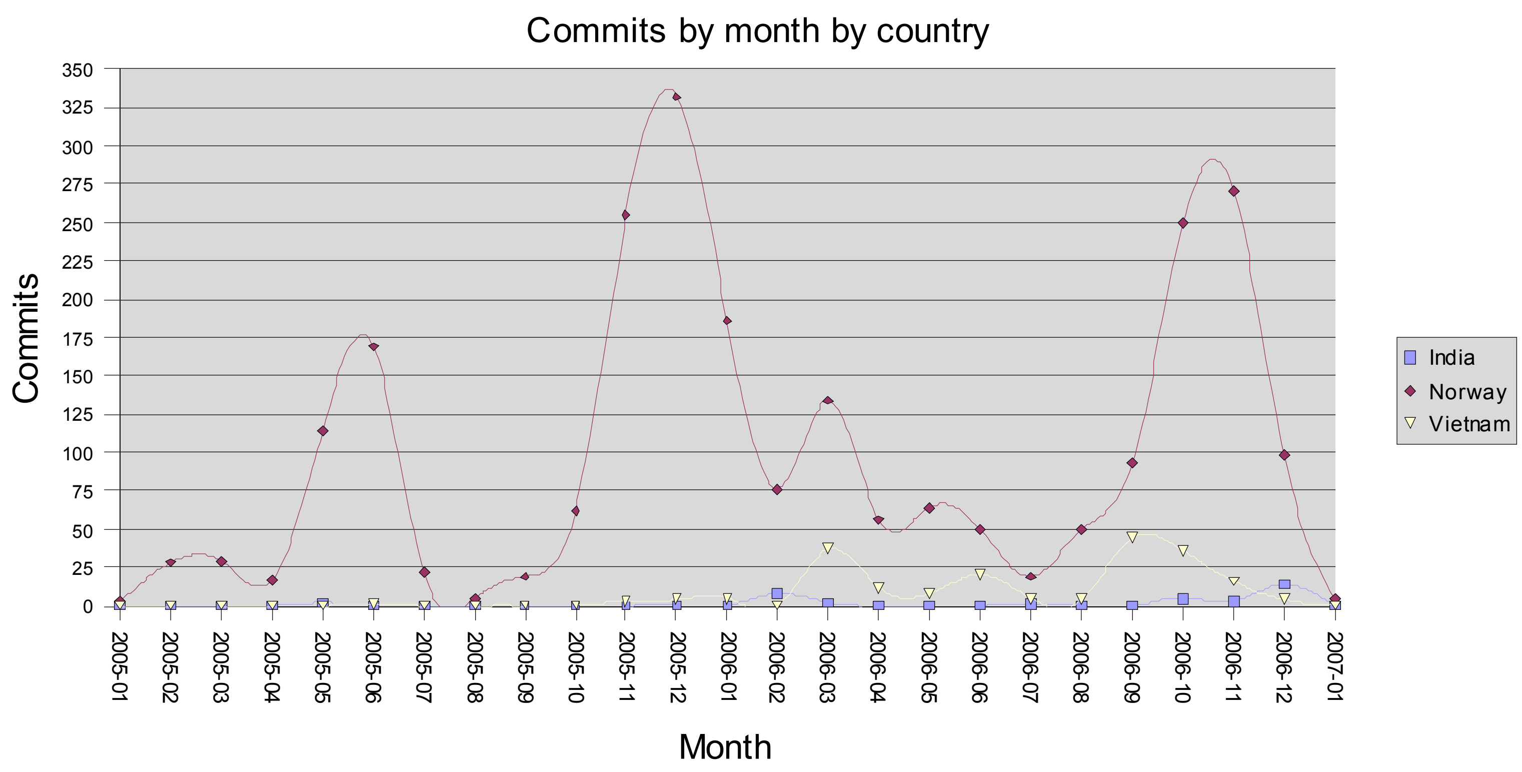
The following graph shows the number of commits by month in the data period.



5.1 Commits by month

The peaks occur in the periods May to June 2005, November 2005 to January 2006 and October to November 2006. There are some low periods in July to September 2005 and June to August 2006. These are slightly odd fluctuations, not really showing so much. But by drilling down we might get some more information.

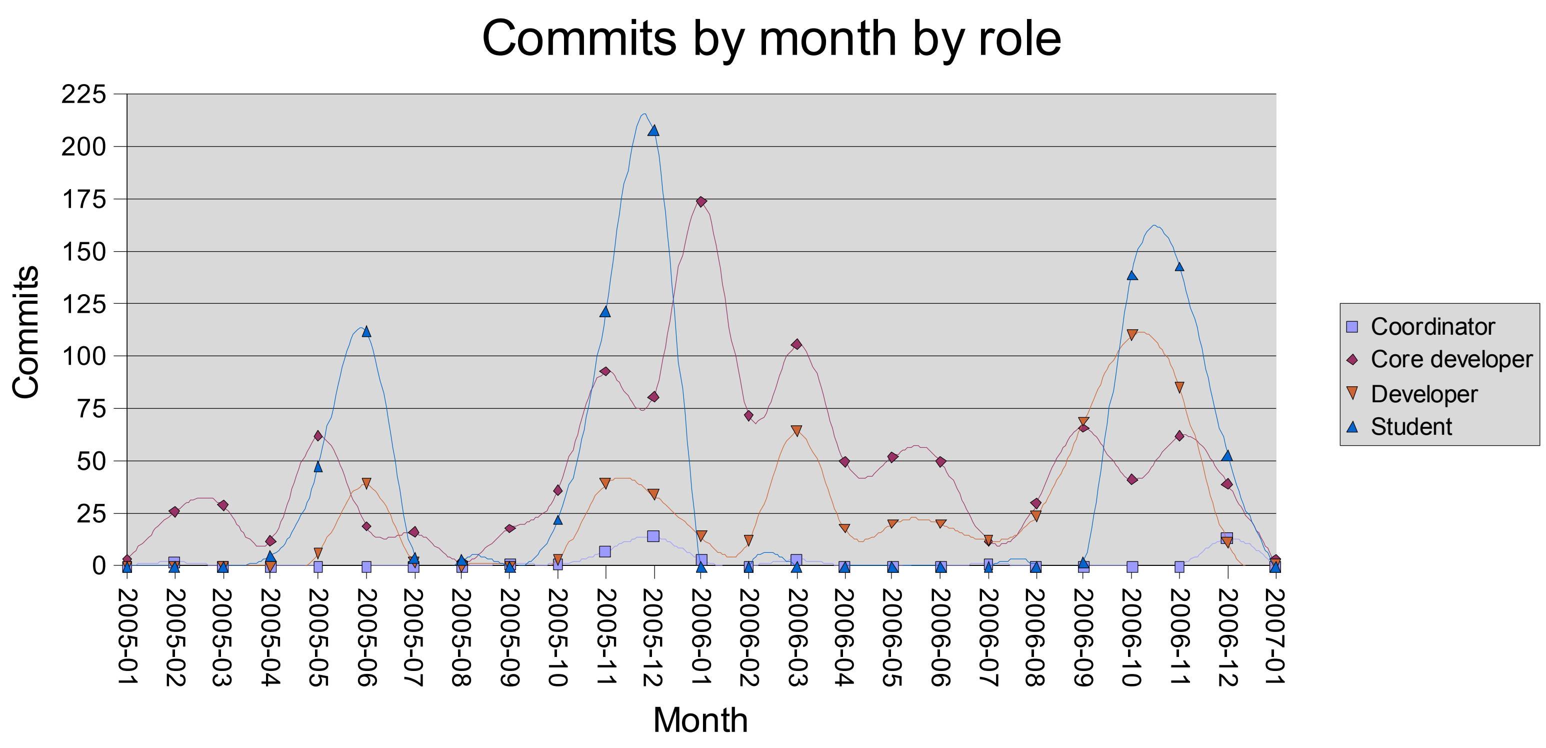
If we break this down by country, we get the following results



5.2 Commits by month by country

In the graph, I've cut Ethiopian activity, as these are too marginal to be interesting, as well as unidentified and administrative activity. From this graph the dominance of the Norwegians is visualized quite clearly. The Vietnamese activity starts around November 2005 and from there on out it seems to match the activity of the Norwegians, but being proportionally lower. It's thus difficult to show any sort of trend with regard to their participation. Indian participation in the period is marginal.

But we know that the Norwegian students represent a considerable part of the activity. By breaking the data down by roles, the pictures changes a little:



5.3 Commits by month by role

The student contribution naturally follows the INF5750 course. Since 2005 it has run in three iterations: The spring of 2005, the fall of 2005 and the fall of 2006. This is reflected in the data where student contribution spikes before final delivery in the course, in May to June 2005, November to December 2006 and October to December 2006. The activity disappears once the delivery is completed. An anomaly here is the first iteration of the course where student contribution seems to continue into the summer. This is possibly due to the fact that the project was new and exiting to all the participants, driving them to participate more. The lowdowns shown in the previous graphs are probably explained by the fact that most Norwegians go on holiday during the summer, often concentrated around July.

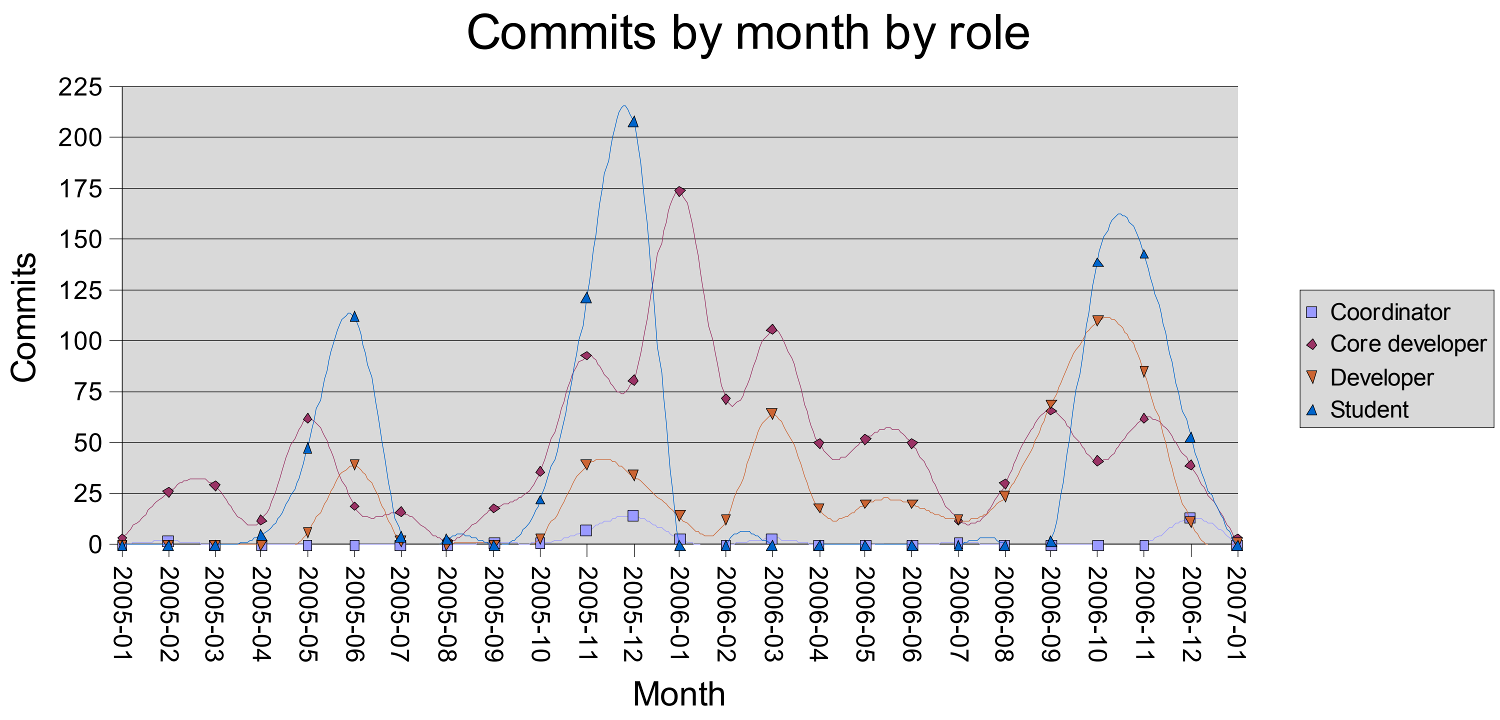
5.2.4.2 Commits compared to milestones

It's interesting to compare this with the release date of the various milestones of DHIS 2, which are summarized in table x.1. Likewise, one could assume that activity generally goes up just before these releases due to more efforts to prepare the code for release.

Milestone	Date
M1	2006-02-15
M2	2006-03-17
M3	2006-04-17
M4	2006-05-15
M5	2006-06-15
M6	2006-12-08

It should be noted that the system was under initial development for a year before any formal release was made. Thus, M1 is released in February of 2006. The initial idea was to release milestones on a monthly basis, using short development cycles, incrementally adding functionality with each release as is common in an Agile development process. This schedule was broken when the M6 release was made as late as six months after M5.

If we look at the results by month again:



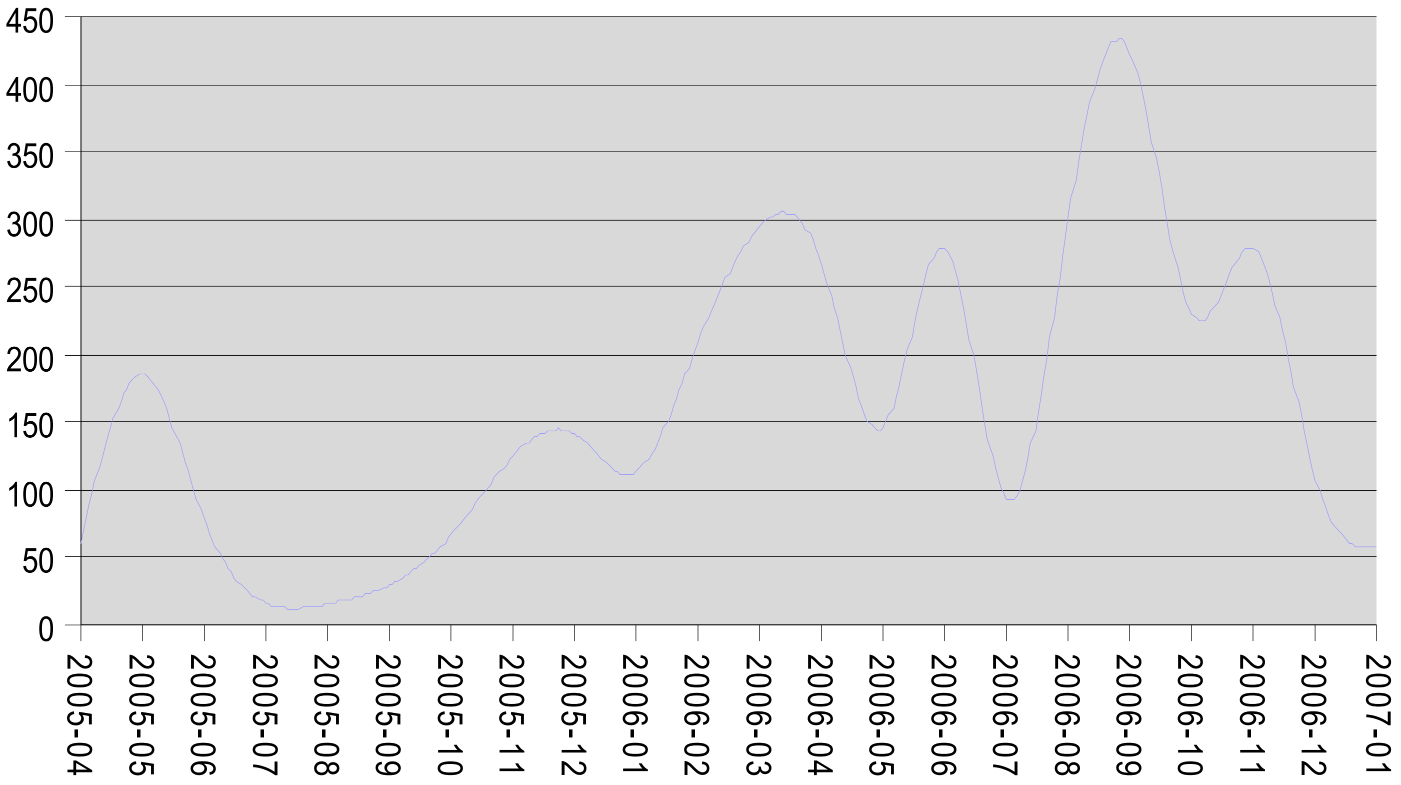
5.4 Commits by month by role

The core developer activity spikes in January 2006, a short time before the release of M1 in February and similarly again in March, before the M2 and M3 releases. The following table presents the commits by month and the corresponding milestone releases.

Month	Commits	Releases
2005-01	3	
2005-02	28	
2005-03	29	
2005-04	17	
2005-05	116	
2005-06	170	
2005-07	22	
2005-08	5	
2005-09	19	
2005-10	62	
2005-11	262	
2005-12	340	
2006-01	191	
2006-02	84	M1
2006-03	173	M2
2006-04	68	M3
2006-05	72	M4
2006-06	70	M5
2006-07	25	
2006-08	54	
2006-09	137	
2006-10	290	
2006-11	290	
2006-12	116	M6
2007-01	4	

5.2.4.3 Mail activity

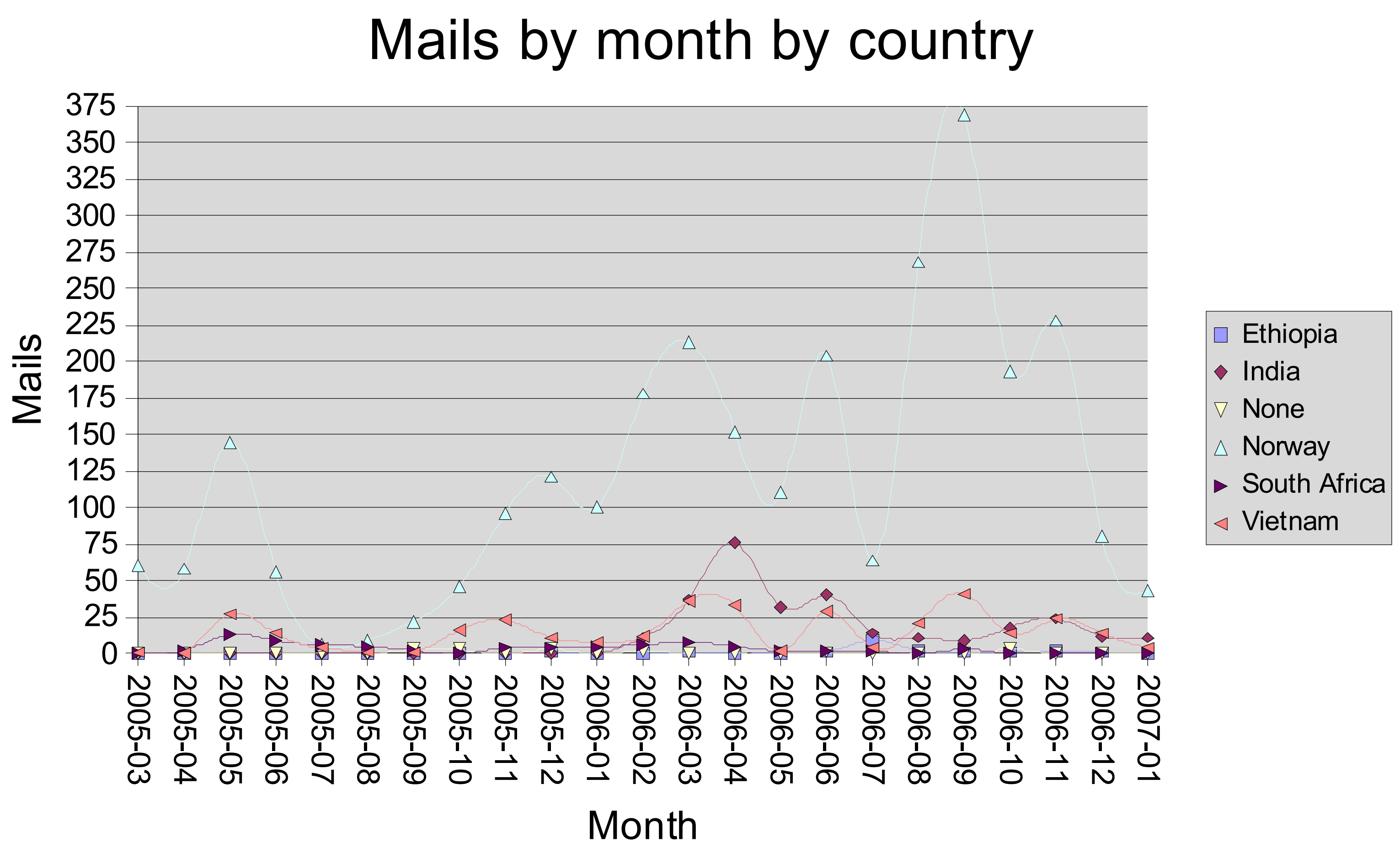
Similarly to repository activity, it's interesting to see if the pattern of participation changes over time. First, we look at the total number of mails per month in the data period.



5.5 Mails by month

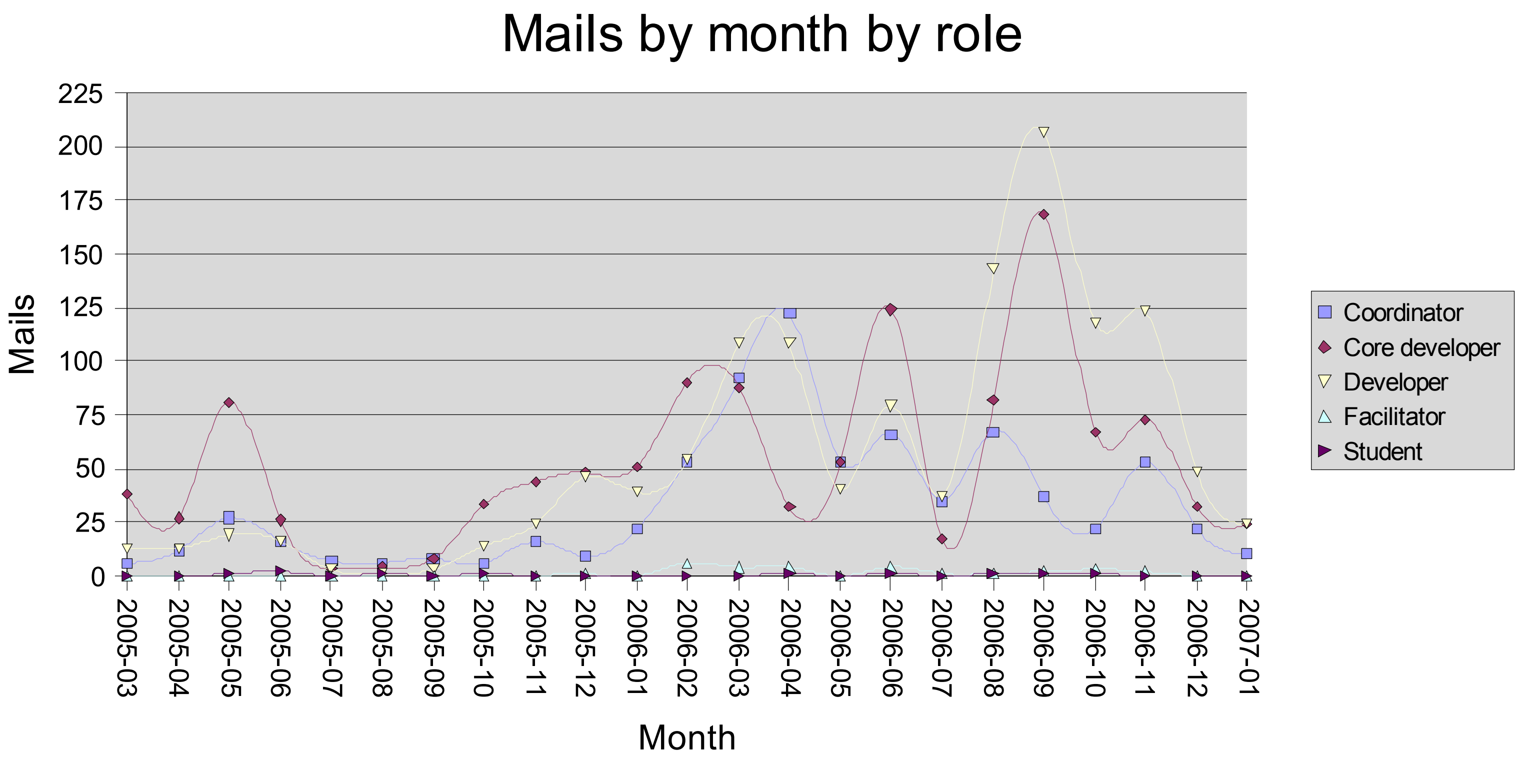
The peaks occur in May 2005, February to April 2006, June 2006, August to September 2006 and November 2006. The low points are in June to September 2005, May 2006 and July 2006. The data show a week increasing trend with regard to number of mails over time. The dip in January 2007 is because of when the material was cut.

Broken down by country, we get the following results. Mails from Tanzania, because of negligible number of mails, and Unknown or Spam mails are removed.



5.6 Mails by month by country

With this result we see that the Norwegians consistently dominate the activity throughout the period. Most of the variations in the previous result are mostly caused by the Norwegians. The Norwegian mail activity reaches an all-time high during August to September of 2006. Interestingly the Indian mail activity spikes in April 2006.



5.7 Mails by month by role

I eliminated the None, Unknown and External roles here. Neither of them represent significant data. From these results we see that in the peak period it's actually developers who represent the largest amount of activity in the highest peak period. We known from the previous results that those in this peak are mostly Norwegian, although part of it is the work of Vietnamese developers.

5.2.5 Problems with the data

It should be noted that the commit data and the mail data start on different dates. The first mailing list archives were lost when the server they were residing on was moved. The data that are missing pertain to the startup phase of DHIS 2 development, driven by the first run of the INF5750 course. I expect that these data would show an increase of participation by the core developers, a few of the current Norwegian developers, and for some of the INF5750 students. Additionally, Norwegian Coordinator 2 was very active in this period, as he was also the course administrator. At the time, there was little activity in India and Vietnam. However, the South African team might have been more active at the time, advising the startup of the new project.

5.2.6 Mailing list content analysis

So what are people talking about on this list? How much is technical discussion, support and project administration? Is there an element of social activity on this list? I used a method of reading and tagging a random selection of the mails of the list, as described in Chapter 3.

I read through a total of 527 mails, roughly 15% of the number of mails sent.
When counting the total number of mails in each category, the results look like this:

Mails	Tag
141	Tech
133	Support
79	Project
58	Req

47	Broadcast
20	Norms
18	Bug
10	Admin
10	Praise
6	SVN
3	License
1	Issue
1	Social

As we can see, support discussions and technical discussions make up most of the activity on the list.

Of these, SVN, Admin, Social, Bug and Issue where only fractionally present. The Social tag got only one clear hit, although several more mails do include a hint of humor.

5.2.7 Thread analysis

In this section I want to see how discussions take place, especially who initiate and who respond to them. Based on the results above, that certain developers, and indeed Norwegians in general are the most active on the code, one could perceive different hypotheses: One would be that people outside the Norwegian team ask questions and the Norwegian developers answer them, giving the rest of the developers a higher amount of initiation. However, this needs to take into account the kind of traffic on the list. When support ranks high in the content analysis, that would seem to support such a hypothesis. Then again, technical questions rank highest, and as the Norwegian developers are most active on the coding, one could expect to find that Norwegians actually rank highest on the initiation and respondent side of the discussions, being most active in technical discussions.

There were a total of 993 threads in the period, with an average length of 3,6 mails. The longest threads had 28 mails. Of these threads, 354 had a length of one, meaning there were no replies. A further 199 had a length of two, meaning one reply. The rest of the thread length statistics are summarized in the Figure x.x below.

Threads	**Length**
354	1
199	2
115	3
75	4
57	5
52	6
35	7
20	8
16	9
15	10
10	11
9	13
9	15
6	12
4	16
3	17
2	14
2	18
2	19
2	28
1	26
1	21
1	25

1	27
1	24
1	23

Not surprisingly, the number of threads goes down as the thread length go up. The median is somewhere inside mails with only one reply. What is somewhat surprising is the large amount of threads with no replies. These are either meant as broadcast messages or questions that never receive an answer. This may indicate a weakness in the content analysis: There the technical discussions and support requests seemed to dominate, but this show that the number of broadcast messages might be a lot higher. By not taking into account thread context when analyzing the content of one mail, I miss mails that effectively are broadcast messages.

If we break the initiations down by country, we get the following results:

Country	**Initiations**
Norway	686
Vietnam	133
India	125
South Africa	27
Ethiopia	11
Tanzania	1

As we can see, the Norwegian participants dominate thread initiation as well with 686 or roughly 70% of the total. The second and third place is held by Vietnamese (with 133 or 13%) and Indian (with 125 or 12,5%) participants respectively, amassing a total of 228, or around 26%.
Mails with “None”, “Unknown” or “Spam” are eliminated from this list.

5.3 Qualitative analysis

In Chapter 3 I gave a general introduction to various collaborative tools, and introduced quickly the tools being used in DHIS 2 development. In this section I go more into detail on how the tools are used, based on interviews with participants and observations from my participation in the project.

5.3.1 The developer mailing list: dev@dhis.hisp.info

One of the most central tools for collaboration in DHIS 2 development is the developer mailing list, dev@dhis.hisp.info. All developers I interviewed reported using this channel for communication. It is indeed the most active of all the tools in the project. It sees regular daily activity.

As is the case with DHIS 2 development in general, the members of the Oslo team are the most active on the dev-list. As we have seen, the list primarily sees two kinds of traffic: One is technical discussions and questions among the developers and the other is support requests from administrators and developers, bug reports and feature requests. Some typical problems are called build errors, often triggered by one of the developers checking in a piece of code which breaks the build process for another developer.

Some participants have reported that they find the dev-list too technical. The list sees several code specific discussions. An example:

```
When working on the indicator csv export GUI I get this exception the
second time I do an export (the first export works):

80209 [SocketListener0-1] WARN  org.mortbay.jetty.servlet.ServletHandler
- Error for /app/dhis-web-
importexport/advancedIndicatorExport.action?filename=
foo2.zip&includeParent=false&periodSelection=all&fromDate=&toDate=&exporte
```

```
rFormat=indicatorcsv
java.lang.NoSuchMethodError:
org.hisp.dhis.service.indicatorstore.IndicatorType.getHibernateLazyInitial
izer ()Lorg/hibernate/proxy/LazyInitializer;
        at
org.hisp.dhis.service.indicatorstore.IndicatorType$$EnhancerByCGLIB$$44665
064.getHibernateLazyInitializer(<generated>)
        at org.hibernate.type.EntityType.resolveIdentifier
(EntityType.java:274)
        at org.hibernate.type.EntityType.resolve(EntityType.java:303)
        at
org.hibernate.engine.TwoPhaseLoad.initializeEntity(TwoPhaseLoad.java:116)
(...)

People seem to fix this issue by either going back to version 3.0,
patching 3.1 or upgrading to 3.2 GA..  Can we do this upgrade? Our current
version is 3.1.3.

I have commited the code, but as long as it dosent work the menu will be
linking to the old version so to try this point the browser to dhis-web-
importexport/exportIndicatorGui.action.

The OS/JVM used seem to have an effect on this, I´m running this on OS X
(Darwin Kernel Version 8.8.1) using JDK 1.5.0_06-112.
```

One particular comment relates to posting Java errors, called Exceptions, on the list. This is a long trace of several errors following the path through the code that cause them. For a few, less Java savvy informants, I think this is a source of confusion and a degree of fear.

The list probably sees a large group of lurkers, although this is somewhat difficult to track. Some members have reported that they do read the traffic on the list, although that they usually focus on what they find interesting. They usually don't respond, as they don't have anything to add to the discussion. One Indian developer formulated it this way:

> *"If there is anything related to me, I read it."*
> - Indian developer

If the subject is something he is interested about, for example a module which affects his own code or is especially relevant to the local team, he reads it. This feedback is repeated by other participants as well. They do read mails on the list, but only those they find relevant. The developer reports that he doesn't feel comfortable asking questions in discussions on the list. At the same time he reports that he wants to send mail to the mailing list to get feedback about a module he developed, which seems to indicate using the list is something one warms up to.

But there are some problems with the content on the mailing list, from this simpler point of the length of the communication:

> *"If the mail is too long, I skip it."*
> - Indian developer

An Indian facilitator echoes this and extends it by saying that because of the sheer volume of communication on the list and the fact that it's too technical he stopped reading this altogether. An even more serious problem is reported by Nguyen (2007:114):

> *"[T]he [Ethiopian] team did not find it useful for their needs, as took so much time to get the information they wanted. For example, the reply of the team in Oslo on the mailing list often was not specific at all, and instead of giving precise directions, they suggested a link to*

> *an online article or requested to review a previous email. That was why after trying it several times, the team stopped using it."* (my changes)

It's difficult to properly analyze the problem without knowing the context. This is possibly related to different expectations from the two teams: The Ethiopian team is under a lot more pressure from local requirements and situations, and needs quick, direct solutions to their problems. The Oslo team responds in an "educational" way, pointing to resources the team can use to learn, and thus indirectly solve their problems. This is difficult if the local team does not have time to read books to solve immediate problems. At the same time, as noted by Nguyen, the Oslo team does refer to older mails which can be read in the archives so that the team doesn't have to repeat itself. But, we have seen from the thread analysis that many threads simply go unanswered. This would also mean that many questions are not answered which would contribute a lot to the problem.

People generally seem to be comfortable with the tone on the mailing list. However, one of the developers stated:

> *"Sometimes I feels some sentences are irascible, and they make the others not happy, this is bad for working together."*
> - Vietnamese developer, by mail

This point to the fact that mailing list communication is not without its problems. It may be easier to write more free of restraint, resulting in statements that may be too harsh. On the other hand, the aural and visual aspects of the communication is lost, which may make it difficult to interpret the intention behind the words.

Early on the administrators of hisp.info attempted to set up a users@dhis.hisp.info, a place where users could contribute with feature requests and some bug reports. It was also envisioned as a list similar to the DHIS list in South Africa, which has large amounts of traffic, also directly health related discussions. When the users-list was created, a mail was sent out asking people to use the users-list for non-technical matters. The list has seen basically no activity since inception. One of the explanations for this is that there is no large user community. As mentioned, most of the traffic in India goes to an internal mailing list. Furthermore, few actions have been taken to remind people of the list, and enforce that non-technical discussions should go to that list. One coordinator was slightly surprised when I mentioned the user list and exclaimed that he'd forgotten all about it.

In addition to the mailing lists themselves, there exists a set of guidelines (HISP.info 2006) for how the mailing list should be used. Examples include writing in English, writing below the text you are responding to and avoiding SMS speak like "thnx" and "plz". Some of these rules, like writing in English, are common sense rules: The developers are from many different countries. Other rules are based on longer standing traditions on the internet. The writing below that you reply to is one of them. This is a practice which has existed on newsgroups and mailing lists since their early inception and is considered part of netiquette (http://en.wikipedia.org/wiki/Netiquette). This is a tradition unknown to many non-technical participants on the list. Technical participants have also been known to reply above and then quote an entire mail, probably because they have limited time and it is faster to do so. In the mailing list guidelines, these rules are said to apply to all mailing lists on hisp.info, including the users mailing list and the more general mailing list. However, the rules have not been enforced on these lists. On the dev list, there have been occasional comments, but nothing really serious.

One interesting aspect of coordination on the mailing list is the concept of lazy consensus, defined by the Apache foundation as *'A decision-making policy which assumes general consent if no responses are posted within a defined period. For example, "I'm going to commit this by lazy* *consensus if no-one objects within the next three days."'* (Apache 2006). This occurs frequently when a developer is a bit unsure of whether or not his or her changes are acceptable, but generally believe them to be. There are many issues raised on the mailing list, but only a few, typically one or two, usually respond to one type of issue. Instead of acquiring a consensus or complete agreement based on all the developers, this method is usually more effective. If anyone in particular has objections, they have a time span to raise them in.

Finally, all mails sent to the mailing list are stored in an archive. This archive is available for review online. This feature has proved essential in making older discussions and experiences available. It is an increasing trend to refer directly to the URL of an archived mail to answer a question sent on the lists. By using the archives, the individual developers do not have to store mails on their own mail accounts for future retrieval. The archives are not in wide spread use except for by some of the Norwegian participants. Theoretically the archive makes the mails searchable, but the current implementation has its limit. The address of the sender is hidden in order to prevent spam bots from getting hold of them. Other archiving software allows users to display mails in a threaded view, or per month basis, which makes navigation easier. The archive was strongly upgraded when the mailing lists were replaced with a new system during the spring of 2007, but the archives still lack a search function, limiting their usefulness.

5.3.2 Source code management with Subversion

DHIS 2 uses Subversion (SVN) as a source code management system. In many ways this is the most powerful mechanism for collaboration we have. It allows multiple developers to work on the same codebase, even the same files. If something goes wrong, one can always revert to older versions.

SVN has been set up to send e-mails whenever a revision is registered. Mails are sent to the scm@dhis.hisp.info mailing list. The membership of this list has been somewhat synchronized with the membership on the dev-list, meaning most, if not all, developers receive the notice. Each mail, or commit-mail, details which files where added, removed or modified, the log message supplied by the developer, by which developer and when. Several times, developers have read these mails, and forwarded them to the dev-list with comments like "Why was this done?", "It can be done better this way", "Is this really what we want?", "Take a look at these resources". As such, commit-mails represent an effective hook for starting conversations around the actual code.

An example of the communication which has taken place around the SCM mails:

```
[Developer A] wrote:
> Author: [Developer A]
> Date: [Some date]
> New Revision: [Some revision]
>
> Modified:
> trunk/dhis-2/dhis-web/dhis-web-commons-
resources/src/main/webapp/exception.vm
> Log:
> <> Made the stack traces in the exception page not
display by default. The
> user can click [+]s to display the stack traces.
Is this actually a good idea? I picture users copying and
reporting the part of the stacktrace they see, while the
important part of the data are way down in the trace. Then
we have to ask them to attempt to reproduce the error,
expand them and send us the entire trace again.
This depends, of course, on how many and spot on errors
being thrown out to the web GUI, and if our useful feedback
stems/will stem from reports from the console log trace or
exceptions seen in the GUI.
[Developer B]
```

In this example Developer A committed a change and Developer B criticized it. Developer A replies that he was unsure whether or not to commit this, and that he should perhaps have consulted the dev list first. The exchange resulted in a wider discussion about what sort of support structure one expected around the system, and thus what sort of errors to display to the users. A coordinator chimed in:

```
On [Some date], [Developer B] wrote:
> This discussion should perhaps spur a consideration on
how we generally
> organize the support system. Which people are users
supposed to report to,
> what should they report and where?
I think the end users will mainly be reporting problems to
the local implementors for now, and the implementors will
then report onto the list if they cannot figure out what is
wrong.
In the future, I do hope more users will be able to report
onto the users lists (either one for their country/language
or the general English one). My guess is it wil take
considerable time before that will happen at a noticeable
scale - but we should definetely aim for it, along with
more general discussions around the use of DHIS2 and
the data, as on the present South African list.
[Coordinator C]
```

What we see here is how the code changes and the SCM mails spark more general discussion about the surrounding infrastructure. Interestingly in this example, the infrastructure discussion is driven by the technical change, rather than by requirements put forth by users directly.

Finally, as noted by Nordal (2006:), the SCM list has been used to sanction bad practices in the use of Subversion. In his example a developer gets criticized for not supplying log messages, a short text explaining why a certain change was done. It is clear that some of the developers have expectations to how the SCM system should be used. These expectations are, for example in Nordal's case, rooted in past experiences with SCM systems and their use.

In a study, Nils Fredrik Gjerull (2006), discovered that of the five most active committers, who most often contribute source code into DHIS 2, four where from Oslo, while one was from Vietnam. I have repeated and extended this experiment and found basically the same results. There is a small tier of developers who are very active on the code, with one single developer standing out with close to 550 commits. This mirrors my own experiences and experiences documented in other studies (Nordal 2006): There is a core set of developers who contribute to the project on a consistent basis.

But using Subversion is not without its problems. The Ethiopian team reported that their firewalls didn't allow traffic on ports used by the SSH protocol. This is the most common protocol used for Subversion. As a result, they couldn't check out the source code at all. In the spring of 2007 a couple of Norwegian developers set up access to the Subversion repository using the HTTP protocol, but at the time of writing, the Ethiopian team has not started using it.

Another problem with SVN is that checking out code takes a lot of time on slow or unstable lines. Nguyen explains:

> *"Internet is something like a novelty in Ethiopia and maybe also other developing countries (India and Vietnam are different case as they have rather good Internet infrastructure but still very far from the Oslo standard)."* (Nguyen 2007:112)

This feeling has been echoed by several sources in my material, even by Norwegian developers using quite fast and stable lines. Luckily this is only a problem the first time you check out code. Subsequent communication with the repository is through small transfers of file differences, rather than whole files and directories.

Based on his experiences in Ethiopia, Nguyen once again suggests an alternative approach:

> "*We kept the source code in two laptops of two members of ART group and the backup was done locally in the laptops and by USB disks."* (Nguyen 2007:120)

One of the reasons for this was the problems with internet access. But the other problem is that team *"should use all of our energy to do the actual work rather than the management work"* (ibid). Setting up and maintaining a source code repository is thus perceived as extra, possibly irrelevant management work, which detracts from the actual work of producing the needed software. It's interesting to note that Subversion can be used on a local network where neither firewalls nor bandwidth are usually a problem.

Nguyen touches on another issue later when he states that *"learning a version control tool was in itself a complex task"* (Nguyen 2007:124). Part of the "management work", then, involves taking time to learn the workings of the tool. At the same time, both Nguyen and many from the Ethiopian team had attended the INF5750 course where Subversion featured prominently.

A senior software developer later commented on the development process and the lack of using an SCM system, arguing:

> *"Therefore, no back up of history of versions. All source code can disappear if the personal computer's hard drive damages. Not be able for multiple developers work at the same time."* (Nguyen 2007:128)

It seems that part of the problem of lack of usage is due partly to the infrastructure of the countries using it, partly to lack of knowledge of benefits and motivation for using source code management and partly due to lack of knowledge of using the tool itself.

5.3.3 Instant messaging: MSN, Yahoo! and GTalk

Instant messaging (IM) clients are used to coordinate between Norwegian developers on a daily basis. Furthermore, the Vietnamese and Norwegian developers have reported that they use IMs to stay in touch with each other after the Norwegians have returned from visits to Vietnam. IMs are thus used for social interaction as well as for technical questions. This social contact has also resulted in developers asking the Norwegians for help with programming problems directly on IMs, rather than write on the public mailing list.

An interesting point with regard to using IM as appeared in my research is first of all that different IM clients are more popular in different countries. For example, MSN is very popular in Norway, but rarely used in Vietnam. In Vietnam, Yahoo! Messenger is more popular. Conversely, Norwegian developers have mainly used MSN for their communication, while the Vietnamese developers have used Yahoo. To complicate matters further, the members of HISP India use GTalk internally, which introduces yet another IM into the mix.

A problem with the use of IMs is the time zone differences between the different countries. There is a 3.5 hour difference between Norway and India, and another 1 ½ hours between India and Vietnam.

IMs have also been used for several meetings where representatives from Norway, India and Vietnam have participated. The goal of these meetings has been to set priorities on different kinds of functionality, assign development tasks and set release dates.

In the spring of 2007 the team held an electronic meeting about the next milestone release. This time the entire meeting took place as a text chat. Several clients were tried, among them Skype, MSN and Yahoo. The first two proved to be problematic when participants once again fell out of the discussion. They would appear as inside the conversation even after they had left and re-joined, but found that they couldn't write into it. The chat ended up on Yahoo, which proved to be more stable when participants fell out. The log from the conversation was saved and uploaded to the wiki. One of the participants set up a page with notes from the meeting and invited others to improve it. Two or three people made additions and corrections to the notes.

5.3.4 The Hisp.info wiki

The HISP network has a wiki which resides on http://www.hisp.info. This wiki houses lots of information about HISP, BEANISH and DHIS development. In relation to DHIS 2 development, it houses the general development plan and the documentation for the system. The documentation is located in its own space in order to be more visible to users. Instead of writing static HTML files, developers may edit the documentation quickly and easily. This could have been solved by for example source code management for the documentation as well, but this kind of software requires a certain amount of technical know-how. Through the use of the wiki, it is possible to recruit non-technical people to contribute to documentation.

The use of the wiki is somewhat unstructured. It fulfills many purposes and targets many different audiences. It's the primary web page for the BEANISH project, often targeting EU bureaucrats. It's the main site for DHIS 2 development, housing information on the roadmap, releases, functionality, requirements specification and the like, targeting both users, developers and potential customers of the system. It also contains information about the HISP network and its research activities, thus aiming for the participants in the network. hisp.info, it seems, has become the primary website for the global network of HISP itself, supplying outsiders with information about the network. The wiki is thus filling the role of a Content Management System (CMS) more than a wiki.

There are a couple of interesting aspects and developments that deserve some notice: The wiki is used partially as a file sharing service. Documents, scripts and software bundles are uploaded to the server. It should be noted that a different server has functioned as a file server for HISP and DHIS 2 development: http://hispkerala.org, which is administered by a coordinator in Oslo and basically just gives the user a list of uploaded files. HISP India also uses their own server http://uploads.hispindia.org for this purpose. Thus there is not one central place to find files.

The Vietnamese team reports using the wiki for status reports, informing others of where they are, how they can be reached, what they're working on and what they plan to do.

Facilitators from South Africa have embraced the wiki for quick sharing of information and status. One of these facilitators has become much of a champion for the technology. He attempted to set up a similar space for members of HISP Nigeria, but this has not resulted in any activity from this group. Similar spaces have also been set up by coordinators from Norway, but these have not been used by the local teams.

5.3.5 Issue trackers: JIRA and Trac

Software under active development is often followed by an issue and/or bug tracker. Such a piece of software allow users and developers to register issues (also called tickets), such as a feature request or a bug report. Administrators or managers may then comment on the issue and at some point assign the issue to a specific developer. The developer may then register work on the issue, and optionally close it when work has been completed.

In DHIS JIRA is used as an issue tracker. This is supplied by the same company which develops the Confluence wiki software. The advantage to this is that the two pieces of software are closely integrated, for example allowing direct links to issues in the wiki.

The ideal process would be that every decision made in other forums, such as in meetings or on the mailing lists, should be transformed into an issue. This way the project managers can keep track of the progress. JIRA also allows issues to be grouped into different releases of the software, essentially allowing managers to decide which bugs should be fixed and which features should be implemented in an upcoming release. The reality in DHIS 2 development, however, is that this software is rarely used. Issues are registered sporadically and rarely assigned to specific developers. Progress has almost never been reported. JIRA supports sending mails to affected developers when issues are registered or modified. It has not, however, been set up to send mails to a larger audience of developers, which effectively limits the public discussion and awareness around issues in the tracker.

A different issue tracker called Trac was introduced in the spring of 2007. It has a simpler interface then JIRA and mail notifications work without problems: Every time someone adds or changes a ticket, a mail is sent onto the dev-list. The tracker comes with a built in wiki and a source code browser integrating with Subversion. At the time of writing, the issue tracker has almost 150 tickets and is in active use as a coordination tool. Most of them have been submitted by Norwegians, however. Time will tell if this tool sees more widespread use.

5.4 Communication practices and general issues

In this section I go into material which does not directly relate to the usage of tools in DHIS 2, but rather to some general issues with communication in the project and among the nodes.

5.4.1 Tool use in India

The mailing list sees traffic primarily from two people in India. One of them, a coordinator I interviewed explained this in the following way: HISP India has an internal mailing list on which to handle local issues. The various facilitators, implementers and developers in India will report problems to this mailing list first. There our two representatives will attempt to answer them based on what they already know. If they cannot answer the problem, they consult the dev-list. This way, India has organized a first and second line support, filtering requests before they are sent to the core developers. The mailing list has seen diverse activity over the last two years, related to local planning and strategic issues, the health domain, and some technical discussions about different information systems HISP India need. One facilitator noted that this list also had some problems with language. Mails on the list are supposed to be written in English. Some members are not proficient with the language, however, and have tried to communicate in their local language, causing reactions from other members. While I have only cursorily inspected the list, it seems the tone is more based on orders from the coordinators more than on open discussions between all the participants.

During the summer of 2007, the activity was strengthened with an internal issue tracking system for both technical issues and local implementation issues. The goal was to make sure issues were reported and followed up properly. At the time of writing, this system has seen little activity.

Finally, HISP India has a large website which contains information about the teams, the implementation work, user manuals for the software and so on. This website is largely maintained by the national coordinator. A part of the site, however, is used for file sharing and is being used actively by the members.

5.4.2 The language barrier

Language is an issue in DHIS 2 development. The project has developers from different countries and with varying command of the English language. Norwegians receive several years of compulsory training in English in school and are usually quite proficient. This is not the case in India and Vietnam with, varying English skills as a result.

One Norwegian developer working in India reported that members of the team had problems understanding the mails on the dev-list and the text on the wiki. With regard to this, he stated:

> *"I think the biggest problems are terminology words bound to DHIS 2.0 frameworks and tools as well as Concepts from the DHIS itself."*
> - Norwegian developer, by mail

So we're dealing with two different aspects: A general language issue and one of understanding the basic concepts.

One Indian developer describes a "gap between understanding". He expresses it as a two way problem:

> *"The problem is not understood by you people. The solution that you're telling is not understood."*
> - Indian developer

He explains that when people from India write mails to the dev list, the developers from Oslo ("you people") have problems understanding what they're trying to say. And when they try to communicate solutions back, or ask questions or give instructions, these are often not understood by the Indians. He put the reason for this as general language problems. He also explains that very technical mails, for example related to the WebWork framework are difficult to understand. I asked him to give a rough estimate of how often responses are understandable and he put this at 60%. When asked about how he handles this problem he said he'd often try to clarify or communicate for two or three exchanges, and then give up if it's still difficult to understand. If he's asking a general question, he can often draw on his own knowledge and try out different things. But if the question is related to DHIS 2 technologies, he has no alternative than to rely on the developers from Oslo.

But in relation to the gap in understanding, he also says:

> *"Most problems cannot be solved through mail."*
> - Indian developer

The idea is that mail isn't enough to effectively convey problems and solutions. I asked him how this could be improved upon and he suggested several solutions. He explains that in his country they communicate using voice, not just mail. This way they can explain problems more clearly and have a dialog about them. He suggests, as they've done on a few occasions, to use remote login. The person helping will remote login to the user's machine and take a look at the user's screen. This way it's much easier to get a feel for the problem.

Another solution, he says, is that people come to India and see the problem for themselves. These people can then function as mediators with those back in Oslo. This is a point which several of my informants make in different contexts: Instead of requiring everyone to communicate with everyone, it may be better to establish some people as middle men, preferably people who are physically present.

5.4.3 The DHIS 2 codebase

Part of the explanation between the different levels of communication may be how the participants feel about what they're collaborating around: The code of DHIS 2. In this section I refer some comments and issues related to the development process and the code itself. I propose that if a participant doesn't understand or feel comfortable with the codebase or the technologies, he or she will have difficulty in participating. As one informant notes:

> *"It's difficult to ask questions when you don't know what to ask."*
> - Norwegian developer

Much of the capacity building effort with relation to DHIS 2 has been directed at teaching the tools and frameworks as well as how the code in the code is organized. This has not increased the degree of participation in core development or on the different communication tools. Furthermore, many informants and participants have noted that the DHIS 2 codebase and the technology stack it is using is difficult to understand. When asked to start on some functionality, one developer stated:

> *"I am not very sure about this. The technology is so complex. I am afraid of it very much. Let other developers [Oslo team] do it".*
> - Vietnamese developer, cited in Nguyen 2007:100, his changes.

This problem is complicated by the fact that it is difficult to find developers in Vietnam, India and Ethiopia, which have experience with the frameworks involved. While the Java programming language is more wide spread, it still means that the developers must receive a lot of training before becoming capable of contributing code to the system.

Another problem lies in the fact that the system has changed several times, often in radical ways. Modules are re-arranged, packages changed and APIs are modified. During the spring of 2007, one specific problem was raised: As DHIS 2 is database independent, the core developers had not paid any particular attention to the generated database structure. In India, however, partly as a result of development taking too long, they'd begun making their own applications which used the DHIS 2 database structure. This structure would often change, meaning all these local applications would need to be updated frequently. Many of the Norwegian developers maintained that Indian developers shouldn't be accessing the database directly, but rather go through the DHIS 2 Java API.

But as HISP India needed functionality quickly, and because they found the DHIS 2 structure and technology stack difficult to understand, they felt they had to develop their own applications. This points to a concrete, practical problem, but one developer from Vietnam also made an important point:

> *"Every time the system changes, many developers have to re-learn the structure."*
> - Vietnamese developer

The database conflict also highlighted some problems with the asymmetry in use of the communication tools. The Indian developers didn't use the source code repository for their local application, nor any other local repository. No notice was given on the mailing list that these applications were being developed. Subsequently, when one Norwegian coordinator received word that they had been developed, this came as a surprise to the Oslo node. The Indian developers were asked to check the source code into the repository so other developers could look at the code, and contribute to it if they wanted. This would have been especially beneficial to keep the software updated along with the rest of the codebase. When asked, an Indian developer did indeed check the code into the repository, but did not continue to work against the repository with his updates. Instead, the application received updates and functionality to local copies in India.

5.4.4 Communication in HISP and alternative models

Communication in the HISP network itself is somewhat at odds with the communication in the DHIS 2 project itself. E-mail is used extensively both in and between the nodes in the network. I've observed several mail discussions where the original author selects a limited set of recipients. As the discussion progresses, more people are added to it. This happens as participants notice that the content of the discussion might be useful to other members, or when other members may have something interesting to add to it. In some cases, members have been included in, for example discussions about new applications, with the intent of piquing their interest and hoping that they will pick up the task. Additionally, some members of the network maintain frequent contact through individual mails, or discussions between small groups of people.

This process of communication has had some issues. Members of the network have complained that information doesn't flow to enough people. This has manifested itself in many in the network not having known what has been going on elsewhere in the network, contrary to the goal of the network itself. The workshop discussion in South Africa is one illustration of this.

At the same time, there are mailing lists available which go to most members of the network, hisp@hisp.info. This has seen some activity, but not a lot. Much has been reports of interesting articles and the like, but little discussion or information on activities in the network. During the late spring of 2007, one facilitator took the initiative to establishing a newsletter to be sent out on the public mailing list. At the time of writing, the editors are still waiting for input from the different nodes as to what to put in the newsletter.

Some heated debates have taken place on e-mail, using the process of adding recipients when it has felt natural to the author. One such debate turned to the communication practices itself where one person suggested that people needed to be more active reporting issues on public mailing lists. He got a response saying that mailing lists themselves are not the answer to everything, pointing to other issues being more important. At the end of the discussion the list of recipients was quite long, in addition to some responses being forwarded to yet other members privately.

At the same time, mailing lists aren't always useful: They can cause information overload, and their public nature excludes discussions of sensitive information. In such cases, it's better to restrict the communication to the few people who need to know what is happening. With the network's general practice of communicating privately, it is sometimes hard to understand when privacy is required and not. A point I experienced personally when I forwarded the contents of a private discussion onto a mailing list to spur a different discussion. It seemed to be to be safe, but it turned out that is wasn't.

Another mailing list was also introduced during the spring of 2007, called coordinators@hisp.info. It was intended as a place where the people higher up in the network could discuss less technical issues.

The role of the HISP network and its activities also has an effect on DHIS 2 development. One core developer explained how he felt that the communication tools and practices wasn't a problem in his view. As he put it

> *"If someone doesn't dare to write on a mailing list, it's not the communication medium that is the problem. It's the person or the background"*
> - Core developer

He went on to say how the trouble the network sees is caused by a lack of or unsuccessful capacity building in the local nodes and poor leadership and organization at the top level.

Another informant described how the websites of the different nodes in the network were so different and how there is no unified website for HISP. This, he commented, reflected the actual organization: No formal organization and weak leadership. Taking this view, we can think that the asymmetry in the usage of the collaborative tools is just a symptom of a deeper problem in the organization. The idea is then to fix the organizational problems, which should result in better coordination in the network. A more unified set of websites would in his view appear as a result.

A coordinator from India explained that several from the Indian team would rather send mails directly to one of the Norwegian coordinators. This because they'd met him face to face, spent some time with him and felt they knew him. The coordinator echoed this explaining that he and some of the Indians had become friends. In this, instant messaging was essential. They would frequently communicate over Skype or MSN. This could also be about non-work related issues. For several of the participants, this felt safer than using the public mailing list.

The Indian coordinator suggested cultivating this type of contact, and extending into a formal system. Instead of using a public mailing list, which many weren't comfortable with, HISP would assign one person responsible for communication with one other node in the network. The Norwegian coordinator, for example, would be the direct link to the Indian team. This coordinator would then be responsible for compacting, filtering and abstracting the information and then pass it on, for example to the mailing list.

A similar method is already in use on the internal mailing list in India: An Italian researcher working with the teams compacts information from the state he is working in and reports it for discussion on the mailing list.

5.5 Face-to-face meetings

A general problem with electronic communication based on text is how it is not as rich as verbal and/or visual contact. Intonation, sarcasm, humor, body language and facial expressions are examples of aspects that are lost or misinterpreted in textual communication. Statements may be perceived as overtly harsh or not as serious. Electronic collaboration is supplemented by face to face meetings. People from all over the network are sent to other nodes to participate. An example is how master students and developers from Oslo have gone to for example Vietnam to assist in development and implementation efforts. According to several of my informants, meeting a person, if only once, and getting to know that person, aids you when you communicate later.

5.5.1 Travel and capacity building

One Norwegian coordinator, a core developer, and three Norwegian developers have visited India. The coordinator makes frequent visits to the country and has gotten to know the team. One Vietnamese developer has made two visits to India, and three Ethiopian developers have also made the trip on two occasions. These visits were not DHIS 2 related, but concerned the development of a different system. One Indian coordinator and one Indian facilitator have visited Norway several occasions. The coordinator splits his time between the two countries.

The Vietnamese developer has also visited Ethiopia, again to work on a different system together with the team. One Norwegian coordinator and one core developer has visited Ethiopia on one occasion to discuss requirements and issues.

A total of three core developers and four Norwegian developers and several Norwegian coordinators have visited Vietnam. This node has been the target of extensive capacity building. On the one hand, the developers have knowledge the local developers don't have, for example a more detailed understanding of DHIS 2, on the other hand the developers learn about more how the software is used in the local context, what issues face the local implementers. It also means they get to know the people they later communicate with electronically. The capacity buildings have taken many forms as described in Nordal (2006), Øverland (2007) and Nguyen (2007), ranging from lectures to programming exercises to hands-on assignments.

Finally, several participants have traveled to Norway and taken the INF5750 course, among them one Vietnamese developer and three Ethiopian developers. The Norwegian developers have all taken this course.

5.5.2 The Oslo office

The larger part of DHIS 2 development has taken place in Oslo, and most of the core developers and the project management are located there. There is an office at the University of Oslo where both the coordinators and many of the developers have been sitting together and working. This means both a lot of tacit and explicit knowledge has been passed between the participants. A lot of electronic communication is preceded by or supplemented by direct discussions between those at the office.

Several meetings have been held where decisions about the development have been made. It is then a major challenge in communicating this to the rest of the developers and including them in the decision making. This was a major criticism raised against the project managers in the early stages of the project. One solution which I suggested and implemented was a specific wiki page where decisions could be written down. This has not been very successful for several reasons. One could be that it is difficult to define when a concrete decision has been made which warrants that it is place on the page. Another is the question of who updates it. Those who are active are also very busy, making it difficult to maintain such pages. A different issue is one of discipline where one could say there is not a strong enough culture for taking and publishing notes from face to face meetings.

The Indian and Vietnamese nodes also have offices gathering together the local developers. The office in India houses the national coordinator and one central Indian developer, plus visitors from the various states.

5.5.3 The focus group meeting

The use of electronic tools for collaboration became quite apparent when the DHIS 2 team attempted to arrange a focus group about the development process during the spring of 2007. There was a perceived conflict in how the different nodes involved in the development wanted to structure the process, and a desire to learn about each other's views. One participant suggested setting up a focus group and invite representatives from the different teams. He suggested choosing a few topics for discussion, basing it in a real case. At this time, one representative from India was on a visit in Oslo, which meant an excellent opportunity to have some face-to-face communication as well. But one also wanted to include more people, and the solution became to set up a multi-way Skype call. At first the idea was that everyone could participate, so we set up loudspeakers and microphones in the Oslo office. At most, five people were connected through the Skype call, from Vietnam, Zanzibar and India, while five people sat in the Oslo office. Some problems became apparent immediately, for example that the internet connections of the various participants were low-end. In one case it was impossible for a participant to speak clearly across the line. Several of the participants fell out of the call during the conversation due to unstable internet connections and had to be invited again. In the end it primarily became a debate among the people in the office. Two participants tried to get into the discussion by sending text messages in a separate Skype chat. These points of view would then have to be relayed by a person monitoring the chat in the Oslo office. One participant sitting in India managed to speak through the loudspeakers, but there was a significant time delay.

The meeting moved away from a focus group to a more general discussion about the development process, and the needs for support in the various nodes. One particular issue was the use of direct connections to the database underlying DHIS 2. The core developers had intended others to access the data in the application through a Java API. The use of the Hibernate framework abstracted away the database layer in the application. The Indian team found the API to be cumbersome to use, and had in addition little experience with the technologies the system was implemented with. At the same time they faced increasing requirements for functionality in various Indian states which created a need for shortened development time. They had developers on staff which were familiar with other technologies than the ones primarily used in DHIS 2, which made development quicker. This however, required that they accessed the database directly. A problem arose where DHIS 2 developers would not pay attention to what happened to the generated database structure when they introduced changes in the system. This resulted in various home grown applications in India, among them some developed by external companies, breaking when the database structure changed.

Two different traditions and viewpoints seem to conflict in this case: On one hand the core developers aimed for a well-organized and maintainable system, using a locked set of technologies. This resulted in a slow development cycle. The Indian team, however, needed functionality quickly in order to demonstrate the system to various state officials. It was more important that new functionality was developed quickly in a prototyping fashion than using the "right" set of technologies. A sort of compromise was reached: The DHIS 2 developers would lock down database names and structures and avoid changes to the code that removed or changes tables or fields in the database. At the same time, efforts to support the Indian team in using the DHIS 2 technologies would be intensified.

The meeting also highlighted some differences in the way the two teams communicated. The Norwegian core developers were used to using the mailing list and code repository to communicate changes and activity. During the meeting the India team mentioned several applications they had developed locally which the Oslo team didn't know about. These had not been communicated publicly, and only partially to coordinators in Oslo. The code repository was not being used for any of the applications even though they were modules to be used as part of DHIS 2. So, while India was actually involved in a lot of development activity, the Norwegian team thought the Indian team was not working on code.

When asked, one Indian developer did indeed check his application into the repository, but did not initially continue development there or communicated much about it on the mailing list. It was later explained that this particular developer was a little bit shy and also had some problems expressing himself in English, which explained why he wasn't active on the list. Additionally he had been moving around India with varying net access, which also impaired matters. This point to personal matters and comfort levels as an important notion about mailing list activity. After several requests he started using the repository for his daily work on the module.

5.6 DHIS 2 as a FLOSS project

DHIS 2 is released as FLOSS under the BSD license. I interviewed some of the participants with regard to their view of FLOSS and the role of FLOSS in DHIS 2. One core developer stated that his reason for working with the software was that he found developing, making something, fun. When asked what he felt about FLOSS he initially responded that he hadn't thought about it. Nordal (2006) had lots of experience with developing FLOSS and had strong opinions on the choice of a license and the FLOSS profile. For him, it would seem FLOSS was an important driving force. A Vietnamese developer focused more on the monetary and practical side of FLOSS. He commented that it was easier to get support from users for FLOSS software, mainly due to the fact that it's free.

> *"If you use Microsoft, you have to pay. First for the tools, then for the software, and then the users have to pay for the software."*
> - Vietnamese developer

As such FLOSS gives the project a competitive advantage. When asked how he specifically felt about the ideology of free software, he said he wasn't interested in that. It felt too abstract to him.

My own motivation is partly the same as the first developer: Developing and gaining some experience. My relationship to FLOSS is more a negated one: I don't see why the code of this system, or many other systems, should be proprietary or cost money. This is a weaker relationship than having a principle view that software in general should be free.

In their evaluations of the course, the INF5750 students report that working on a real FLOSS project is something they see as a valuable experience. This is something they feel is both inspiring and relevant to future employment.

At the same time, one of the Norwegian coordinators pointed to the idea of freedom as a central idea in the startup up of DHIS 2 development. According to him, it was a necessary aspect of the software in order to realize true South-South-North collaboration. Another reason for choosing a FLOSS process was also to be able to attract external FLOSS developers into the project. In a report from the BEANISH project to the EU (BEANISH 2007), the various communication tools were lifted up as a platform for communication and knowledge sharing:

> *"These tools together make it possible for both developers and practitioners to share knowledge and code globally. As BEANISH is establishing technical teams in several*

countries, ready access to a common communication platform is of the essence."

The fact that the software is free plays an important part in the choice of process, but also especially that users and governments are free to modify the source code, which in the coordinator's view was a way of empowering them.

The fact that DHIS 2 is release under an FLOSS license gains the project some advantages. The Confluence wiki used on hisp.info, for example, is under an Open Source license from Atlassian. This license means that the software is free as long as the project using it is a FLOSS project. Being FLOSS thus gives the project some material benefits. Several countries and cities have recently made explicit moves to FLOSS software, citing cutting costs and improving quality as reasons. An example is the state of Kerala in India which made a commitment to FLOSS software, meaning all software used in the public sector must be FLOSS. The Vietnamese government created a Master Plan using FLOSS software to, among other things, counter software piracy in the country (Øverland 2006). The fact that DHIS 2 is released under a FLOSS license was vital to enter this and potentially other markets.

6 Discussion

In this section I return to my research questions and discuss the material in relation to them.

> **Primary research question:** How does a FLOSS communication model affect global software development projects and how does this model relate to other communication models?

It would seem that the DHIS 2 project is somehow caught between two different communicative practices, with various forces pulling in different directions. On the one hand we have the FLOSS phenomenon and the traditions thereof. DHIS 2 was started as a FLOSS project and has a FLOSS license. Its communication structure is similar to that of FLOSS projects: Most communication is electronic. Communication is open and public, in a forum accessible by all: The mailing list, the issue tracker, the wiki all enable many-to-many communication practice.

On the other hand it would seem HISP has a different tradition. Mails are sent to direct recipients, not to a list. Archiving is achieved only in the mailboxes of the various participants. The author of a mail adds recipients based on a private conceptualizations of who the mail applies to and people that can contribute. One could say that communication is one-to-few or few-to-few. This has led some people in the network to complain that decisions are not visible or transparent, because they are made and communicated in a small circle of people. On the other hand, one part of the reasoning for this is that not everyone is interested in everything. In fact, many of the participants in the network are very busy and receive large volumes of mail already. Several informants noted that there is more activity on the dev list than they can cope with, some of it very specialized. One informant noted that he had simply stopped subscribing to the list.

> **Secondary research question:** How are electronic communication tools used in globally distributed development?

Much of my empirical material describes in detail how tools are used and not in the case. The following section discusses the findings in light of Sahay and Sarker's framework and some alternative tools and practices.

I discuss the secondary research question first as this helps inform the primary question.

6.1 Use of electronic tools in Global Software Development

In this section I summarize the findings from the case and comment on their validity. As mentioned, we cannot generalize in a positivist way and formulate social laws and make predictions based on a single case, but I still believe we can draw out some implications from the case.

As we have seen, the use of the tools is very skewed. The Norwegians dominate when it comes to checking in code and sending mails on the mailing lists. Many of the Norwegians are students in the INF5750 course, but the impact of this group stems mainly from being so many. Individually, the contribute little in terms of volume of communication, and their participation is confined to short periods.

Generally, software development is carried out by a small group of developers. The top five committers have produced 50% of the code in the system. Four of these are Norwegian, one is from Vietnam. Three of them are core developers. This seems to be in line with findings in Weber (2005), and conforms to an 80-20 rule which is, interestingly enough, common in FLOSS projects. One single core developer also has also produced a large part: 22% of the total code. This complicates matters as a lot of knowledge about the system rests inside the head of one person. Interestingly, this has been some of the criticism against the DHIS 1.x development process, where one developer primarily knows all about the system and technical documentation is mostly lacking. It's also clear that most of the contributions of the Vietnamese and Indian teams relate to local modules and translations rather than the core system.

Participation on the mailing list is more diverse than in the repository. More people from more countries participate, although the Norwegians dominate here as well. It's not wholly unexpected, as there are several less technical discussions which other people can participate in. Among these, there are discussions about requirements, which is initiated by coordinators or facilitators which work with the field. It still seem the list is primarily used for technical discussions and support requests, as these categories comprise 26.7% and 25.3% respectively of the total number of mails investigated. Norwegians are most active in initiating discussions on the list. At the same time, over a third (35.5%) of the mails on the list get no replies.

From the case there appears to be many reasons why participation is so skewed. On the one hand, technical infrastructure in the different countries varies greatly, and is often too weak to support active usage of for example source code management. Some participants report finding the list overwhelming in terms of sheer volume of traffic. It has also been described as too technical, and too specialized to interest everyone. At the same time some participants feel that when they do ask questions, they feel that they are not getting useful answers back, for example the replies will be too technical or require them to search for more information. On the social and personal level, some people are simply shy and have difficulty communicating to a large group of people, even when that group is electronic.

The source code itself is described as difficult and complex to understand, despite capacity building efforts. There are a number of frameworks involved in the development which means each developer must go through a long process of learning to be able to write code for the system. Additionally, the frameworks used in the project are not well known in Africa and Asia. Finally, the architecture has changed many times, usually initiated by the Norwegians. This has made it difficult for other developers to keep up.

Instant messaging is used to communicate directly between developers. This is the result of and builds on social contact. Those participants who have met each other face-to-face stay in touch over IMs. This seems to be the same as seen in Imsland and Sahay's (2005) case. In addition IMs are used for meetings when people from different countries participate. This last practice has not been without its problems due to unstable or slow internet lines.

We can look at the case in the lens of Sahay and Sarker's phases. They define different phases a team can go through: Initiation, Exploration, Collaboration and Culmination. If the task at hand is to develop DHIS 2, then it's clear that the Oslo team has a high degree of *task focus*. As the other teams are focused on local modules, only usable in their own country, then their task focus level is lower. As for task ability it is also clear that the Norwegian team has a high degree of this. They have chosen the tools and frameworks and have received extensive training in their use. Despite capacity building efforts, the other teams are not as proficient. Each phase is described in terms of four dimensions: Social responsiveness, virtual presence, shared goals and identity. In the case, we see that the Indian and Vietnamese teams contribute around a total of 17% of the mails on the list. Thus that there is not an awful lot of communication between the teams. The fact that one of the teams did not find the list useful at all is also a strong indicator of a lack of communication on it. It is reasonable to classify the social responsiveness as bilateral or even unilateral rather than mutual.

The virtual presence is high from the Oslo side, as most of the members are proficient with and feel comfortable with the tools, but lower from the other teams. Many from the Vietnamese and Indian teams are available on IM, however, which all in all makes for a strong virtual presence. As the teams are all involved in DHIS 2 development, we must assume some degree of shared goals. The low activity in the repository from the Indian, Vietnamese and Ethiopian teams however, could indicate that the other these teams do not share the goal as strongly. On the other hand it may indicate that the goals have been set from the Oslo side, rather than in unison. At the same time, the difficulties with the codebase may mean that the other teams do share the goals, but are not properly equipped to work toward them. The lower degree of task ability could also indicate this. All in all, it is difficult to conclude definitively based on the available data. As for common identity, it's quite clear that the individual teams have a strong identity. The organizations are called HISP India and HISP Vietnam. As seen in the repository, they contribute mostly code which is specific to their countries. Interestingly, there is no HISP Norway or HISP Oslo. The developers there seem to identify with the DHIS 2 *project* rather than a specific node. All in all, the shared identity seems to be low.

This combination of characteristics, bilateral social responsiveness, high virtual presence, potentially weakly shared goals and a weakly congruent identity seems to place the project in an Exploration phase. The project still has some way to achieve what Sarker and Sahay see as proper collaboration.

6.1.1 The right communication tools for the job?

As a way of learning from the case, we can ask the question of whether or not the tools used in the project are the right tools for the job, i.e. if the tools themselves have a negative impact on the usage, or if they are unfit for certain kinds of projects.

Instead of using a mailing list, one could use a web forum, for example. Both are asynchronous media. In the early period of DHIS 2 development there was indeed such a discussion. The argument against was the push-pull debate. Some developers argued that they didn't have time to actively go out and find information, i.e. log in to a forum, find threads and read them. Instead they preferred to have mail delivered to them with having to go out and actively look it up. A different argument was that some of the developers already participated in other FLOSS projects. Here mailing lists are the norm, and a forum would mean fragmenting where they did their work. The counter argument at the time was that mailing list discussions can become unstructured when many participate. Organizing discussions into threads was also an important point. The opposite point to the idea of fragmented information was also raised in that discussion: Some participants felt that too much was coming into their mailbox and that they never felt they could catch up. For some it meant that they stopped being active altogether, feeling overwhelmed. They preferred a forum because they could visit it when they had time and browse threads organized by subject, and follow discussions in a thread. These points, both positive and negative, have been repeated by other sources in my material: Information overflow is a problem to those who want to read the list in a casual way.

It's difficult to say that this means that either tool, mailing list or forum, are right or wrong for the job. Today, many mail clients support threading. GMail, which is becoming increasingly popular, organizes everything into conversations. Most mail clients support organizing mails into different mailboxes through filters. But is does require the user to do some configuration. Interestingly, many of the most popular web forums rarely make use of threading in discussions, ordering posts in a sequential order instead. This means the problem of being overwhelmed can be reduced. On the other hand, the problem of fragmentation can be reduced by utilizing "watch" mechanisms on forums so that the user is notified by mail when someone makes a new post. If such a notification carries a link, it's simply a matter of clicking the link to be able to read the discussion.

The problem with the list being too technical and having too much traffic can be alleviated by supplying other forums for other kinds of communication. Attempts have been made to achieve this, for example the introduction of a users-list, intended for user feedback, bug reports and the like, and a coordinators list, intended for high level, non-technical discussion of strategic priorities, prioritization. Neither of these lists have seen much activity at the time of writing. This is possibly because no one has informed clearly about the purpose of the lists or pushed for their use. A more deep-set problem may be that many participants are not used to, do not like or are not comfortable with using mailing lists as a tool. In HISP e-mail is used frequently to communicate across borders. E-mail uses relatively little bandwidth and works in an asynchronous manner. Thus it can function well as a tool in countries with slow lines.

Part of the choice for adopting mailing lists in the case was to attract external FLOSS developers. Mailing lists are still the bread and butter of FLOSS development as Fogel (2005) puts it. This means that by adopting the same tools as those used in FLOSS development, FLOSS developers will have an easier time getting into DHIS 2 development. Similar motivations in other projects may motivate the choice of a mailing list over a forum.

Another problem as pointed out by informants in the case was that text could be perceived as harsh. Web forums alleviate this by supplying a vast array of emoticons which can be used to express tongue-in-cheek statements, laughter and so on. These emoticons have evolved from pure text equivalents which can also be used in e-mail, but in this case the participant has to discover and learn them. In the case of forums they are often readily available in the user interface.

Imsland and Sahay (2005) point to some issues with mailing list membership, where the company bosses were present on a list between Norwegian and Russian project managers. This resulted in a sort of self-censorship by the Russian project manager, projecting the view that everything was ok, when the project was indeed late. A potential implication is whether or not you create spaces for communication where "leaders" are not present, affording more relaxed communication among other participants. Some similar solutions have been seen in Imsland and Sahay's case (ibid) with the use of ICQ, and in the DHIS 2 case with IM communication in general.

As we've seen, people from different countries are used to using different IM clients. In Norway, MSN is generally the most popular while people in Vietnam use Yahoo! Messenger more. In India, the HISP team uses GTalk internally. This causes some problems with keeping in touch using synchronous tools. One obvious solution would be to standardize on one client. Every developer should have an account on one of the IMs. This could work, but it would require many developers to monitor one IM network in addition to the one they're using in their everyday lives, and in addition to what they use to coordinate with their local developers. A perhaps more flexible solution would be that everyone registered accounts on each IM network, but that would require people to have several IM clients open at the same time. It would also make it difficult to communicate across three different countries, for example India, Vietnam and Norway. This can be made a little bit simpler by using multi-IM clients like Miranda or Pidgin. These allow the user to define accounts on different IM networks, but use the same program to talk in each of them. It still doesn't solve the problem of communicating across more than one protocol, but it does keep everything in one place. Currently, for example Pidgin supports MSN, Yahoo!, GTalk, IRC, ICQ and many others.

The alternatives to using SCM software is either having a single developer on a module of code, or transmitting and merging files or diffs manually. Such a process is used in the development of DHIS 1.x. This is not without its problems, as it is easy to overwrite files and changes. It works because the number of developers is low and disciplined. Network bandwidth and stability are also issues here, as reported by Nguyen (2007). At the same time, there are several advantages to SCM software such as maintaining a revision history and merging changes made by different developers.

6.2 To FLOSS or not to FLOSS: Is DHIS 2 FLOSS?

In order to discuss the question of the effect of a FLOSS communication on global collaboration, I first discuss whether or not the DHIS 2 project can be perceived as a FLOSS project, and what the consequences of this may be.

Instead of viewing FLOSS as one thing, one can say FLOSS has many aspects. The perhaps simplest and most definable aspect is the legal one. Something is FLOSS if it is released under a FLOSS license, granting the user the right to modify and redistribute the code. For some people FLOSS is an expression of an underlying ideology, for example an idea about free speech, democracy or programming as an art form ("free as in freedom", cf Raymond 2000 and Weber 2005). For others it's a practical dimension that's important, specifically that the software is available for free, known to the FLOSS community as "free as in beer". For others FLOSS may not be something one is or produces, but something one does, a dimension about the process of doing FLOSS. This may include using public mailing lists (as noted by Fogel 2005), open source frameworks or similar. This point to an idea of FLOSS as an onion of meanings.

DHIS 2 is released under a FLOSS license so in all legal aspects, the system is thus FLOSS. In the following, I'll discuss other aspects of ideology, rhetoric and division of labor.

6.2.1 FLOSS ideology in DHIS 2

Stephen Weber claims that *"The principle goal of the open source intellectual regime is to maximize the ongoing use, growth, development and distribution of free software"* (Weber 2004:84) Furthermore, the intellectual property schemes aims to empower users by ensuring access to source code, pass a large proportion of the rights to use to code to the user and constrain users from putting restrictions on other users with regard to property rights. The most extreme variant is the GPL license, which stipulates that all code based on GPL'ed code must also be GPL'ed. So, if Weber is right and DHIS 2 is an open source project, then we should expect to find a view among the project managers and developers of DHIS 2 which sympathizes with spreading FLOSS software.

Weber argues further that *"[L]icenses act as a practical manifestation of the social structure that underlies the open source process"* (Weber 2004:85). The discussions about licenses in the general open source community relates to different views on intellectual property rights and moral and pragmatic issues around the spreading of FLOSS. Kristian Nordal (2006:61) recounts a discussion around licenses for DHIS 2 as one where few people participated. The options were GPL and BSD. He notes that DHIS 2 being a FLOSS project was never questioned and that people seemed not to care as long as it was an OSS license. But there is a question here of whether or not this means that it was the question of licenses which was not interesting to the developers, or whether or not the idea of being FLOSS was in itself not interesting.

At the same time, DHIS 2 inherits a strong ideology from the HISP network. The underlying ideology of HISP is promoting the districts and empowering users. This is accomplished to a system where the users may define data elements, organization units, reporting frequencies, and so on, themselves. In many ways this view of empowerment is very much aligned with the ideology of FLOSS: Users are empowered through giving them access to the source code and allowing them to change it, basically letting them make the program into whatever they need. This was, somewhat ironically, simpler with the 1.x versions based on Microsoft products. These products emphasize building user interfaces in a click-and-drag environment, rather than writing source code. This makes it easier for the actual users of the software to customize it and develop new, although limited functionality. With DHIS 2 you need to know a series of technologies and frameworks, and especially be able to program in the Java language.

Another perspective is that of the leaders in HISP. Based on my own observations, it seems these are less interested in the ideological side of FLOSS and more interested in the practical and monetary side. The leaders are not well versed in FLOSS licenses and use FLOSS primarily as a sales argument when attempting to get into new markets.

So, we can say that the motivations are varied. Based on the conversations and interviews I've had with several participants, it doesn't seem that a FLOSS ideology of free as in freedom is strongly present in the network.

6.2.2 FLOSS rhetoric in DHIS 2

Mats Alvesson (2004) makes a point about how rhetoric is important in relation to knowledge-intensive firms. It's not that one is knowledge-intensive, but that one makes a claim to be so. This in turn creates a set of expectations from the firm's surroundings and creates justification for different kinds of support. As such perhaps we can add another layer to the onion, one of rhetoric.

This way of thinking could be applied to DHIS 2 too. As we have seen, DHIS 2 development has several qualities that differ from "traditional" FLOSS projects. Still, the project managers and other internal stakeholders continue to define it as a FLOSS project. The rhetoric of being FLOSS gains the project some advantages for example access to software under special licenses and access markets, for example countries and regions which require FLOSS software.

Another issue is with recruitment. One of the reason for defining the project as FLOSS was to attract developers from outside HISP. This has only been moderately successful, perhaps because the rhetoric hasn't been followed up with concrete action. The project has not been explicitly marketed towards FLOSS communities and recruitment has been limited to hiring developers in Vietnam and India, and recruiting from the INF5750 course. This also relates to the rhetoric of FLOSS, however: DHIS 2 is used as a project in this course because it is FLOSS. Several students note that the possibility to work on a real FLOSS project is valuable in this regard. In theory at least, the FLOSS rhetoric thus gains the project access to development resources.

6.2.3 Division of labor in DHIS 2

In keeping with Weber's description of a BSD style project, the DHIS 2 project definitively has a close knit team of developers. In 2006, the managers decided to define a group of core developers, which were responsible for the core part of the system. The members of that group were selected based on activity and experience. All members were from Oslo, all had been involved since the start of the project and had contributed large amounts of code. Furthermore, DHIS 2 is also licensed as BSD, which would seem to underline the point.

The DHIS 2 project has a peculiar pool of developers, drawing on students from a University course in addition to a few employees and core developers. Weber argues that a FLOSS project is defined by people selecting their tasks voluntarily (Weber 2004:62). As we have seen, this is only partially true in this case. The INF5750 students must perform some work on DHIS as part of their course work. The projects they work on are determined by which projects the course managers define, how many students are interested in the different projects and a certain amount of randomness. Furthermore, the entire project is under a form of control, where there are two project managers which define tasks. This is further grounded in feature requests from a user base and a set of requirements that applied to DHIS 1.x. The tasks the different developers take on are determined by a negotiation process where the managers ask different developers to work on different tasks, usually with a positive response. Additionally, some developers work in their spare time, writing more general code, which is more similar to what Weber envisions. In many ways then, the DHIS 2 project is an atypical FLOSS project. Similarly we could expect to find other types of solutions for electronic communications than in more classic FLOSS projects. On the other hand, it may be that Weber's usage of voluntary selection of tasks as a prime identifier of FLOSS projects may not actually be all that typical. In projects with a clear and demanding user base, where developers are not just scratching an itch, there is a stronger need for coordination and prioritization. The kind of negotiation going on in DHIS 2 is indicative of tasks not being totally voluntary, that there is a sense of urgency that compels the different developers to take on tasks that limits an otherwise complete freedom of choice.

Recruiting from a community of FLOSS developers is problematic, however. In projects like the Mozilla web browser or the Linux kernel, the developers are scratching an itch, they are user-developers, producing software they themselves use on a daily basis. The software itself is what Braa et al (1995:3) refer to as context-free, which means its not related to a specific domain of knowledge. DHIS 2, on the other, is deeply embedded in a specific domain, public health. DHIS 1.x has been developed in close collaboration with users and the requirements are drawn from their experiences. External FLOSS developers will not readily have access to this knowledge, which means they can't easily pick up tasks in DHIS 2 which are not of a purely technical nature.

6.2.4 FLOSS communication in DHIS 2

As we have seen, DHIS 2 is legally FLOSS through its license. FLOSS ideology is not very strong in the project, although the rhetoric of FLOSS is important. The division of labor does not conform directly to Weber's (2004) view of the central aspect of FLOSS projects: Voluntary selection of tasks.

But what remain is the communication structure and the choice of tools which are very similar to that of most FLOSS projects: A public mailing list, a wiki, an issue tracker and a source code repository. Several developers pull for discussions to be public on the mailing list. We can say DHIS 2 is a hybrid between a FLOSS project and more classical software development carried out by an organization. Alternatively, it's a software development project which employs a FLOSS communication model.

6.3 Communication models in Global Software Development

In this section I discuss the various communication models seen from the case and what advantages and disadvantages there there to them. I first look at a FLOSS model based on the FLOSS litterature and findings from the case. I then proceed a network model which is drawn from the case and the HISP network. The development process is also viewed through the lens of distributed communities of practice, which leads to a discussion of a composite communication model.

6.3.1 The FLOSS communication model

The FLOSS communication model can be seen as a community model. The participants communicate in a global all-to-all structure. The communication happens through a set of electronic tools like mailing lists, source code management systems and issue trackers. The model has some benefits in the sense that it produces what Scacchi referred to as informalisms which effectively form the participants' or the organization's collective memory.

FLOSS projects are typically the result of scratching one's own itch, where the participants are user-developers. But as the case has shown, the FLOSS communication model can also be applied to a project with more stable organizational frames and where the developers are not users of the software they produce. Actually, in the case, many of the developers are far from the operational realities of the domain of the software: Public health care in developing countries.

In the case, the FLOSS communication model was selected as a strategic way of building globally distributed teams. The idea was that the open tools would help foster electronic collaboration between participants from all over the world. All developers would have access to resources in the project. The public communication would also ensure that information about the development would be shared by all participants, and codified in the informalisms: The wiki pages, the mailing list archives, the commit logs and so on.

The case shows, however, that this is not without problems. First of all, the strong focus on electronic and internet based tools requires a minimum of access to infrastructure: This includes stable and adequate power networks, internet access and sufficient bandwidth and system configuration to allow the different forms of internet traffic. In the case, where members from developing countries participate, this is problematic: In the area where the HISP India head office is located, you have to assume at least 1 ½ hours of power outage each day. The network used by the Ethiopian team blocks SSH access, which limits the use of SCM tools. The internet lines in Oslo have very high bandwidth, connected directly to the network of the University of Oslo. In Vietnam, Ethiopia and India the net is slower, making tasks like checking out code or searching for information online unbearably slow.

Second, a FLOSS communication model requires that each participant builds up a competency for using the various tools and internalize the practices associated with using them. This is further complicated by social aspects. Some developers may not be comfortable communicating over a public mailing list to several other participants he or she does not know.

As seen in the case another problem is the volume of (i)relevant information going on public lists. Not all participants are interested in all aspects of the development. For some it may be overwhelming to see all the discussions on a public mailing list.

Another issue is ownership to the communication model. In the case, the Norwegian developers were strongly socialized into the usage of the various electronic tools and the norms for using them. The tools themselves had been selected and set up by the Oslo team. This may have resulted in an initial bias in the usage and ownership of the tools, and may also explain partly why the Norwegians are more active on the list and in the repository.

There is a question if there is a different view of FLOSS, and thus to FLOSS communication in the North and in the South, which may explain some of the differences. I have too little data to say much about this, however. Ghosh et al (2002) point out that FLOSS projects are more common in the North, which may have bearing on this.

In terms of Sahay and Sarker's framework we can say that virtual presence in using the FLOSS model is quite high. All members of the team are visible through direct communication on mailing lists and the like. The informalisms of the project make this explicit. In its original form, the model assumes an identity bound together by the underlying motivations of the FLOSS community, or through scratching an itch. As a result, communication and social responsiveness is mutual. However, the notion of shared goals and identity can be problematized: As coordination is loose and public discussion is encouraged, many conflicting views may arise. Issues are resolved through discussion, or, in the case of the meritocracy, resolved through a vote. In the case of the benevolent dictator, this person has final decision authority. If anyone disagrees with these decisions, however, they are free to create a fork of the program and release a competing version of it.

If we apply the model to situations that differ from a classical FLOSS project, the issues may play out differently. I have explored a situation where multiple teams within a loosely structured organization are involved in the development. Furthermore, the case is complicated by the attempt at distributing development between developed and developing countries. Some teams were focused on local requirements, indicating that goals were not shared and identity was not strongly connected to the project as a whole. As a result, communication is more bilateral.

6.3.2 The network communication model

Some participants suggested an alternative model of communication. Instead of everyone communicating publicly, communication between the teams would be handled a few individuals, and each team would have a contact person in the Oslo team. This person would be responsible for getting, understanding and translating requirements and issues and communicate them to the Oslo team. This particular requirement is specific to the case, so let's generalize it. Instead of everyone communicating with everyone, the teams communicate internally and to other teams through some representative. At first glance this is similar to what was described by Sahay & Sarker. The challenge is reaching a Collaboration phase. However, the idea of a person responsible for communication between specific teams would be a specialization of team communication. Following Wenger et al (2005), we can refer to such a person as a coordinator.

The advantage to such a model is described in the case: The participants are comfortable communicating with the coordinator to whom they already have an established relationship. This person can also ensure that questions are not forgotten and actually followed up by the receiving team. The coordinator will also be well informed of what goes on the node in question. Each team member does not need to learn how to communicate with the other team, which includes learning the team's communication tools and practices, resolve language issues, gaps of understanding and so on. This is taken care of by the coordinator.

One of the challenges of this model is the fact that information has to move through an extra link to get to its destination. The first team, probably through some representative, has to explain the issue to the coordinator, who must then understand and explain it further to the members of the receiving team. With the language issues and lack of context in the transfer from one country to another, things may get lost in translation. Another issue is that of work load. One person effectively becomes responsible for managing a potentially large information flow.

Furthermore, as shown by Imsland and Sahay (2005), the communication may be inefficient. As teams communicate by proxy, the developers can't readily learn from each other and understand each other's contexts. An extension of this problem is the fact that communication is a private affair between the teams and the coordinator. The knowledge being transferred is not readily available to others in any sort of formal archive. If the coordinator is also put in charge of documentation, then the workload becomes even greater.

A generalization of this model is what we see in the HISP network: Instead of communication through special coordinators, participants have connections through many paths in the network. Recipients of information are added by the author based on private conceptualizations of who the information applies to. In the case we can say that the network is reconfigured every time a message goes through it. Participants construct it from a pool of potential recipients. The model is summarized in the following figure:

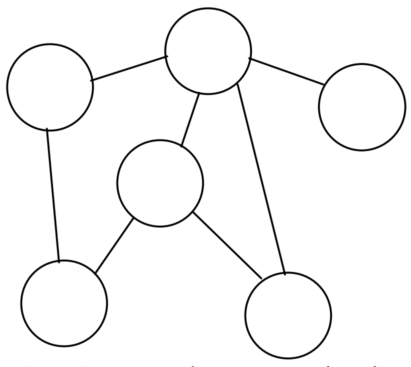


Figure 6.x: A network communication model

6.3.3 DHIS 2 development as a community of practice

If we view DHIS 2 as a community of practice (CoP), certain things come into view. The CoP has a center and a periphery. The center is made up of those participants who participate fully in the CoP, having internalized its practices and norms. Around them there is a periphery of participants who may be moving in towards or out of the center. If we look at DHIS 2 as a CoP and define producing code and using the tools as the core activity, it's clear that the Oslo team and the Norwegians represent the center. These developers have internalized the communication practices in the project and are the most active contributors to the code and in using the tools. Developers from other nodes are not as active and rarely contribute code to the core of the system.

The literature on CoP introduces the concept of legitimate peripheral participation. The developers from the other nodes are potential full participants in the DHIS 2 practice. We can focus on the idea of learning trajectories, on how the peripheral participants do and can move into the center. As mentioned, I will take a more instrumental view of learning trajectories than Wenger intended: We can view them as institutional paths the participants can take. With regard to DHIS 2 development, we can see some potential trajectories: One goes through the INF5750 course, which teaches both the tools and technologies used in the development, but also the norms for how to develop software DHIS 2 style. Another possible trajectory for learning is represented by the wiki, which has pages describing how participants can get started as DHIS 2 developers. Finally, there is the capacity building activities, where Norwegian students travel to the other nodes and basically teach people how to develop on DHIS 2 through pair programming and face-to-face collaboration.

The INF5750 course represents perhaps the most clear learning trajectory into the community. In the course one learns the basic tools and technologies used in DHIS 2 development, but also a lot of the rules and expectations that come with them, and how to use the collaborative tools. As such, Norwegian students will often be a part of the periphery of the CoP, some of them following the trajectory further inward to a status of developer, most falling off when the course ends. Those who are inside the center, the core developers, have all been connected to this course. On the other hand, some of the developers from other nodes have taken this course as well, including one developer from Vietnam and some developers from Ethiopia. They have not become "full" participants as a result to this, which may mean that the trajectory is not effective enough, or doesn't work on its own. One criticism may be that it focuses on the tools and technologies, not so much on DHIS 2 itself.

Another trajectory is the wiki, which has a link "Getting started as a DHIS 2 developer". This points to information on how to subscribe to the mailing lists, how to set up the development tools used by the developers and to general documentation for the system.

Finally, developers from Norway have traveled to the different nodes and assisted in coursing of the local developers. This means that the participants can talk to each other directly and have a dialog about problems.

But all in all, this perspective means viewing DHIS 2 as one unified CoP, of which certain individual members are in the center and some are in the periphery. This monolithic view of the development may be appropriate for many FLOSS projects, with several individual developers spread out across different locations. The resulting communication structure, an open all-to-all mailing list for example, is thus perhaps well suited. There are often no teams and no specific organizations involved. But is DHIS 2 development one large CoP, with a series of individual members? As the national teams are so prominent, this may not be an appropriate view. The teams have their own practices, local problems and different ways of communicating. Furthermore, DHIS 2 development is tightly tied to HISP as an organization. Instead, we DHIS 2 conforms more to a distributed CoP containing many cells, each with their local practices and activities.

One such cell would in this view be HISP India which is a formally registered organization with a leadership and formal structure. The team is large, with members distributed all over the country. HISP India also employs several developers. The team has its own communication structure, where the perhaps the most interesting one is an internal mailing list. HISP Vietnam also has employees who have worked closely together.

Wenger et al (2002) focus on the role of the community coordinator to manage communication in the network. This person will point people in the right direction, introduce people to each other and sometimes introduce subjects for discussion to keep the community active. In DHIS 2 development, there are people with the title coordinator, but these do not have an explicit role with regard to the global community. They are coordinators for their local organizations.

The HISP India coordinator has functioned as a main contact point between India and Norway, being one of the most active participants on the dev list. He reports feedback and bugs back to the development team. One of the Norwegian coordinators has had a similar role towards India, keeping in touch with them and bringing their requirements to the attention of the project. Thus, even though the community is distributed, the Norwegian team remains its core. After all, they still contribute most of the code to the project and represent the bulk of participation in the various collaborative forums.

If we look at distributed CoPs, Wenger et al (ibid) point to the following points as important:

1. Achieve stakeholder alignment.
2. Create a structure that promotes both local variations and global connections.
3. Build a rhythm strong enough to maintain community visibility.
4. Develop the private space of the community more systematically.

With regard to promoting local variations and global connections, the FLOSS communication model may fall short: It focuses on public all-to-all communication and has no strong conceptualization of a local team. Rhythm and community visibility is on the other hand good, as discussed with the virtual presence put forward by Sahay and Sarker (2005). In the case we saw that this was not as high in the networked model. In the case, members of the HISP network complained that people in the network weren't aware of what was going on elsewhere. The private space in the case is created by initial face-to-face meetings and then maintained through IM. This would be one way of building such a space in other situations. However, Imsland and Sahay (2005) and the case has shown that this is not unproblematic. In the case, stakeholder alignment for how to communicate was not built up: The Oslo team selected the communication tools and practices. The Vietnamese and India nodes contributed mainly to local modules rather than to the core of the system. By involving more of the participants or nodes in the selection of tools and frameworks earlier in the process, the problem may be reduced, increasing the chance of a higher degree of what Sahay and Sarker (2005) referred to both as task ability and task focus.

In many ways the idea of a distributed community of practice represents an alternative view on the FLOSS model for communication. Instead of enforcing an all-to-all communication, distributed CoPs thrive on one-to-one and many-to-many connections. Wenger et al sees this as crucial to develop personal relationships between the participants, helping to iron out cultural differences and reduce problems in trying to get people to *"let their hair down"* (Wenger et al 2005:121) in front of all the other participants. In this model, then, the network model exists alongside the community model. There is an open forum which everyone can see, but there is also a wealth of communication going on between participants directly, as well as in local sub communities.

The challenge lies in fostering effective communication and translation between the different communities. An overall problem with using a model of distributed communities is that there is no guarantee that information will be accessible to all. There will be considerable learning of tacit knowledge going on between participants, but this information is rarely made explicit anywhere. The FLOSS community model makes knowledge explicit through producing informalisms which are public (Scacchi 2006). Fogel's (2005) advice is to keep discussions public. In an optimal setting, these discussions are stored in archives for future retrieval. On the other hand, archived mails are not material that is easily available to new developers. There is a need to compact, translate and clean up such material and transform it into a more easily accessible form. The FLOSS community model does not ensure this more than the CoP model does.

6.3.4 A composite communication model

Based on the idea of a distributed CoP and how it can be modeled, we can also think in terms of a composite communication model. Instead of saying a project needs to choose one communication model, many kinds of communication models can coexist. In the case of DHIS 2, the dev-list and the FLOSS-like communication model can be kept as one avenue of communication along side other more network-like or hierarchical models, but as seen from Wenger et al (2005), more time will have to be spent thinking systematically about the structure. More specifically about how to make sure information flows between the different parts of the project using different communication models, how to document and inform participants about decisions, and so on. As suggested by Wenger et al, using coordinators may be one way of achieving this. A similar idea was put forward by some of the informants in the case. The private space between participants would need to be developed more systematically to make people comfortable. The FLOSS communication model could still be available to most or all participants in the project, giving them a channel, a tool for communication with all the other participants. An example composite model is shown in the following model:

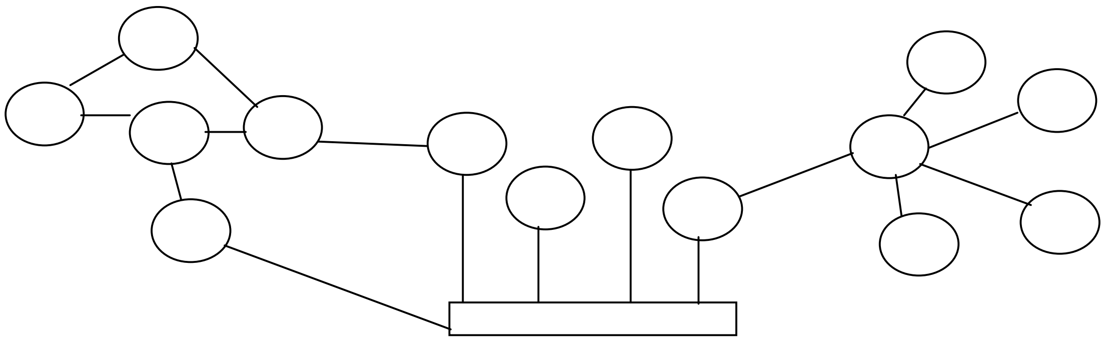


Figure 6.x: A composite communication model

In terms of research this model affords two things: The first thing is to draw our attention to the points where different communication models interface or overlap, and the potential conflicts or translation issues that might occur there. Following Wenger et al (2005) one solution is a coordinator to perform this translation job. The second is to allow us to view a globally distributed development project, not as one unit, but as potentially containing several subunits with different communication models. Through this view we can "zoom" in and out, depending on the unit of analysis, and still be aware of where and how this unit interfaces with others.

The model need not apply simply to a geographical separation, but can denote different forms of communication taking place during different phases of a decision making process. Initial discussions can happen in private spaces first, then move on to a public mailing list. We can also perceive it as segmenting communication about different subjects. Strategy can be discussed in a network model, involving relevant participants, coding coordination can occur using a community communication model and implementation efforts can be directed through a hierarchical model. A community can provide a global meeting place, while the other structures handle day-to-day coordination in a project.

6.4 Implications from the case on Global Software Development: Reducing gaps of understanding

In the case we saw how language and the "gaps of understanding" represented a problem. One such gap was the language used in the project. Projects pay extra attention too situations where the participants are not communication in their first language. This is consistent with findings by Wenger et al (2004:119): If a participant is unsure about the language, he or she is much less likely to participate actively.

The use of textual communication also made things more difficult: It is difficult to explain a problem in a second language and without the recipient either knowing about the context or seeing the problem. This points to a contextual gap, not specific to the knowledge domain of the software development project, but with everyday problems and situations in the development. This is slightly different from what Braa et al (2007) refer to as knowledge incongruence. More rich media like remote login and video could help in this regard. Ultimately, being co-present and seeing and discussing problems is more effective in solving them.

Furthermore, this problem was aggrevated by technical jargon. Members of the Oslo team were too technical in their replies. The technical jargon made things more difficult to keep up with discussions unless the participant feels comfortable with the technology involved. There was a gap in knowledge about the underlying tools and frameworks between the different nodes. Another issue pointed to by the case is the importance of the codebase in the project. In the case, the codebase is effectively produced by a small group of people, and a large part by one person. The project was largely started in Oslo, and while developers from Vietnam participated relatively early, most of the core development activity was done by Norwegians. The tools and frameworks were also chosen from Norway. As such, the Norwegians had a head start on learning and using the frameworks, causing a technical gap of understanding or incongruence between the teams from the beginning. The software only got more complex, and was changed often, which apparently widened the gap.

Capacity building is also one way of reducing the problem. One informant pointed to weak capacity building as one of the potential causes for problems. But what sort of capacity building is most effective has not been studied in depth. In Vietnam, the Norwegian masters students gave general lectures and gave programming assignments. In Ethiopia, Nguyen (2007) asked the students to do practical things with PHP code before teaching them the more abstract concepts of programming in the language. The INF5750 course itself uses a combinations of lectures and project work with guidance by a teaching assistant. As shown by Øverland (2006), different countries have different educational systems, which result in people being more receptive to different kinds of learning.

It’s interesting to note that the Vietnamese team fell apart due to completely other reasons than technical skills. Up until the collapse, the team had started to produce more core related code. Based on that special situation, it’s difficult to ascertain the effectiveness of the local capacity building and which types work. We can note however, that despite the fact that several participants have taken the INF5750 course in Norway, it has not led to increased activity on the codebase.

Another interesting finding is how the mails sent by the SCM system and later from the ticket system starts discussions about the code. These represent hooks or triggers for communication: A change in the code or the state of the project is made explicit, including an explanatory comment and communicated to all the other participants. Such hooks and triggers may me explored futher.

Concluding remarks

For this thesis, I chose the following research objective:

> **Research objective:** Explore how different communication models may be constructed and their effects on global software development projects.

This was concretized into two research questions:

> **Primary research question:** How does a FLOSS communication model affect global software development projects and how does this relate to other communication models?
>
> **Secondary research question:** How are electronic communication tools used in globally distributed development.

Through the work thesis I developed the concept of communication models to analyze data from globally distributed software development projects. A communication model encompasses the pathways through which the participants are communicating, the norms which affect the communication and the electronic tools used for the communication.

Four such communication models were explored: A hierarchical model, a network model, a community model based on Free/Libre Open Source Software development (FLOSS), and a composite model, based on findings from the case and the literature around distributed communities of practice (CoP). The composite communication model was suggested as a way of deconstructing the communication pattern of an organization with different teams which may have different internal communication structures.

The FLOSS model is effective in terms of helping to keep discussions public and everyone informed about what is going on in the project. As seen in the case, the team which embraced this model communicates extensively through the various electronic tools. But the model has issues: The underlying physical infrastructure must be strong enough to allow effective use of tools, meaning stable internet connections and power supplies. This is especially relevant for development involving participants from developing countries. The model may also be ineffective if the development involves different teams with a strong internal identity. The FLOSS model has a focus on individual participants talking to a community of developers, rather than as members of teams with their own priorities and activities. Finally, communicating through public media to several participants may have a social cost: Not everyone is comfortable talking to a large group of people.

The case was used to give a rich description of how communication tools are and aren't used in one globally distributed software development project: DHIS 2. It is not possible to generalize in a positivist way how communication is and will be in similar projects. Rather, the findings can be used in comparisons with other projects in future research.

The case shows that it would be useful to do more research into how different types of tools affect development and are suited for different kinds of projects. Do you choose individual mails, a mailing list, a web forum or a combination of those? Are other tools more suited for the job? Comparative research on the matter may be fruitful. The research from the FLOSS field shows that the use of tools such as mailing lists and issue trackers are in abundant use, and in many ways make distributed FLOSS projects possible. The situation may be different for development projects within other organizational contexts, such as firms or non-profit organizations. The case of development in developing countries, with potentially unstable lines and low bandwidth also affect the choice of tools.

Another potentially fruitful avenue of research would be on hooks and triggers for communication. In the case, the SCM system sent mails onto a mailing list which sparked discussion. It would be interesting to identify more such potential triggers and how they can be utilized in a globally distributed project.

From the material several *gaps of understanding* were identified, among them the problem with communicating in a second language. The case also points to a technical gap of understanding, caused by the codebase itself. The complexity of the code, the various tools and frameworks involved and the frequent changes in the system made by a small group of people, made it difficult for other developers to keep up. This point to the need to establish a better baseline and collaboration from the start in such distributed projects. It would be useful to look more closely at the effects of the complexity and revisions of the codebase on the communication in a distributed development project. A contextual gap was also seen, where mails could not convey the fullness of a problem encountered.

Parts of the data from the case are based on activity on the mailing list and in the source code repository over a period of two years. Several incidents occurred during the spring and summer of 2007, such as the introduction of a new issue tracking system and code browsing tools, an IRC channel and a few confrontations between the teams regarding the direction of the project and how to coordinate. It would be interesting to extend the data to include this period, to see if this had any effect on the communication pattern.

Furthermore, I have focused on the official channels for communication in the project, rather than on tools used in the local nodes. Looking more closely at communication inside the India node, and expanding on the experiences covered by Nguyen (2007) in Ethiopia would bring more interesting and potentially more diverse views on the case.

The study is rather limited in terms of the effect of different cultures on how communication is done. Studies from other research fields such as Anthropology and Culture and Communication may serve as enlightening in this regard. Furthermore, different cultures' relationship to FLOSS development in general, in order to further illustrate which issues which may arise in projects using a FLOSS-like communication structure. More in-depth study can also be done on the identity the participants construct and what affects this: Whether they feel they belong to individual teams or to a project as a whole or both. Similarly, the field of Computer Supported Cooperative Work (CSCW) has studied distributed work mediated by electronic communication, although not on such an international scale as Global Software Development. It may be interesting to combine findings from these two fields to help build a better understanding of globally distributed development.

References

- Alvesson, M. (2004): Knowledge Work and Knowledge-intensive Firms. Oxford University Press. Oxford.
- Baskerville , R. L. & Wood-Harper, A. T. (1996): A critical perspective on action research as a method for information systems research. In: Journal of Information Technology
- Braa, J., Monteiro, E., Sahay, S. (2004): Networks of Action: Sustainable health information systems across developing countries, In: MIS Quarterly Vol 28 No. 3
- Braa, J., Monteiro, E., Reinert, E. S. (1995): Technology transfer vs technological learning: IT-infrastructure and health care in developing countries. In: Technology for Development vol 6 (1), IOS Press
- Braa, J. et al (2007, submitted for publication): Openshoring in south-south-north networks: a knowledge and development perspective. In: MIS Quarterly.
- Feller, J. & Fitzgerald, B. (2000): A framework analysis of the open source software development paradigm. In: Proceedings of International Conference on Information Systems
- Fogel, K. (2005): Producing Open Source Software: How to Run a Successful Free Software Project. URL: http://producingoss.com/ [Read: 2007–04-15]
- Fitzgerald, B. (2006): The Transformation of Open Source Software. In: MISQ
- German, D. M. (2003): The GNOME Project: a Case Study of Open Source, Global Software Development. In: Software Process Improvement and Practice 2003, Issue 8.
- Gjerull, N. F. (2006): Open Source Software in Developing Countries: The HISP Case in Ethiopia. Masters thesis, University of Oslo
- Ghosh, R. A., Glott, R., Krieger, B., Robles, G., (2002): Free/Libre and Open Source Software: Survey and Study. FLOSS. Deliverable D18: Final Report, Part IV: Survey of Developers.
- Hersleb, J. D. & Moitra, D. (2001): Global Software Development. In: IEEE software, March/April Issue.
- HISP.info (2006): Mailing lists. http://www.hisp.info/confluence/display/DOC/Mailing+lists [Read: 2006–01-07]
- Imsland, V & Sahay, S. (2005): Negotiating Knowledge: The Case of a Russian–Norwegian Software Outsourcing Project Scandinavian Journal of Information Systems, Volume 17
- Kalleberg, R. (1997): Methodological Issues in Comparative Social Science. In: Mjøset L, Engelstad F, Brochmann G, Kalleberg R, Leira A, editors. Jai Press, Inc.
- Lanzara G. F., Morner, M. (2003): The Knowledge Ecology of Open Source Software Projects. 19th EGOS Colloquium, Copenhagen
- Lave, J. & Wenger, E. (1991): Situated Learning: legitimate peripheral participation. Cambridge: Cambridge University Press.
- Miller, J. A. (2004): Promoting Computer Literacy Through Programming Python. Ph.D. dissertation, University of Michigan URL: http://www.python.org/files/miller-dissertation.pdf. [Read: 2007-054-14]
- Mockus et. al. (2002); Two Case Studies of Open Source Software Development: Apache and Mozilla, ACM Transactions on Software Engineering and Methodology, 11 (3), 309-346.

- Nguyen, Thanh Ngoc (2007): OSS for health care in developing countries: The case study of DHIS2 and patient based system in Ethiopia and Vietnam Masters thesis, University of Oslo
- Nordal, K. (2006) The Challenge of Being Open – Building an Open Source Development network. Masters thesis, University of Oslo.
- Raymond, E. S. (2000): The Cathedral and the Bazaar. URL: http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ [Read: 2007–04-13]
- Sahay S. (2003): Global Software Alliances: The challenge of 'standardization'. In: Scandinavian Journal of Information Systems, Issue 15.
- Scacchi, W. (2004): Free and Open Source Development Practices in the Game Community. In: IEEE Software, Jan-Feb Issue.
- Scacchi, W. (2006): Free/Open Source Software Development: Recent Research Results and Methods. In: M.V. Zelkowitz (ed.), Advances in Computers, Vol. 69, 2007.
- Schmidt, K. & Bannon, L. (1992): Taking CSCW Seriously: Supporting Articulation work. In: Computer Supported Cooperative Work Journal, vol. 1, no. 1
- Shaikh, M. & Cornford, T. (2005): Learning/organizing in Linux: a study of the 'spaces in between'. In: Proceedings of the fifth workshop on Open source software engineering
- Walsham, G. (1995): Intepretative case studies in IS research: nature and method. In: European Journal of Information Systems Issue 4.
- Weber, S. (2004): The Success of Open Source. Cambridge, Massachusetts, London, England: Harvard University Press.
- Weber, M. (1978): Economy and Society: an outline of interpretive sociology. Berkeley: University of California Press
- Wenger, E. (1998): Communities of practice: Learning, Meaning and Identity. Cambridge: Cambridge University Press
- Wenger, E., McDermott, R., Snyder, W. M. (2002): Cultivating Communities of Practice. Boston: Harvard Business School Press
- Wikipedia 2007: Pareto Principle. URL: http://en.wikipedia.org/wiki/Pareto_principle [Read: 2007-07-15]
- Øverland, L. (2006): Global software development and local capacity building: A means for improving sustainability in information systems implementations. Masters thesis, Univerity of Oslo

Appendix: Scripts for data processing

The following scripts where used to process and order data from the repository and mailing list archives.

```
#! /usr/bin/python

"""
------------------------------------------------------------------------
r2647 | hanssto | 2007-01-03 22:48:08 +0100 (Wed, 03 Jan 2007) | 5 lines
"""

import re
import sys

log = ''

if len(sys.argv) > 1:
  source = open(sys.argv[1])
else: #use stdin
  source = sys.stdin

log = source.read().strip()

boundrary = '------------------------------------------------------------------------\n'
headerRe = re.compile('r(\d*) \| (.*?) \| (\d{4}-\d{2}-\d{2} \d{2}:\d{2}:\d{2})')

entries = log.split(boundrary)

for entry in entries[1:]:
  print '%s\t%s\t%s'%headerRe.match(entry).groups()


#! /usr/bin/python

import sys

log = ''

if len(sys.argv) > 1:
  source = open(sys.argv[1])
else: #use stdin
  source = sys.stdin

log = source.read().strip()

entries = log.split('\n')

for entry in entries:
  tokens = entry.split('\t')
  print "insert into commit values ( %s,'%s','%s');"%(tokens[0],tokens[1],tokens[2])

#! /usr/bin/python

"""
archives.zip: 27. sept 2005-28. sept 2006
13 Mar 2005-9 Jan 2007

<li><em>From</em>: <a href="mailto:postmaster%40dhis.hisp.info">address@hidden</a></li>
<li><em>Subject</em>: TEST</li>
<li><em>Date</em>: Tue, 27 Sep 2005 15:12:13 +0200 (CEST)</li>

for alle html-filer:
  Les from, subject, date
  Skriv til fil
"""
import glob, re, time, datetime, sys

msgfiles = glob.glob('msg*.html')
outfile = open('summary', 'w')
filedata = ''
fromre = re.compile(r'<li><em>From</em>:.*?mailto:(.*?)".*?</li>')
subjre = re.compile(r'<li><em>Subject</em>: (.*?)</li>')
datere = re.compile(r'<li><em>Date</em>: (.*?)</li>')
fail = 0
```

```
faileddata = ""
result = ""

for file in msgfiles:
  filedata = open(file).read()
  match = fromre.search(filedata)
  if not match:
    fail = fail + 1
    faileddata += "!-!-!-\n" + filedata
    continue
  mail = match.group(1)
  mail = mail.replace('%40', '@')
  mail = mail.strip()
  match = subjre.search(filedata)
  if not match:
    fail = fail + 1
    faileddata += "!-!-!-\n" + filedata
    continue
  subj = match.group(1)
  match = datere.search(filedata)
  if not match:
    fail = fail + 1
    faileddata += "!-!-!-\n" + filedata
    continue

  date = match.group(1)

  #Sat, 6 Jan 2007 13:54:06 +0100 (CET)

  date = re.sub( '[+-]\d\d\d\d', '', date.strip() )
  date = re.sub( '\(.*?\)', '', date.strip() )
  date = date.strip()
  #print date
  try:
    date = time.strptime(date)
  except:
    try:
      date = time.strptime(date, '%a, %d %b %Y %H:%M:%S')
    except:
      try:
        date = time.strptime(date, '%d  %b %Y %H:%M:%S')
      except:
        print "fail!"
        sys.exit()
  date = time.strftime( '%Y-%m-%d %H:%M:%S', date )

  result += "%s\t%s\t%s\t%s\n"%(date,mail,subj,file)


print "Fail %d"%fail
open('result.txt', 'w').write(result)

#! /usr/bin/python

import sys

if len(sys.argv) < 2:
  sys.exit(1)

file = sys.argv[1]

data = open(file).read().strip()
data = data.replace("'", "\\'")
lines = data.split('\n')

res = ''

for line in lines:
  tokens = line.split('\t')
  try:
    res +="insert into mail values ('%s','%s','%s','%s');\n"%(tokens[3],tokens[1], tokens[2],
tokens[0])
  except: print line

open('insert.sql', 'w').write(res)
```

```
#! /usr/bin/python
"""
Examples:
TESSST, PRECIOUSSS
Re: [Dev] TESSST, PRECIOUSSS
RE: [Dev] TESSST, PRECIOUSSS
Re: [dev] Problemer med maven jetty:run-war
RE: [dev] Sorting in Core
[SVN] r42 - in dhis-2/dhis-services:
[dev] Periods
[DEV] Transfer keys and values as hashtable in webpage
Re: [dev] Re: m1 compile errors / was: [dev] request for bridge testing
OT: Summer of Code and Thiruvananthapuram decleration
Re: M1
Re: Database Maintenance
Re: [Dhis14] DHIS 2.0 Milestone 1 released

So... Absolutely all variants of "re: [dev]" are most assuredly replies to the list.
All mails that do not start with "re:" are most assuredly thread starters. Including those that
start with "[SVN]" and "[DEV]".
Some are re's to mails from other lists, alà "re: [scm]"
Some mails are sent to multiple lists, or multiple recipients, which makes it somewhat arbitrary
what the subject becomes.
There are examples (at least one) of subjects that are completely equal when compared lower case
(Indicators vs indicators).
A mail can only belong to one thread. Once it is assigned, it cannot be assigned again.

There -will- be errors in the data...
"""

import re
import sys

if len(sys.argv) > 1:
  source = open(sys.argv[1])
else: #use stdin
  source = sys.stdin

maildata = source.read().strip().split('\n')

mails = []
threads = {}
replies = []
starters = []
res = []
secres = []
unsure = []

def checkSubject( subj, prelen ):
  for testsub in threads:
    if subj == testsub or subj[prelen:] == testsub[prelen:]:
      return testsub
  return None

def normalize( str ):
  return re.sub( '\s+', ' ', str )

for mail in maildata:
  mails.append( mail.split('\t') )

for mail in mails:
  subject = normalize(mail[2])
  subj_low = subject.lower()

  if subj_low.startswith('re: [dev] '): #Definetly a reply...
    #Do a preliminary sorting:
    if threads.has_key(subject[10:]):
      #sys.stderr.write('sorted\n')
      threads[subject[10:]].append(mail[3])
    else:
      replies.append(mail) #Sort later
    continue

  if not subj_low.startswith('re:'): #Definetly a starter
    starters.append(mail)
    threads[subject] = [mail[3]]
    #print('starter: ' + str(mail)) ###
    continue

  #Reply from the [scm] list sent to the dev list, most likely a starter
```

```
    if subj_low.startswith('re: [scm]'):
      if threads.has_key(subject[10:]): #Is this actually a reply to a topic?
        #sys.stderr.write('sorted scm\n')
        threads[subject[10:]].append(mail[3])
      else: #It's own starter
        starters.append(mail)
        threads[subject] = [mail[3]]
        #print('starter scm: ' + str(mail)) ###
      continue

    """
    Now it get's tricky... Some "re:"'s are replies if there is a matching starter subject.
    Others are replies to mails sent off list, but are thus starters in the context of the list.
    We don't know all the starter subjects yet, so we can't be sure about them. But we can do a
    preliminary sorting.
    """
    if subj_low.startswith('re:'):
      #Do a preliminary sorting
      if threads.has_key(subject[3:]):
        #sys.stderr.write('sorted re\n')
        threads[subject[3:]].append(mail[3])
      else:
        res.append(mail) # Sort later
      continue
    """
    This entire block may be uncessesary though. A few of them (if not all) might be thread starters
    never getting registered, thus not warranting a reply. We know they're sorted by date, right.
    """

    """
    So what the heck is this??
    Hasn't happened so far...
    """
    sys.stderr.write('unsure ' + str(mail) + '\n')
    unsure.append(mail)

  for mail in res:
    subject = normalize(mail[2])

    if threads.has_key(subject[4:]): # "re: "
      #sys.stderr.write('sorted re secondary\n')
      threads[subject[4:]].append(mail[3])
      #print 'sorted re secondary'
      continue

    if threads.has_key(subject): #Existing verbatim subject?
      #sys.stderr.write('sorted equal\n')
      threads[subject].append(mail[3])
      #print 'sorted re equal'
      continue

    testsub = checkSubject(mail[2], 4) #Different case from existing subject?
    if testsub:
      threads[testsub].append(mail[3])
      #print "different case re"
    else:
      threads[subject] = [mail[3]]
      starters.append(mail)
      #print('starter re: ' + str(mail)) ###
      #sys.stderr.write(str(mail) + '\n')

  for mail in replies:
    subject = normalize(mail[2])
    if threads.has_key(subject[14:]): # "Re: [dev] Re: "
      #sys.stderr.write('extended re\n')
      threads[subject[14:]].append(mail[3])
      #print 'sorted extended devre'
      continue

    if threads.has_key(subject[10:]): # "Re: [dev] "
      #sys.stderr.write('sorted secondary\n')
      threads[subject[10:]].append(mail[3])
      #print 'sorted devre secondary'
      continue

    if threads.has_key(subject): #Existing verbatim subject?
      #sys.stderr.write('sorted equal\n')
      threads[subject].append(mail[3])
      #print 'sorted devre equal'
      continue
    testsub = checkSubject(mail[2], 4) #Different case from existing subject?
    if testsub:
      threads[testsub].append(mail[3])
      #print "different case devre"
    else:
      threads[subject] = [mail[3]]
      starters.append(mail)
      #print('starter re: ' + str(mail)) ###
      #sys.stderr.write(str(mail) + '\n')


  threadid = 0
  for subject in threads:
    mails = threads[subject]
    for mailid in mails:
      print "insert into thread values ( %d, '%s' );"%(threadid, mailid)
    threadid += 1

  #! /usr/bin/python
  """
  #2005-03
  msg00083.html  2005-03-19 22:50:21   Tag1,Tag2
  msg00087.html  2005-03-21 01:31:26   Tag1,Tag3
  msg00100.html  2005-03-21 15:25:21   Tag1
  """

  import sys

  if len(sys.argv) > 1:
    source = open(sys.argv[1])
  else: #use stdin
    source = sys.stdin

  lines = source.read()
  tokens = []
  tags = []

  for line in lines.split('\n'):
    if line[0] == '#': continue
    tokens = line.split('\t')
    #print tokens
    tags = tokens[2].split(',')
    for tag in tags:
      print "insert into tag values ( '%s', '%s' );"%( tokens[0].strip(), tag.strip())
```