NTNU
Norwegian University of
Science and Technology

Open Source, Distributed IS Development

A Study of the Development and Implementation of a Hospital Information System in India

Samson Valland
Per Øyvind Øygard

Master of Science in Computer Science
Submission date: June 2011
Supervisor: Eric Monteiro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

Open-Source software has become increasingly more common in IT-organisations. Despite this the focus of studies on open-source has largely been focused on large system software. In our thesis we have worked on a software development project in Shimla, India, to create a hospital management system for the district hospitals in the state of Himachal Pradesh. Through our studies we have looked at the challenges of developing and implementing an open-source IS system in low-resource environment. Our results show that such an undertaking can be succesful, but that distributed development poses a lot of challenges, and that the use of open-source software, while free, still necessitates a lot of work and close communication with the community.

acknowledgments

We have been fortunate to had good and helpful people around us throughout the whole process of writing this thesis both in India and Norway. We would especially like to thank:

- Eric Monteiro for firm and useful guidance
- The HISP India organisation for letting us come to India and collaborate on their project.
- The HISP India Shimla team for taking good care of us during our stays in Shimla.
- Mari Tharaldsen for proofreading
- Family and friends for moral support

Contents

List of Figures

Acronyms

DHIS District Health Information System.

EMR Electronic Medical Record.

HIS Health Information System.

HISP Health Information System Programme.

HMIS Health Management Information System.

HospIS Hospital Information System.

HospMIS Hospital Management Information System.

ICT Information and Communications Technology.

IPD In-Patient Department.

IS Information System.

IT Information Technology.

OPD Out-Patient Department.

OSS Open Source Software.

PHC Primary health care.

Chapter 1

Introduction

Open Source Software (OSS) development is a field that in the last decade has seen a number of studies. The results have by now shown that despite claims to the contrary, the OSS movement has been capable of producing software that is both successful and of a high quality. These studies are, however, in large part limited to older and larger projects, particularly looking at Apache, Linux, and similar low-level applications. While these are valuable insights, they are in many ways not comparable to the growing number of smaller OSS projects where the focus is user-level applications. Where the classical OSS projects are often driven by a few developers' need to "scratch their own itch", focusing on personal issues and pet-peeves, commercial adoption has increasingly changed this. Commercial entities are seeing the value of leveraging the OSS model both as a way to harness the work already produced by others, as well as enticing others to contribute to their own projects. Dealing with customer demands and strict deadlines in this way is quite foreign to what many would consider the classical OSS way of conducting development, and poses many problems which potentially undermine the very advantages that OSS are meant to provide.

The project we have had the opportunity to work on is the development of a Hospital Management Information System (HospMIS) in India. The system is based on an American made open-source Electronic Medical Record (EMR) system called OpenMRS. OpenMRS represents a change in the classical OSS direction, away from the low-level server application, and towards a field that has previously been dominated by a few large multi-national companies. Health informatics is an area that, with a few exceptions, has seen very little traction from the open-source world, despite a rising demand from the third world. The challenges in such an undertaking are numerous, from strict requirements in implementation and functionality, to the difficulty of cooperating with the open-source community in what is a very complex and time-restricted project. In addition to these technical challenges, the nature of working in a distributed team with relatively few resources available create a highly complex project which bears few likenesses to either classical OSS projects, or the classical cases of Health Information System (HIS) that are common in scientific studies.

Our goal in this project has been to examine the effects of this complex development environment upon the development and implementation process. We hope this can aid us in further understanding of how OSS best can be leveraged in costumer-oriented and resource-limited projects.

1.1 Research Question

Any HIS development projects presents enormous challenges in terms of implementation and standardisation, and in many ways this goes doubly so for a system developed and implemented in a third world country. The lack of resources inherent in such a project adds numerous obstacles, not least in terms of time. Experience and frameworks for development is also often poor, and local knowledge about the use of information systems lacking. Through our work our goal has been to look at the processes used in the development and implementation of the system, and try to understand the challenges and solutions involved, and how such a process could feasibly be improved in future projects.

To this end we have created four research questions which we hope will help highlight some of the most important aspects of the development:

RQ1: How is the development process of an information system affected by a distributed and low-resource environment?

RQ2: What are the biggest obstacles in implementing a successful, computerised health information system at a low-resource, paper based health facility?

RQ3: What efforts have been made to standardise the system?

RQ4: What steps could be taken to improve the development and implementation of the system?

1.2 Organisation of the report

The report is comprised of nine chapters. The first chapter, Introduction, presents the case and the research questions of the report.

Chapter two presents the theoretical framework used in the report, starting with HIS. We continue by describing distributed software development, open-source software, and finally agile development methods and software standardisation.

In chapter three we present the research setting of the project, including India, Health Information System Programme (HISP), the local team and the software used, OpenMRS. We continue in chapter four with the research and data collection methods used in the project.

Chapter five consist of the results of our study, starting with the development process of the product. We continue by describing the implementation process in chapter six, and the communication practices in the project team in chapter seven.

Finally in chapter eight we analyse our findings with regards to our research questions, and in chapter nine, we summarise and conclude our results and experiences from the project.

1.3 Contributions of the thesis

First and foremost with this thesis we seek to look at the use of an OSS project within another OSS project. This is in itself not an uncommon phenomenon, but the novelty of OpenMRS and the limitations of the HISP project make for a very complex project which is very different from most OSS projects commonly studied, as well as a field that is still poorly understood. It is our belief that this thesis can help shed some light on the obstacles involved in the development of localised OSS projects, and how development and implementation processes can be improved in future projects.

Secondly, we have looked at the interaction between the developers in the OpenMRS project and the HISP project. The size of both of these projects makes for a very interesting case that we think can help both the OSS community and other organisations making use of OSS to better understand how to efficiently communicate and cooperate.

Chapter 2

Literature

In this chapter we will describe the theory upon which we have founded our analysis. This includes health information systems, agile development methods, and distributed development.

2.1 Health Information Systems

A HIS is a system that integrates data collection, processing, reporting, and use of information necessary for improving health service effectiveness and efficiency through better management at all levels of health services[Lippeveld et al., 2000]. It is not a specific type of application, but a collection of tools, systems, users, routines, etc. that together seek to improve health services. To better understand its definition, it might help to look at the definition of Information System (IS) in general: Heeks defines IS as *"systems of human and technical components that accept, store, process, output and transmit information. They may be based on any combination of human endeavours, paper-based methods and IT"*[Heeks et al., 2002] This implies that IS does not necessarily need to involve IT at all, and when it does, IT can be presented as a component alongside humans and organisations. If you look more into the thought of Information Technology (IT) as an equal component, some state that the interplay between these technical, human and organisational factors can best be addressed by conceptualising IS as social networks. [Walsham, 2001b]

Most health information systems consist of many different systems. Some with specialised purposes, others more overlapping and aiming to combine functions and objectives. Standardisation and interconnectivity between the subsystems can often be crucial for improving health services. Figure 2.1 shows how different HIS can be classified and linked, based on their areas of usage:

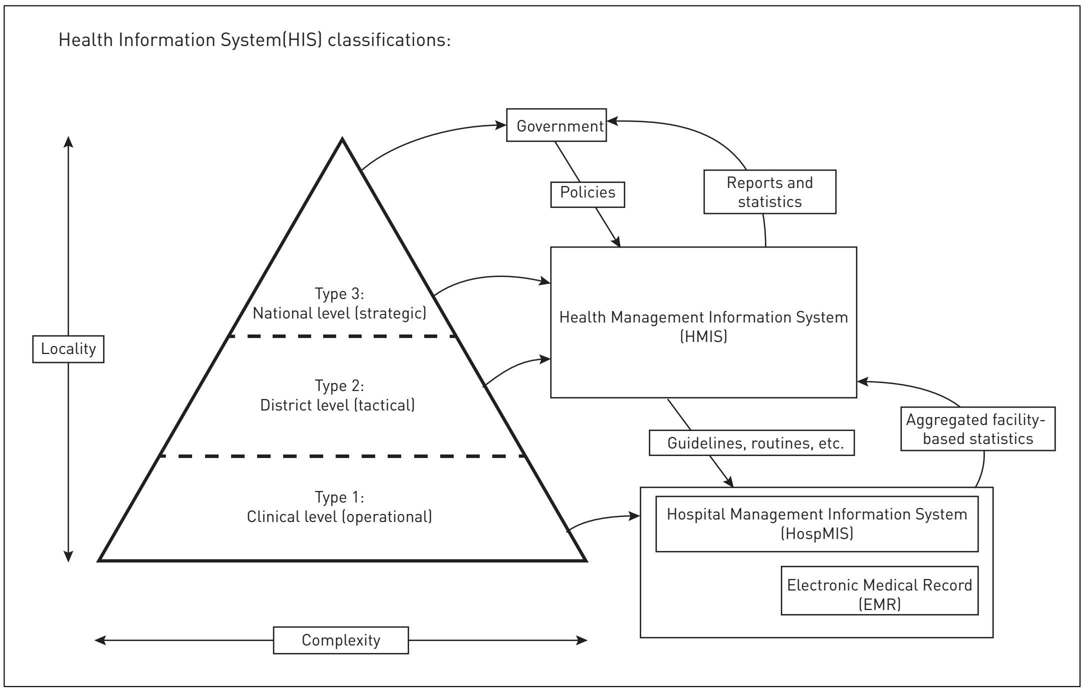

Health Information System(HIS) classifications:
Government
Policies
Reports and statistics
Locality
Type 3: National level (strategic)
Type 2: District level (tactical)
Type 1: Clinical level (operational)
Health Management Information System (HMIS)
Guidelines, routines, etc.
Aggregated facility-based statistics
Hospital Management Information System (HospMIS)
Electronic Medical Record (EMR)
Complexity

Figure 2.1: HIS-classifications

The different subsystem can be described as following:

Electronic medical record(EMR) A medical record is a systematic documentation of a patient's individual medical history and care. EMRs have existed for more than 30 years, but the majority (90%) of hospitals worldwide are still using paper-based medical records.

Health Management Information System(HMIS) These are typically aimed at aiding Primary health care (PHC) decisions and provide useful statistics based on aggregated data, for instance programme monitoring and evaluation systems for monitoring tuberculosis, maternal and child health, family planning and epidemiologies.

Electronic Health Records(EHR) systems The term electronic health records is often mixed up with EMRs, but there is a noteworthy difference – While an EMR is most often maintained and used within an institution, an EHR's aim is to gather and generate a thoroughly record of a patient's health history regardless of institutions he has been visiting. In Norway (among most western countries), there is an ongoing process to develop and implement nation-wide EHR systems through linking public hospitals' EMR systems together, but there are many challenges related to standardisation, privacy and socio-technical issues.

Hospital Information System(HospIS) As with HIS, a collective term for IS related to running hospitals. For instance, HospMIS and EMR often forms the basics hospitals' HospIS.

Hospital Management Information System(HospMIS) Systems to support tasks related to accounting, personnel, supplies, etc.

Vital registrations systems Births and deaths registration systems.

There is practically no one who questions the value of HIS in the western world, but its relevance in developing countries has been discussed. Walsham and Sahay[Walsham and Sahay, 2006] concluded that "*the debate has been resolved with a clear yes answer.*" Also, Information and Communications Technology (ICT)-initiatives are by many considered to be the most effective measure to improve the quality and efficiency in the health care sector. The Case of eHealth [Silber, 2003], cited in [Helsedepartementet, 2004], presented at the European Commision's first high-level conference on eHealth in 2003, states that "*eHealth is the single-most important revolution in health care since the advent of modern medicines, vaccines, or even public health measures like sanitation and clean water.*"

Even though ICT-initiatives in the health sector and the idea of HIS for improving the overall quality of health services has been around since the start of the IT-era, it is still a field with large potential, also for the most developed countries of the world. For instance, in England they have an ongoing project implementing a nationwide electronic health record system for secondary care. A qualitative analysis report[Lippeveld et al., 2000]concludes that *"experiences from the early implementation sites, which have received considerable attention, financial investment and support, indicate that delivering improved health care through nationwide electronic health records will be a long, complex, and iterative process requiring flexibility and local adaptability both with respect to the systems and the implementation strategy."* This is not a unique case; most western countries dedicates a large proportion of budget funding to the field. In Norway in 2008 the government spent over 217 billion NOK in total on health expenditures, of which nearly 800 million NOK were spent on IT-consultancy.[1] Norway is one of the leading countries on the use of ICTs in the health sector. It was the world's first country who got an almost complete distribution of electronic medical records among general physicians, and was also the first to implement a full patient record-system in a hospital, as early as in 1993. As of today, 100% of the public hospitals use electronic medical records. In addition many other digital information systems are common, such as tools for image analysis, diagnosis support, electronic health record systems for special care, and so forth.

[1]Source: Statistics Norway - Health Accounts, 2006-2008: `http://www.ssb.no/english/subjects/09/01/helsesat_en/`(June 11, 2011)

2.2 Distributed Software Development

Outsourcing and distributed development has in recent years become increasingly more common in the software development industry. As early as 2000 it was reported that 70% of US companies had outsourced [Carmel and Agarwal, 2000] parts of their business process. Since then the numbers have only gone up, with an increased focus on outsourcing as a cost-cutting measure [Scardino et al., 2006]. Even disregarding direct outsourcing to emerging markets, an increasingly global marketplace has forced many companies to adapt to a more distributed working process. In this chapter we will look at some of the challenges and pitfalls commonly encountered in a distributed development environment.

2.2.1 Challenges

Distributed teams are not a new idea, but the prevalence of high-speed Internet has massively increased the possibilities for remote collaborations. The Internet has also contributed to a much more global market, which in turn feeds the need for local expertise, and collaborations between different teams. While the advantages of such arrangements are many, the problems associated with distributed teams are both numerous and well documented - and if not avoided, potentially crippling for any project.

Communication is a challenge for any project. Insufficient communication can lead to both mistrust and faulty products, if not complete failure. This is, as most people can probably guess, an even bigger challenge when teams are split up and face-to-face time is reduced. Teams will often be split across continents and time-zones, necessitating asynchronous modes of communication, like e-mail. E-mail is widely used in most companies today, but problems arise when email is the main source of communication. Where you in a collocated environment can have a ten minute discussion on a subject and reach an agreement, an email discussion can in many cases take days due to lack of work-time overlap, and the relative low bandwidth of expression in email. The result is stalled processes and frustrated team-members.

In the cases where synchronous communication like teleconferencing is made possible, there are still many issues which hinder efficient knowledge-transfer. Lori Kiel [Kiel and Eng, 2003] describes teleconferencing between a Canadian and a German team, where language barriers and cultural differences cause severe problems.

> *German participants reported frustration at not being able to follow or participate in the discussion. Canadians often interpreted the silence coming from the other office as an indication that no one in Germany wanted to participate or add to the discussion, and carried forward with the meeting.*

In addition to cultural and lingual problems, teams will often be split unevenly, with management and customer-facing contacts being separated from the developers. Such a split is hard to avoid since you can't move the customer, but it nevertheless often creates an extra imbalance in the relationship between the different teams.

Knowledge sharing is a very important part of the development process. Perry[Perry et al., 2002] report that on average their developers would spend 75 minutes a day in "unplanned interpersonal interaction." There is little reason to believe this number is any different for us, as informal communication is an important way of sharing knowledge. This lack of face-to-face communication necessitates other means of communication instead. Internet Relay Chat (IRC) and Instant Messaging are common forms of remote communication, though they are limited in their ability to serve as informal communication, thus rarely as good as the real thing. In cases where you are unable to properly share knowledge, you quickly run the risk of losing tacit knowledge, which can create problems at later points.

2.3 Open Source Software

> Linux is subversive. Who would have thought even five years ago (1991) that a world-class operating system could coalesce as if by magic out of part-time hacking by several thousand developers scattered all over the planet, connected only by the tenuous strands of the Internet? Certainly not I.
>
> ([Raymond et al., 2001])

The Internet has ushered in many changes in the software industry, and one of the most prominent of these is the Open Source movement. What used to be a closed market controlled by large, monolithic vendors like Microsoft and IBM, has flourished into a web of companies and individuals exchanging ideas and collaborating together to create free and open software. Despite much skepticism, this method of development has proven to not only be viable, but to produce high quality software which today powers everything from phones to mission-critical computer clusters. In this chapter we will look at the background and philosophy of the OSS movement, and the methods and practices employed to facilitate collaboration.

2.3.1 The Open Source Philosophy

While what most people will consider Open Source originates in the late 80s, the idea of shared intellectual property goes back much further. One of the more prominent examples is the Motor Vehicle Manufacturers Association founded by Henry Ford and other car manufacturers in 1911 to facilitate cross-licensing and ensure fair competition. The result was that car manufacturers were able to share technology and patents amongst each other without exchanging money. [Flink, 1976] In the software world the practice of sharing code was also common practice in the beginning. Computers were generally limited to researchers and academics, and it was considered natural to distribute the source code with the programs so others could modify and contribute back. Throughout the 70s, as computers become increasingly more powerful, and software more complex, this started to change. While free software didn't disappear, larger companies realised that there was money to be made, and used restrictive licenses to prevent modification and distribution. An example of this clash between the free and commercial culture can be seen in Bill Gates' now-famous "Open Letter to Hobbyists" [Gates, 1976], where he implied that what many called sharing was in fact stealing. This state of affairs continued throughout the 80s, with free software remaining limited to academic and hobbyist circles.

The modern free software movement and philosophy can in many ways be traced back to Richard Stallman and the GNU Project. The GNU Project was started in 1983 with the stated goal of developing "a sufficient body of free software [...] to get along without any software that is not free." [Stallman, 1985]. Their software was unified under a common license, called the GNU Public License which was published in 1989. The GPL stipulates that any software that uses GPL licensed code, must also release their own source code. The GPL was pioneering in that it introduced "copyleft", which gives users the right to distribute copies or modified versions of a work, as long as the license is kept intact. Essentially, this means any piece of software which uses GPL licensed code, must also carry a GPL license. This "viral" method of enforcing openness, has been a point of contention in the software world, with many opposing the FSF's methods **Skaff referanse**. There is, however, little doubt that the GNU Project was successful in their goal, as in combination with Linus Torvalds' Linux kernel, the GNU/Linux operating system is today used on everything from mobile phones to enterprise computer clusters. **Ekspander litt, Free vs OSS** The success of Linux has garnered attention from many parts of the IS community, notably from Apple and Google who both use Open Source software in combination with proprietary software to great effect.

2.3.2 Open Source Methodology

While there today are many differing grades of OSS, some of the most successful products have been developed, distributed and supported on a voluntary basis by and for the users themselves [Von Hippel, 2005]. This seeming altruism, and the ways in which high quality software is able to crystalise out of seeming chaos, has been the focus of many studies over the years. [Lerner and Tirole, 2002] phrase the question as “Why should thousands of top-notch programmers contribute freely to the provision of a public good?”. Raymond (2000), one of the leading OSS advocates argued that there are at least three motives for why someone would want to create or contribute to a project. First they may want to use the piece of software themselves. Second, they may enjoy the programming itself. And last, they may view contributing to important projects as a way of enhancing their own reputation among their peers. Surveys among programmers by Hertel and Niedner (2003) and Lakhani and Wolf (2003) have proved to support Raymon’s views.

2.4 Agile Software Development

As famously noted by Fred Brooks in his 1986 paper on software development, there is no silver bullet. Despite this there have been numerous attempts over the years, and many claimed success stories, at finding new development practices which can drastically improve programming efficiency and reduce the number of software bugs. In an industry where failure rates are estimated to be between 50-70% [Erickson et al., 2005] this doesn’t come as much of a surprise. This trend has in no way stopped, but in recent years there has been a move away from the more monolithic and all-encompassing processes of old, and towards leaner, more agile methodologies. In this chapter we will explore some of the techniques associated with Agile development, and the potential advantages and drawbacks they offer.

2.4.1 Background

There is no formal definition of what an agile development method is. The term is based on The Agile Manifesto [Manifesto, 2001] which states:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> *Individuals and interactions over processes and tools Working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan*
>
> That is, while there is value in the items on the right, we value the items on the left more.

This manifesto was created by a group of 17 software developers gathered in Snowbird, Utah to discuss lightweight development methods. This event was the result of many different directions of research into alternate software development techniques throughout the nineties. The dot-com boom, and an increasingly growing and changing software industry, was demanding faster time-to-market and shorter product-life-cycles. This was fundamentally incompatible with the established software development practices like Waterfall which emphasise linear development processes with well defined and separate specification and implementation phases. eXtreme Programming and Scrum were two of the leading champions of this new way of thinking about software development. These were both born out of corporate environments and as such had already been tested in the field.

Four years after the manifesto, a survey[Dybå and Dingsoyr, 2008] of European and American companies revealed that 14% of them were using agile methods, and that 49% of them were aware of and interested in adopting them. It might be rash to attribute all of this to the agile movement; indeed it might be tempting to argue that agile was simply able to fill a void created by an IT industry hungry for new ideas and a possible edge, but regardless of reasons, it is today one of the most talked about subjects in the software development industry.

2.4.2 Common activites

Erickson describes agility as a "means to strip away as much of the heaviness [..] as possible to promote quick response to changing environments." In essence, agile methods seek to split the development process into smaller, more manageable chunks without long-term plans. This is done to keep the project as flexible as possible, and to prevent you from having to make decisions until as much relevant information as possible is available.

Agile methods also place a lot of emphasis on person-to-person communication. Customer contacts are particularly important throughout the process to answer developer questions and clarify implementation issues. They should also be present during iteration meetings to represent the customer's interests during planning.

Most agile methodologies use daily face-to-face meetings among team members. During these meetings team members can explain what they have done previously, and what they plan to do. It also gives an opportunity to discuss problems and roadblocks with which they are struggling.

Another important distinction between agile and more formal methodologies is the focus on working software ahead of documentation. Software is seen as the primary measure of progress, and together with face-to-face meetings form the basis of the development cycle.

2.4.3 Extreme Programming

This concept can be seen as one of the ancestors to agile methodology as it was introduced a few years earlier (in the late nineties). It received its name from the concept of focusing on best practices at an "extreme level". These practices are as following:

- The Planning Game
- Small releases
- Metaphor
- Simple design
- Testing Refactoring
- Pair programming
- Collective ownership
- Continuous integration
- 40-hour-week
- On-site customer
- Coding standards

Arguably the most famous, and what many associates with XP, is the pair programming practice. This is a programming technique where two programmers sit together in front of a computer, and while one is coding, the other reviews each line of code subsequently. Studies have shown that this technique can be very beneficial, leading programmers to produce shorter and less error prone code.

2.4.4 Scrum

Early on, around year 2000, the practice was considered a mere buzz. It has sustained regardless, and many companies find the methodology useful. Due to its early use in the industry it has also been researched significantly more than most other agile methodologies, and is as such considered quite mature.

However, despite its popularity, many companies have failed to implement it completely. 40-hour weeks and pair programming especially have proven to be less used than the other practices.

Scrum is an iterative, incremental methodology for project management and -control. Today it is one of the most used and known agile methodologies. It is known for its high level of repetition through series of iterations, called sprints, which are usually from 2-4 weeks long. The product backlog is the core of the entire project. This can most easily be explained as a prioritised list of requirements from the customer. This list is updated at each sprint start to allow for flexibility in project planning. Like all agile methodologies it also focuses on high interaction with the customer; conducting meetings at the end and preferably also at the start of each sprint. These aspects leads to an extremely high visibility of the progression, available resources, and upcoming issues.

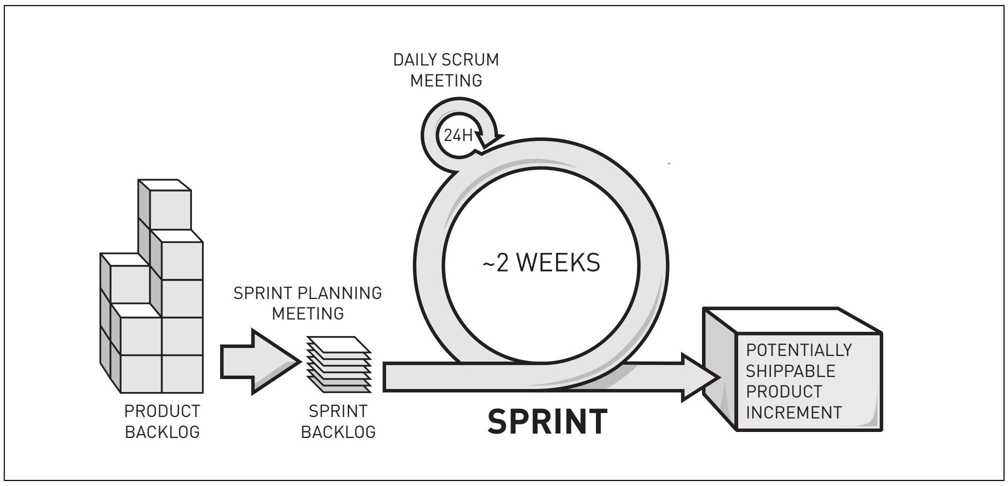


Figure 2.2: A typical Scrum process

2.5 Standardisation

Standardisation can be beneficial for both users and producers, and since Henry Ford first introduced the assembly line, the idea has become crystallised. The idea also applies well to newer areas of economics, such as ICT. When developing large, complex information systems or infrastructures, design and diffusion of standards is a crucial aspect. There is a great demand for standardisation, and there is even a common notion that it must be enforced in order to succeed. On the other hand, a large body of literature has proven that this is not simple, and that highly generic and standardised systems do not travel well. The larger the systems, the more complex and harder to standardise and distribute they become. As mentioned earlier in the HIS-chapter, information systems could be conceptualised as social systems where the interplay between the different components/actors such as users, management or technology is the focus of interest. One should not look at the technological part of an information system as an isolated product, it will always be influenced by the social context in which it has been developed, implemented and used in. They are not 'pure' technology, but rather socio-technical systems [Hanseth, 2000]. This thought implies that successfully adapting a completely "standardised", black box system in many cases is impossible. Rolland and Monteiro (2002) states that:

> *it is necessary to strike a balance between sensitiveness to local contexts and a need to standardize across contexts* ([**?**])

In relation to our project it is also worth mentioning the cultural gap which further contributes to increasing the difference in practice, and creates an even bigger tension between standardisation and localisation.

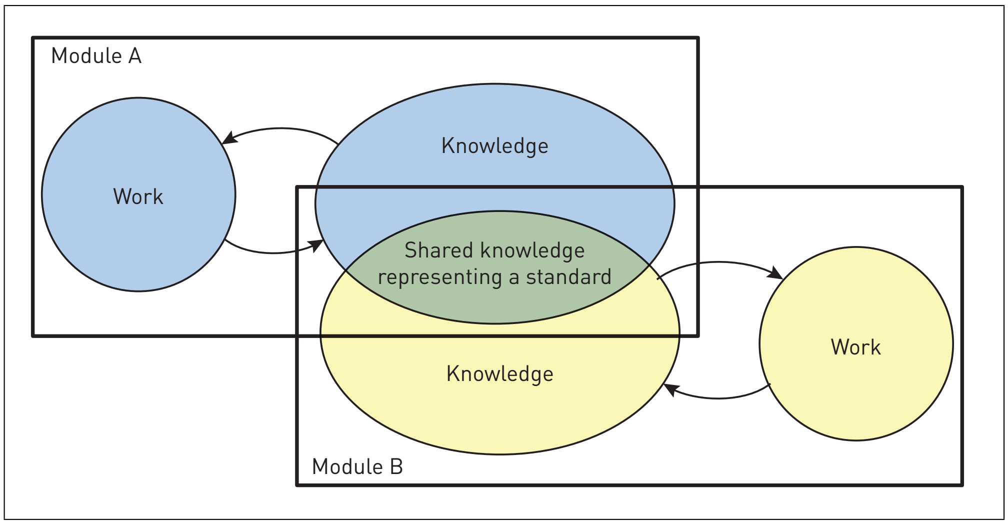


Figure 2.3: Shared knowledge representing a standard (adopted from Vaishnavi and Kuechler)

2.5.1 Motivation

One of the biggest benefits of standardisation might be the economical. Ole Hanseths paper [Hanseth, 2000] presents the basic lessons learned by economists who have studied standardisation processes. Economics of standards have certain characteristics and concepts that are very interesting to look at, such as increasing returns and positive feedback, network externalities, and installed base.

Increasing returns means that the more a product is produced, sold and used, the more valuable or profitable it becomes. This applies especially well to the IT sector, and software are definitely a type of product with these characteristic. Most of the production costs occur in the initial design and development process. When you have developed the software, duplication costs are negligible, which exponentially lowers the unit costs the more units are sold. Another aspect that is even more important for IS and Information Infrastructures, is the positive feedback gained as more users are using the product. A perfect example of this is the Internet. Its value grows as more and more people are using it.

Effects of positive feedback(often called network effects): Network externalities, it is better to be connected to a bigger network.

- Network externalities and positive feedback can lead to path dependency. Newer version needs to be compatible or build on the installed base. (ex. email)
- Increasing returns can lead to lock-in. High switching costs etc. Not only caused by hw/sw, also information structures, complex networks. Gateways(ex. telefon og tv over internett)
- Possible inefficiency: Not given the best solution wins(ex. VHS over BETA, MS over Apple)

The previous list points out the more general possible effects of standardization, but let us try to find some more relevant reasons for standardization in the case of IS:

Enforce some notion of control and coherence across the different context.

"Standardization enables coordination, which in turn enables the exercise of control over distance" (Law 1986, cited in Rolland & Monteiro 2002)

2.5.2 Trade-offs/Approaches

Many countries have attempted, or more correctly are in the process of, standardising their health care systems which in the longer run will hopefully lead to a national standard health system. Many different approaches have been tried with different levels of success. [Coiera, 2009] looks at three different strategies in three very large initiatives in UK, USA and Australia.

- UK (top-down): The English National Health System (NHS) National Program for IT (NPfIT) in many ways serves as an international beacon for healthcare reform, because of its clear message that major restructuring of health services is not possible without a pervasive information infrastructure. Yet no one can deny that there have been plenty of setbacks, misgivings, clinical unrest, delays, cost overruns, and paring back of promised functionality, culminating in demands from some political quarters to shut down the program. The NPfIT was bound to experience some difficulties purely on the basis of its scale and complexity.

- US (bottom-up): The United States, with its highly fragmented and decentralized health system, sits at the other extreme.

- Australia (middle): Australia have directed their initial public e-health investments into developing nation-scale standards, well before contemplating any actual systems being built. By definition, there is always a lag between standards as published and as implemented on the ground.

Chapter 3

Methods

This chapter describes the research and data collection methods used in our empirical work, carried out during the stay in India.

3.1 Research method

Research methods can be classified as quantitative or qualitative based on characteristics such as data collection methods, research goal and approach. One should choose the method based on what you aim to study[Silverman, 2005].

As the name implies, quantitative research methods focuses on numbers and the assumption that numbers can represent strong scientific evidence for a phenomena. Quantitative research was originally developed to study natural phenomena in the natural sciences, but have over the years been accepted in the social sciences, including methods like surveys, laboratory experiments and mathematical modelling. Nevertheless, research in social sciences tends to be based on qualitative methods, which were developed for studying social and cultural phenomena. Common qualitative methods are action research, case study, ethnography and grounded theory.

Quantitative research is most often used to prove specific theories and provide understanding about natural phenomena, while qualitative research aims to provide insights and in-depth knowledge to better understand a social phenomenon. Qualitative methods are increasingly being employed in IS research, due to a general shift towards focusing on managerial and organisational issues instead of technical issues. To gain deep insights and understanding of the challenges of distributed software development, we have chosen to use an interpretive case study approach.

3.1.1 Action Research

Action research implies action taking, more specifically meaning that the researcher contributes in a project or organisation (normally related to the subject of study), and works together with the other members to solve a problem. HISP can be regarded as an ongoing action research project. Action research focuses on solving real life problems, with an emphasis on the interaction taking place. Through this process the researcher gains new theoretical and practical knowledge. Susman and Evered describes the process by outlining 5 phases, illustrated in figure 3.1 [Susman and Evered, 1978].

In our case we contributed to the project by developing two modules for the OpenMRS system being developed. We worked individually on one module each for the duration of our stay. Through this we worked closely with the other members of the development team, and gained a good understanding of the practices, processes, and communication methods in the project.

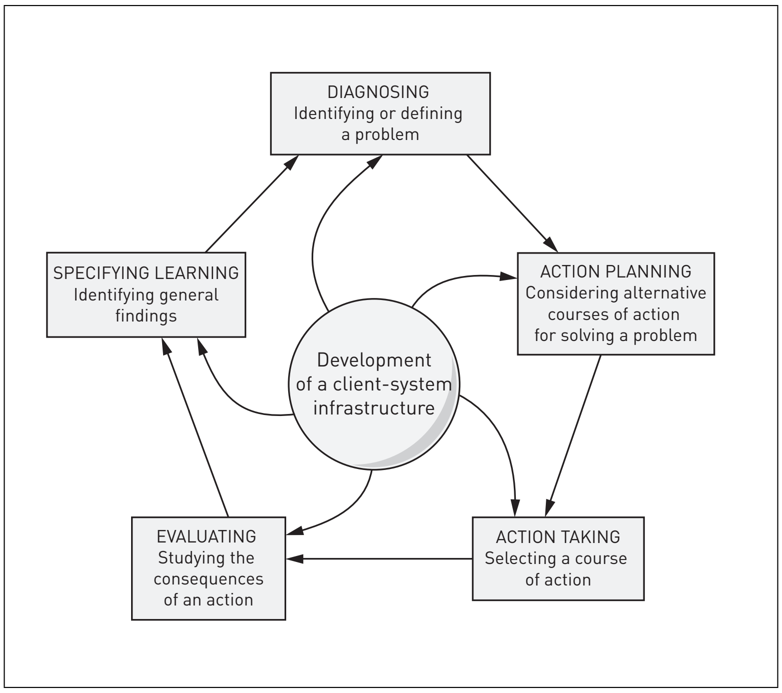

DIAGNOSING
Identifying or defining a problem
ACTION PLANNING
Considering alternative courses of action for solving a problem
ACTION TAKING
Selecting a course of action
EVALUATING
Studying the consequences of an action
SPECIFYING LEARNING
Identifying general findings
Development of a client-system infrastructure

Figure 3.1: Action research

3.1.2 Interpretive case study

According to Klein and Myers, IS research can be classified as interpretive if it is assumed that *"our knowledge of reality is gained only through social constructions such as language, shared meanings, documents, tools, and other artifacts"* [Klein and Myers, 1999] Interpretive Research was conceived as a counter method to Positivism. Positivism looks at reality as fixed and measurably independent of the observer. Interpretive research is more flexible, saying that knowledge is only gained through social constructions such as language, meanings, documents and other artefacts. As our goal is to gain deeper insights and build on the social constructs of the Shimla team, our approach can be classified as interpretive. Case study is a commonly used method in qualitative IS research, and entails that the researcher conducts an in-depth study of an event, activity, process, person or group of individuals over a certain time: a case. A variety of data collection methods can be used, often interviews and observations.

3.2 Data collection methods

Our research is based on qualitative collection methods commonly used in case studies, namely observation, interviews, questionnaires and document analysis. Here we will give a short account of each of the methods and describe how we specifically collected the various data.

3.2.1 Observation

We took actively part in the development team in Shimla, and interacted with a number of people on a daily basis. The size of the team varied from about 10 to 15 people, depending on people's obligations (such as meetings at the hospital or other instances). Some of the members came from HISP India centrally, or other regional offices, and only stayed for a couple of days or weeks to fix something specific or do a smaller task. The core group of members who were almost always at the office was about 8 people. This size was easy to deal with and we could quickly establish a picture of the whole team, the hierarchy of the members and their roles and characteristics, as well as become familiar with them on a personal level. The office had an open space solution, which made it easy to observe what was going on during the work hours.

We initially wrote daily logs, but due to long and exhaustive days at the office, this was often postponed and done in batches a couple of times per week instead. By doing this, we were able to keep track of the progress, and make good notes of different events and discussions that arose during our stay.

People we interacted with during our stay:

- In Shimla:

  Project leader The head of the office in Shimla and project coordinator of the OpenMRS implementation

  Health Information Officer Medical Doctor with a master's degree in Medical Science and Technology

  Implementers 3-4 people with various IS and HIS backgrounds.

  System administrator Situated in New Delhi most of the time, but on site for a period at the end of our stay.

  Senior developer Stayed one and a half week after our first arrival, before leaving the project

  Vietnamese developers Two developers from Vietnam came to Shimla at the beginning of April, with the purpose of staying for 6 months.

  Trainee developer Newly educated developer, hired a month before our first arrival.

- Other:

  HISP India Manager Most of the time situated in New Delhi.

  Remote developer Left Shimla a week before our arrival, currently working from Vietnam.

  System architect Based in Dublin, Ireland.

  OpenMRS core developer Based in Seattle, USA, twelve and a half hour behind Indian time

In addition we were in touch with the OpenMRS community through mailing lists and IRC chat. In the beginning it were on a daily basis, but later, especially this spring, it have been very little.

3.2.2 Document analysis

Much data was gained electronically, through chatting, mail correspondences, document sharing, and Skype meetings. Since the project team is not only situated in Shimla, but consisting of team members scattered all over the globe, there was a lot of communication, and as such documents constitute a noteworthy part of our analysis alongside the other types of collected data.

3.2.3 Questionnaire

A small questionnaire for all the team members was conducted. Because of the team size, we did not achieve any statistically solid, quantifiable data, but the collected data was aiding us in forming a clearer picture of the team, and team members' assumptions and opinions about certain issues or phenomena.

3.2.4 Interviews

We conducted a semi-structured interview with the project leader. This was the only prepared interview we conducted, but we also gained much qualitative data through informal conversations that can be regarded as unstructured interviews with other team members. A lot of our information was gathered through these informal conversations, as it was easier for team members to discuss things more freely and in-depth when not at work. We took notes whenever these occasions occurred, and the formal interview with the project leader was recorded and transcribed later.

After our last stay in Shimla we also conducted a final telephone interview with the senior developer in Vietnam which proved very helpful.

3.3 Scope and limitations

Our research was conducted over two visits. The first period approximately five weeks, between October 20th, and November 26th 2010. The second was about seven weeks, from february the 15th to april the 9th. During these two periods we took part in the development of the project's OpenMRS implementation taking place locally, and interacted with and observed the people associated with it.

Other development taking place in Shimla was not taken into consideration in our research. We did not research the implementation at the hospital, nor did we interact with any of the users or focus on that aspect of the development process. The stage at which the development was did not warrant this type of research, and we focused instead on the internal development process. Finally, we have not focused on the HIS aspects of the system.

3.4 Reflections

Overall our data collection was fairly passive. Most of the time spent in India was spent developing the different modules, and as such we gained a quite one-sided view of the system development as a whole. In hindsight we would likely have had better data if we had taken a more active part in the implementation and training effort at the hospital.

Our communication with the other developers staying in Shimla could also have been better. While there was a somewhat difficult language barrier due to their poor english, it would likely have helped us if we had put more of an effort into investigating their experiences and point of view in the system development process.

Chapter 4

Research context

In this chapter we will describe the context of the project, including India, HISP, and OpenMRS.

4.1 India

The Republic of India is situated in South-Asia, with borders to Pakistan, China, Bhutan, Nepal, Bangladesh and Burma. It is the seventh-largest country by geographical area and the second-most populous, with over 1.2 billion inhabitants[1]. It consists of 28 states, each ruled by a governor appointed for 5-year terms by the federal president. The states have primary control over education, health, police, and local government. States are further divided into districts. There are also seven Indian union territories, which are sub-national administrative divisions ruled directly by the federal government. India is popularly called the world's largest democracy, and

[1]Source: Census of India 2011, `http://www.censusindia.gov.in/2011-prov-results/prov_results_paper1_india.html` (June 11, 2011)

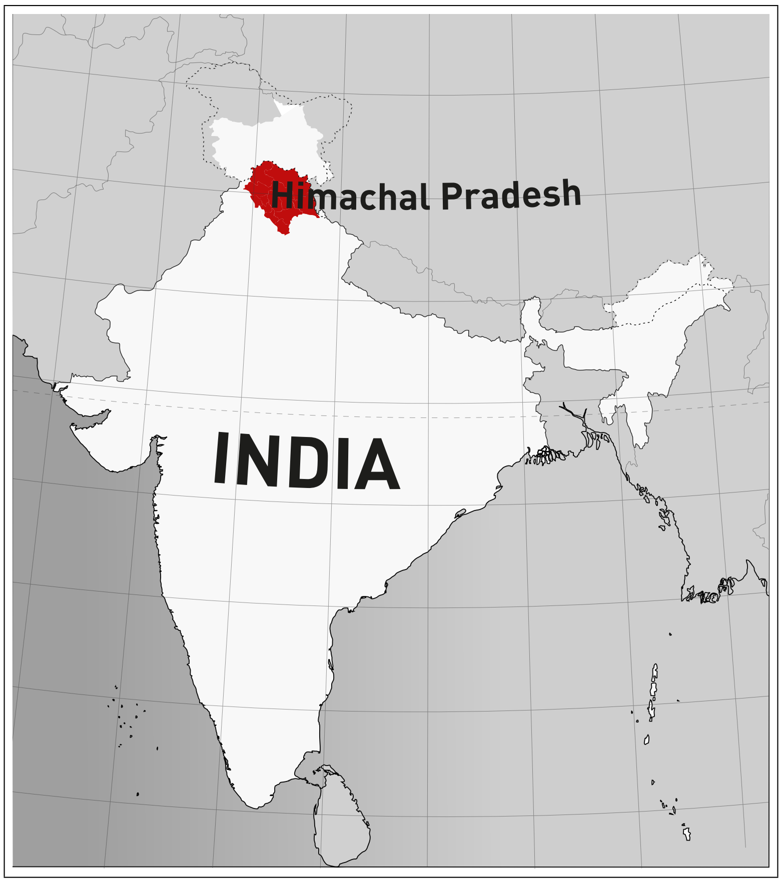


Figure 4.1: Map of India

have been so for more than 60 years, since their independence from Britain in 1947.

India is a strikingly diverse country, it resembles more a continent than a single country. There are a vast number of different languages, climates, cultures and customs. Although hard to measure exactly, the most recent census of 2001 states that 29 languages have more than a million native speakers, 60 have more than 100,000 and 122 have more than 10,000 native speakers. Officially, the Indian constitution approves 22 ”scheduled languages”, which are languages that may be used by states in official correspondence. In Himachal Pradesh, Hindi is their “scheduled language”, as well as the federal primary official language, together with English as secondary official language. Hindi is by far the most common language, spoken by about three-fifths of the population, in one form or another.

The last 15 years or so India have had an astonishing rapidly economic growth, partially due to their out-sourcing service accomplishment in the software and ICT sector. Walsham states:

> *It is likely that India will remain a major player in the ICT industry for years to come and thus its global image as an ICT success story will continue* ([?])

On the other hand there are a lot of things that are not so simple. In respect to the increasing wealth from this out-sourcing industry, it have been questioned how much this benefits the poor. The under-five child mortality rate was as of 2009 66 (6.6%)[2] and the percentage of malnourished children under five being shockingly 46%.

4.1.1 Himachal Pradesh

Himachal Pradesh (HP) is one of the northern-most states of India, situated at the foot of the western Himalayas. It is divided into 12 districts: Kangra, Hamirpur, Mandi, Bilaspur, Una, Chamba, Lahul and Spiti, Sirmaur, Kinnaur, Kullu, Solan and Shimla. It is a very hilly area, with elevation ranging from about 350 to 6000 meters above sea level. This leads to a diverse climate, from hot and sub-humid in the southern low-lands, to a cold, alpine and glacial climate in the northern, mountainous parts. Above 2200 meters snowfalls are common in the winter, which lasts from late November till mid March. As of the latest census from 2001, about 6.1 million people live in the state.

[2]Source:Unicef: `http://www.childinfo.org/mortality_ufmrcountrydata.php` (June 11, 2011)

HP is considered to have one of the better health care situations in India. For instance, the infant mortality rate in HP is below one third of the Indian average. The chillier climate is probably one of the main reasons for this, as they do not have as many tropical diseases nor the same sanitary needs as in most parts of India. When compared to the western world, however, the numbers are still quite high, in 2006 totalling 36 deaths out of 1000. In Norway, by comparison, this number is 2.8 out of 1000.

4.1.2 Shimla

Shimla is the state capital of Himachal Pradesh. From 1864 it was used as the summer capital by the British Raj. Due to the burning heat in the Calcutta area, the British made the effort of moving most of their administration up to this much chillier place every summer. It has about 400 000 citizens, which makes it a relatively small city by Indian standards.

4.2 DDU Hospital

Deen Dayal Upadyay Hospital(DDU) is the selected pilot hospital for the project, and is located near the Shimla town centre. The official numbers states that it holds about 300 beds, but in reality there are a maximum 200 beds. DDU is an old heritage hospital and was opened in 1885. The 125 year old buildings are in a very bad condition, as there are strict rules and guidelines for repairing heritage buildings. It is built entirely of wood, and during the rainy season there are lots of leaks, causing bad conditions and posing a risk to computers and other electronic equipment. There are also a lot of monkeys in the area, vandalising the buildings and power- and network cables. Nevertheless, due to its location and the lack of other public hospitals

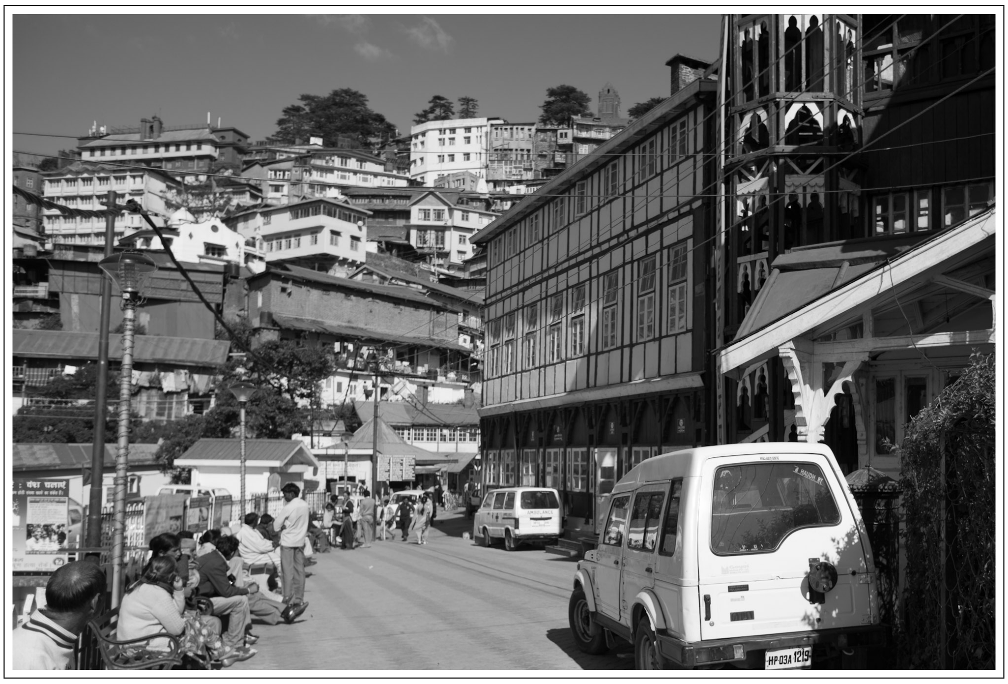


Figure 4.2: The DDU Hospital

nearby, it is a much visited hospital. There are an average of about 1200 out-patient(OPD) visits per day. There is a simple map floor plan drawing of the area below(fig. 4.3).

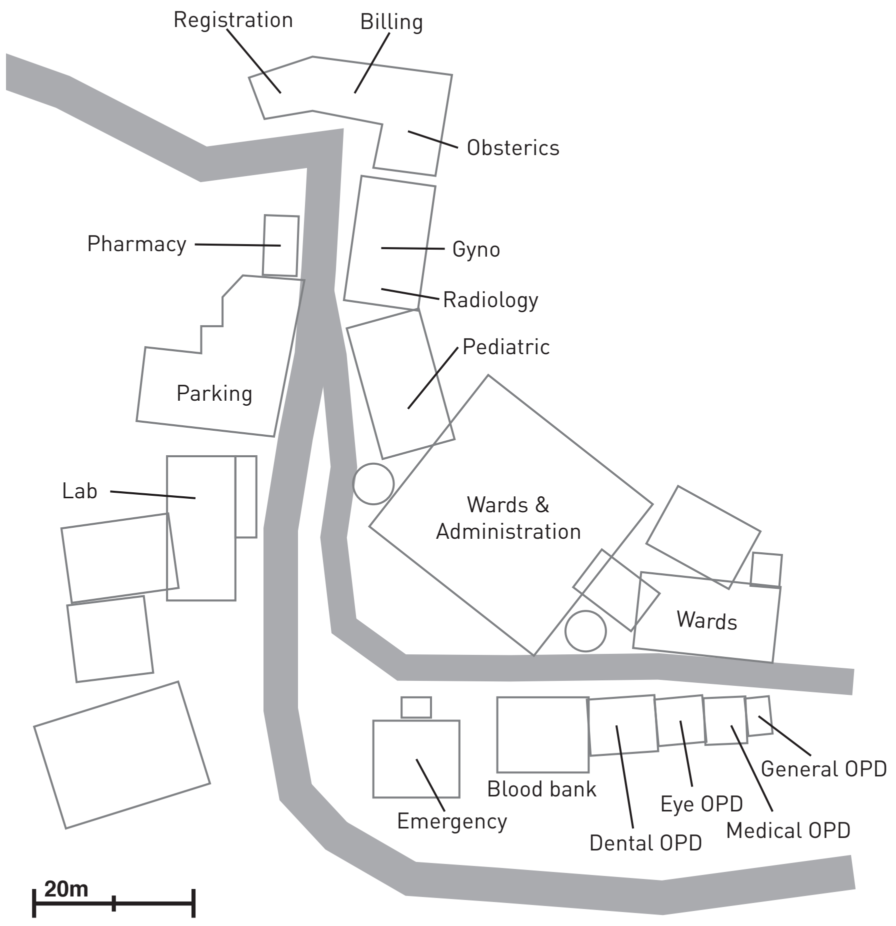


Figure 4.3: The DDU Area

4.2.1 Departments

Registration

Every patient coming to the hospital first has to visit the registration department. Here they have to provide their name, age and address which then get

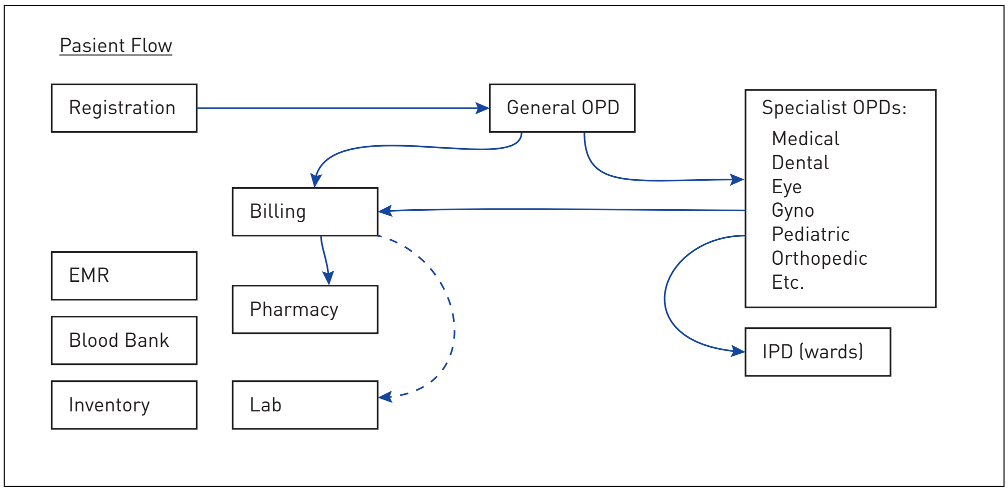


Figure 4.4: Patient workflow at DDU

entered into OpenMRS. In the same process they also get a further referral, for most of the patients being general OPD. When this is done, a patient slip, containing their name and newly created id, among the rest of the demographic data and the name of the department they have been referred to, gets printed out on a dot-matrix printer and given to the patient. This is the only form of identification and proof the patient has to get further in the process. It is very important that he keeps the slip, as we will see later in this description of the departments. Although this is not the right place to make an analysis or critique of the system, it is worth noting that this is the starting point of a patients life in the system, hence the root of many of today's biggest issues with it. The demographic data registered is often too incomplete to be unique for the patient, and the generated, unique ID is too long for a patient to remember. This may seem absurd for someone coming from a country where everyone is used to memorising a unique number and using it from an early stage in life, but this is the situation and a source of great issue, considering the redundancy in the data base building up and the tracking of revisiting patient(who will most likely get registered as a new one). It is hard to do anything about it - many of the patient do not know how their names are spelled correctly, or what their address is, or even what their correct date of birth is - they often vaguely know their age.

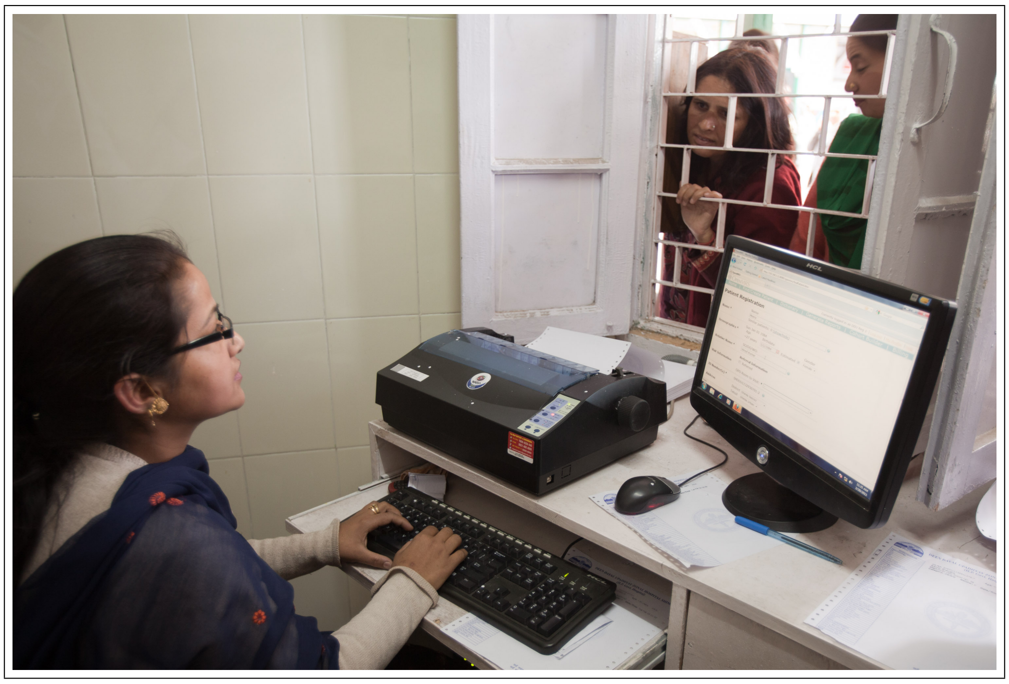


Figure 4.5: One of the two desks at the registration unit

General OPD

The hospital has one general Out-Patient Department (OPD). This is the busiest of all the OPDs. It is operated by one or two physicians, depending on the work load and the doctors' availability. When a patient arrives, he will present his slip to the physician or a data clerk, and will then receive a consultation by the physician. The physician will decide what to do next, in most cases (50-70% of the visits) he will prescribe drugs and give the patient some instructions before sending him home. In the rest of the cases he will refer the patient further, to some of the specialist OPDs, an In-Patient Department (IPD) ward or to some of the investigation units. When there is an investigatory procedure that needs to be done(i.e. a blood test or an x-ray), the doctor fills out a requisition slip for the patient, containing his ID, instructions and type of investigation. Before the tests are carried out, the slip needs to be stamped at the billing unit for confirmation on received payment.

Billing

Before the patient can leave the hospital, he has to pay for the services provided. This is often the first time since the registration OpenMRS is used. The patient gives the identification slip to the billing clerk, who looks up the patient in the system and registers the items he should be billed for based on what the physicians have written down. If there are tests that are yet to be performed, the clerk also register these based on what's written on their lab tests-slip. Finally a bill is printed, the patient pays and the papers get stamped with the date. The patient is then free to go, unless there are remaining tests to carry out.

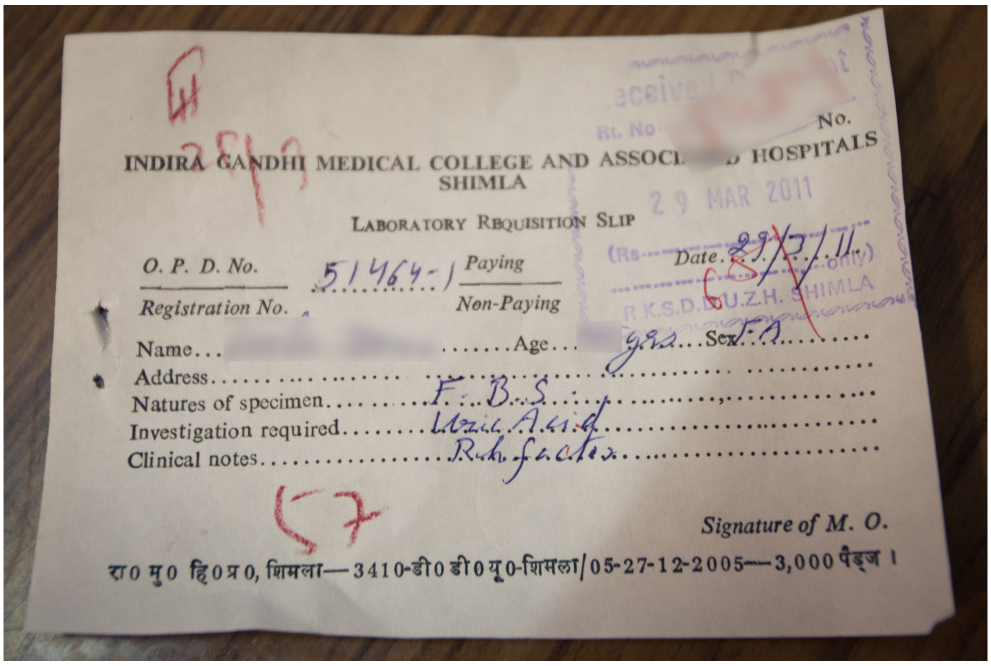
Laboratory

DDU have a laboratory capable of performing many of the most common and useful tests at a hospital. They have a medical robot which makes it possible to analyse up to a hundred blood tests in one single batch each day.

The patients bring their requisition slips(as shown in fig. 4.6) from the physicians, pre-paid and approved with a date-stamp at the registration, and samples(blood or urine) are taken. The lab technicians make a work list to keep track of the tests. The requisition slip is kept, and when the tests are done, the results are returned to the patient, often together with other sheets of paper with results. The patient then has to return to the physician who requested the test to get a follow-up consultation.

Specialised OPDs

The specialised OPDs are:



घ 2519

ecived [redacted]

Rt. No [redacted] No.

29 MAR 2011

(Rs .... only)

R.K.S.D.D.U.Z.H. SHIMLA

INDIRA GANDHI MEDICAL COLLEGE AND ASSOCI[redacted]D HOSPITALS
SHIMLA

LABORATORY REQUISITION SLIP

O. P. D. No. 51464-1 Paying / Non-Paying Date 29/3/11

Registration No.

Name... [redacted] ....Age... [redacted] yrs...Sex F

Address......................

Natures of specimen........F.B.S.......

Investigation required........Uric Acid........

Clinical notes..............Rh. factor.........

57 68

Signature of M. O.

रा० मु० हि०प्र०, शिमला—3410-डी० डी० यू०-शिमला/05-27-12-2005—3,000 पैड्ज ।

Figure 4.6: A filled-out lab-requisition slip

- Surgical:
  - General
  - Obstetrics and gynaecology
  - Orthopaedic
  - Ear, nose, throat(ENT)
  - Dental
- Medical:
  - General Medicine
  - Paediatrics
  - Skin
  - Casualty (though grouped in Medical OPD it caters for emergency services of all the Specialties, but as it does not have direct access to any of the OT's it is currently grouped under Medical.

> Further understanding is needed to decide its functionality, which may even require it as a separate classification. Most Hospital information systems have emergency as a separate module.)

The specialised OPDs work in much the same way as the general.

Pharmacy

There is one pharmacy located inside the hospital grounds, and many other drug stores located nearby. None of these are run by the hospitals, but there are no restrictions on where to buy their prescribed drugs, although you won't find the less common drugs in other stores than the one located on-site.

Wards

The hospital has about 200 beds, distributed on [tall] different wards, classified by patient groups and gender(all wards are either male or female). Every in-patient has a small paper-based record-sheet where all the information gets registered and updated. Each ward has a small office the nurses uses for record-keeping.

4.3 HISP

The Health Information System Programme (HISP) was initiated in 1994 by researchers from Norway and South Africa as a pilot project to establish a research and development programme for developing health information systems. Since then it has expanded into a global network active in about 15 countries, mainly in Africa and Asia. The countries are:

- South Africa
- Mozambique
- Norway
- India
- Malawi
- Tanzania
- Vietnam
- Ethiopia
- Nigeria
- Botswana
- Zambia
- Zanzibar
- Sierra Leone
- Tajikistan
- Mali

Their aim is to improve the health care systems in developing countries by increasing health care workers' capacity to make good decisions based on accurate health care data. Their vision is:

> *”Development and implementation of sustainable and integrated health information systems that empower communities, health workers and decision makers to improve the coverage, quality and efficiency of health services.”*

HISP is not a singular, defined project, but is spread globally as a loose network of projects and partners involving various academic institutions and governmental and non-governmental health institutions. It follows a participatory approach to support local management of health care delivery and information flows in selected health facilities, districts, and provinces, and its continued expansion within and across developing countries.[Braa et al., 2004]

4.3.1 DHIS

District Health Information System (DHIS) is a management and data collection tool developed by HISP to capture and analyse aggregated health data. The development started in 1996 and has been ongoing since, through an iterative process consisting of different HISP members. The first releases of the system, 1.3 and 1.4, were based on Microsoft Access for data storage and Visual Basic together with Excel for user interaction. The software was already free but dependent on other commercial and proprietary components, which was considered a big drawback, and they chose to port the whole application to Java in subsequent versions. Another important motivation for the conversion was that it made them able to benefit from using other, powerful open source tools and software such as Spring, Hibernate and JUnit. The development of the new, open source and web-based version started in 2003 at the University of Oslo. The first release came in February 2008, after a long, collaborative development process involving students, researchers and experienced developers in Norway, India, South Africa, Ethiopia, and Vietnam.

4.4 HISP India

HISP India is an Indian NGO (non-governmental organisation) and part of HISP. Its main activity since the start in late 2000, has been development and implementation of DHIS in different parts of India. It consists of a multidisciplinary group of workers with backgrounds in informatics, medicine, public health, computer science, anthropology and development studies. Over the years of implementation, the organisation has grown and gained much knowledge, together with a growing governmental support.

The state government of Himachal Pradesh contacted HISP India after an unsuccessful two-year process where they had attempted to find a company willing to develop a HIS for their district hospitals.

The Request for Proposal (RFP) that HP had submitted was substantially as follows:

1. Deploying a Health Management Information System (HMIS) to encompass information gathering, knowledge management and facilitate decision making.
2. To develop and provide health information infrastructure, to help in daily operations, clinical practice and ensuring quality of service provided to the citizens in an efficient way.
3. Effective utilisation of resources (human, capital etc.).
4. Adhering to standards and leveraging the latest technology developments in the health practice.

While HISP India had no experience developing such a system, they were nevertheless very interested in learning, and were over time able to work out a suitable implementation plan along with the HP state government.

4.5 Shimla Team

The team in Shimla consists of a number of implementers, developers, and health-care professionals. The project leader has a background from the Indian government, and has worked in various parts of HISP previously. Together with three others she oversees specification, implementation, training and communication with the people at the hospital. Each implementer is assigned one or more modules as their chief responsibility. The doctor on the team is responsible for overseeing the medical consistency and usability of the system as a whole.

In addition to the OpenMRS project the office is also responsible for the deployment of DHIS in the state. There is some overlap of work for some people doing OpenMRS planning and training, but for the most part DHIS work is done by other employees who don't work with OpenMRS.

The local team initially requested three software developers for the duration of the project. The work involved was estimated to require quite a bit of skilled labor from the start, and to continue for a long time as the scope of the project expanded throughout the state. This was initially fulfilled with the help of various students and Ph.D candidates associated with HISP, who worked on site in Shimla. However, due to their academic background and for various personal reasons, these developers were only available for a short period and the last of the initial developers left Shimla shortly after we joined the project. In addition to this there is one trainee developer recruited locally. Many attempts were made by the HISP India manager to recruit skilled developers, but it proved to be very hard due to the costs associated and the fact that few developers are available outside the major technological hubs in India. The trainee turned out to lack the skills necessary to help much in terms of software development.

4.6 OpenMRS

OpenMRS is an open source software project developing a medical record system platform to benefit health care in developing countries. The project was conceived in 2004 by Paul Biondich and Burke Mamlin from the Regenstrief Institute, Indiana, USA during a visit to Kenya. They recognized the need to scale up the treatment of HIV in Africa, and in cooperation with Hamish Fraser of Partners in Health [3] and Chris Seebregts of the South African Medical Research Council, they founded OpenMRS.

While the focus was to be on treatment of HIV and TB in sub-Saharan Africa, it was quickly decided that OpenMRS was to be a full featured EMR system. A broader aim was for OpenMRS to be an open source, collaborative project as a foundation for EMR development in "the field", the developing countries.

4.6.1 Design Characteristics

Upon project start, OpenMRS evaluated other EMR-initiatives in development countries, and formed a belief that

> *"overwhelming need for basic clinical data management (often to provide data to funding agencies) along with the need for rapid response in the face of limited technical resources led to many disparate, "stovepipe" efforts which often stored non- coded values and rarely scaled well."*

[Seebregts et al., 2009]

To overcome these challenges their solution has been to try to make OpenMRS highly flexible and modular. Through providing a foundation and the building blocks for new projects to start implementing and tailor the system to their needs easily, they hope to succeed and make more sustainable EMR-implementations than others before them. They earnestly admit that OpenMRS can be seen as "just another stovepipe", but hope that by using open source tools, promoting localization through a modular design, and sharing the work, it can be a seed to something bigger.

[3] `http://en.wikipedia.org/wiki/Partners_In_Health`

The core of OpenMRS is a web application programmed in Java using a number of other open source components, including:

MySQL Widely used relational database management systems

Tomcat Apache servlet application for Java web applications

Hibernate Object to relational mapping and persistence application

Spring Application framework

Apache Subversion(SVN) Revision control and code sharing tool. OpenMRS maintain their own SVN server for the code of both the application and also all the modules.

The high level architecture is illustrated in figure **??**.

Data model

The core data model is an enterprise-quality data repository based on 30 years of experience from the development of Regenstrief Medical Record System, combined with practical experience from Partners in Health and other developmental partners. We will not describe the model in too much detail, but have created a very simplified model to illustrate the most important domains (fig. 4.8):

Patient Basic information about patients in this system. Can be seen as the center of the model, since the majority of data getting stored relates to individual patients.

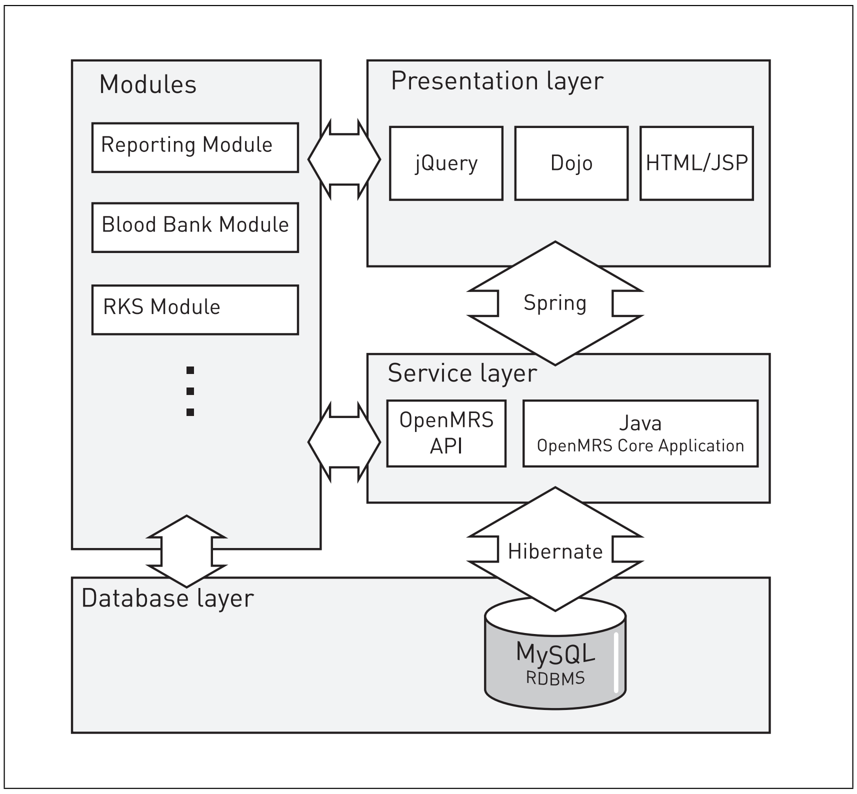


Figure 4.7: OpenMRS high-level architecture

Concept Concepts are defined and used to support strongly coded data throughout the system.

Encounter Contains the metadata regarding health care providers' interventions with a patient.

Observation This is where the actual health care information is stored. There can be several observations per Encounter.

Among these domains, "Concept" could need some further explanation. A concept can in practice be almost anything, but a rule of thumb is to use it sparsely and create well-defined concepts to avoid ambiguity and redundancy. Concepts can also have sub-concepts. A good example is a blood type-concept. There are eight different types of blood - eight concepts. But one

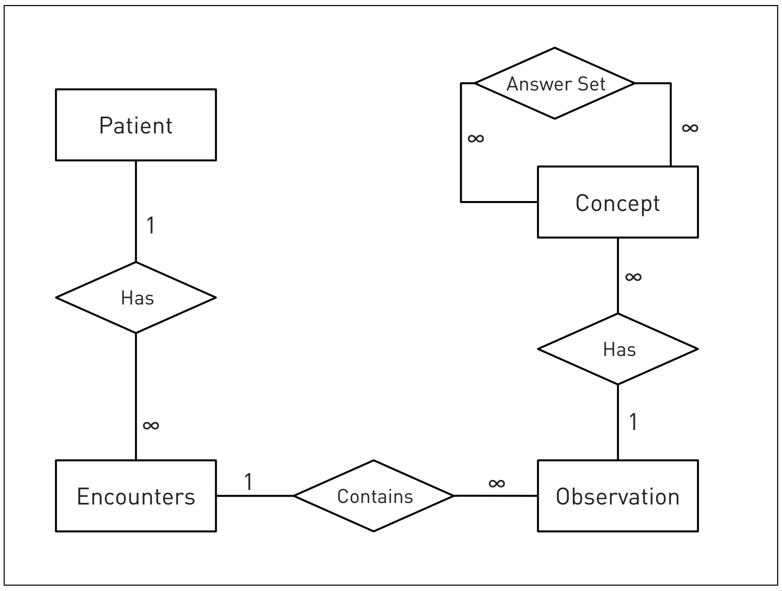


Figure 4.8: Simplified OpenMRS data model

must not forget the concept of a blood type. Therefore, one should create one blood type group-concept, and a concept for each individual blood type. These individual blood types serves as answer concepts of the parent blood type concept. The purpose of the concept dictionary is that one can easily create flexible semantic relationships and context-dependent metadata for almost anything throughout the whole application.

Application Programming Interface

In addition to its scalability, the data model is tightly constrained. This leads to a high complexity and an inherent barrier for new developers to start coding. OpenMRS uses an Application Programming Interface (API) to hide the database transactions and make it possible for developers to use the model in a normal object-relational manner, such as "getObservations(patient)". Hibernate provides this object/relational persistence layer between Java and the data base, regarded as the backbone of OpenMRS. The API solution also provides a higher level of data integrity as it limits the ways of interaction with the underlying data model.

Spring/MVC

OpenMRS uses the Model-View-Controller(MVC) design pattern. This is a vastly used architectural pattern. The purpose of this is to isolate business logic from the user interaction and by doing so reducing the architectural design complexity and elevate the flexibility. Spring is the tool used to manage this.

4.6.2 OpenMRS community

As in many other open source projects, the OpenMRS community is big and scattered. At the same time, the group of core developers considered the biggest contributors are not so vast and mainly based in the US. Most of the communication happens electronically, and OpenMRS uses a number of different internet-based collaboration and communication tools. Each tool has their functions and together they cover all forms for needed communications. OpenMRS are doing what they can to make their community grow, and they have recently stepped up their efforts. Their web pages are newly redesigned and they have started to promote themselves through external social network channels such as Twitter and Facebook. On their wiki-page they post some of their collected activity data, and this is a month-by-month summary for the last 3 months. Since it does not go further back than the three last months, it is likely to believe they have just started with this collection. However, the stats clearly show that the activity and interest in OpenMRS has increased over the last months.

2010	Sep	Oct	Nov	Dec
Announcement List	232	236	244	253
Dev List	281	293	309	322
Implementers List	306	317	329	340
Twitter Followers	695	762	840	900
Facebook Fans	875	1,367	1,463	1,5
OpenMRS ID's		498	580	679

2011	Jan	Feb	Mar	Apr	May
Announcement List	260	266	280	292	297
Dev List	347	354	381	376	377
Implementers List	351	348	349	354	359
Twitter Followers	969	1,008	1,076	1,143	1,217
Facebook Fans	1,57	1,59	1,605	1,616	1,848
OpenMRS ID's	784	829	931	1,044	1,124

We will briefly describe the most important means of communication in the project, their purpose and advantages below.

Mailing lists

The OpenMRS community maintains several mailing lists, divided by specific roles or functionalities, namely:

- Announcements
- Developers
- Implementers
- Users
- SVN Commits
- Security Updates
- Infrastructure Status
- Interns
- Mentors

The developers-mailing list is the most active of the lists above, and serves as one of the primary channels of communication among developers. Compared to other asynchronous forms for communication over the Internet, mail is most likely the one with highest chances of getting received and read. On the other hand, people do not like spam, and if you send an e-mail on the mailing list it should be of a certain level of importance and relevance to the receivers. This turns out to not always be the case, and many developers have a very low threshold for using mail.

Although the majority of the active contributors to the mailing list are also the ones who have been working in or on OpenMRS the longest, there are a lot of new and inexperienced developers who use the list. These normally receive answers to their questions within a day of asking. Since everyone sees the mail and at the same time knows that everyone else on the mailing list can also see it, people feel obligated to answer if they can.

Wiki

OpenMRS provides a comprehensive wiki that serves as the main documentation pool, though primarily for developers and implementers. It consists of user guides for entry and setting up the development environment, together with more detailed documentation on both modules and the core system.

The most important information seems to be up to date, but module-specific documentation is often missing or highly outdated. One page had a header bluntly stating that "This User Guide is basically useless. Needs to be rewritten.". It is also not mandatory for a developer to create any documentation on what he is making, which is probably one of the main reasons for this issue. In open source – and especially highly distributed – projects, documentation is an important source of knowledge, and the lack of such documentation can be frustrating, especially for new contributors. This is also the case in OpenMRS and a good example can be found on the developers mailing list:

From: Sender 1
Just a friendly reminder: the [...] module is completely worthless to everyone else besides its main developers until someone writes the widget reference in the wiki...

From: Sender 2
It's required by the [...] module.

From: Sender 3*
Wow, [Sender 1], calling me out! :) There is a wiki page for the module - [Link] It demonstrates how to use it by default (granted no reference here of all possible options, but I wouldn't call it useless). There is also a page that demonstrates the widgets in action if you've installed the module, available here: [Link] If you are a developer, you can look at the source of this page to see the variety of ways/options that the page demonstrates the tag can be used. So, agree that a reference of all options would be nice (and I encourage anyone to help take this on), but it should still be usable in it's present form...

From: Sender 1
sorry [Sender 3], i wasn't meaning to call you out. i've just noticed that this module is required by a number of modules [...] but I haven't found myself eager to read through .java files to see what the module really does whenever i need a textbox. You've done all this work to make ui stuff easier – share the wealth.

*Sender 3 is the main developer of the module

IRC

In many cases, the most responsive way of communication is through the OpenMRS Internet Relay Chat(IRC)-channel. Usually some of the core developers are logged in, and for urgent and apparently easy-to-answer questions, this might the best place to ask. The chat room also appears to be very informal, and people do not hesitate to ask seemingly too small questions. On the other hand, the informality leads to a lack of initiative and to no one feeling obligated to answer your request. The activity is therefore close to zero, and a look at the log underpins this - the following excerpt from the log of December 11th was the only human activity this day:

```
13:24:27 *** user has joined #openmrs
13:24:42 <user> hello everybody
13:25:01 <user> i need a help to install openmrs codebase
13:25:23 <user> can anybody help me
13:27:02 <user> can anybody help me
13:28:40 *** user has quit IRC
```

Events

Since 2006 OpenMRS has arranged a series of workshops and conferences, the largest one being the annual Implementers Network Conference. A paper from 2009 on OpenMRS Implementers Network states [Seebregts et al., 2009] that:

> *"Although internet-based collaboration tools have proven to be highly effective in supporting OpenMRS implementations, regular face-to-face meetings and training courses are fundamentally important to supporting this process. It seems unlikely that Open-*

> *MRS would have reached the same level of success in Africa without an annual meeting and training courses."*

As the community keeps growing, so does the frequency of events. They have also started to plan meetings in other regions besides South Africa and the US, where most of their activity has been focused so far.

Chapter 5

Development

In this chapter we will give an account of the fieldwork in respect to the development conducted in Shimla from the middle of October to end-November 2010, and the middle of february to the middle of april 2011. We participated in the project as system developers primarily with the goal of developing two modules for the HISP India OpenMRS project. In addition to our work on the project, we will also describe the work leading up to our first arrival, and the development methodology used. The overall timelines for our two stays, including the initial project startup, is illustrated in figure 5.1.

5.1 Initial Development

Development of the OpenMRS modules was initially started in May of 2010 by four developers located in Shimla. One of them had done some work with OpenMRS previously, although not to a large extent, and as such they had to figure out a lot of things on their own throughout the process. There was not a lot of communication with the people at OpenMRS which exacerbated

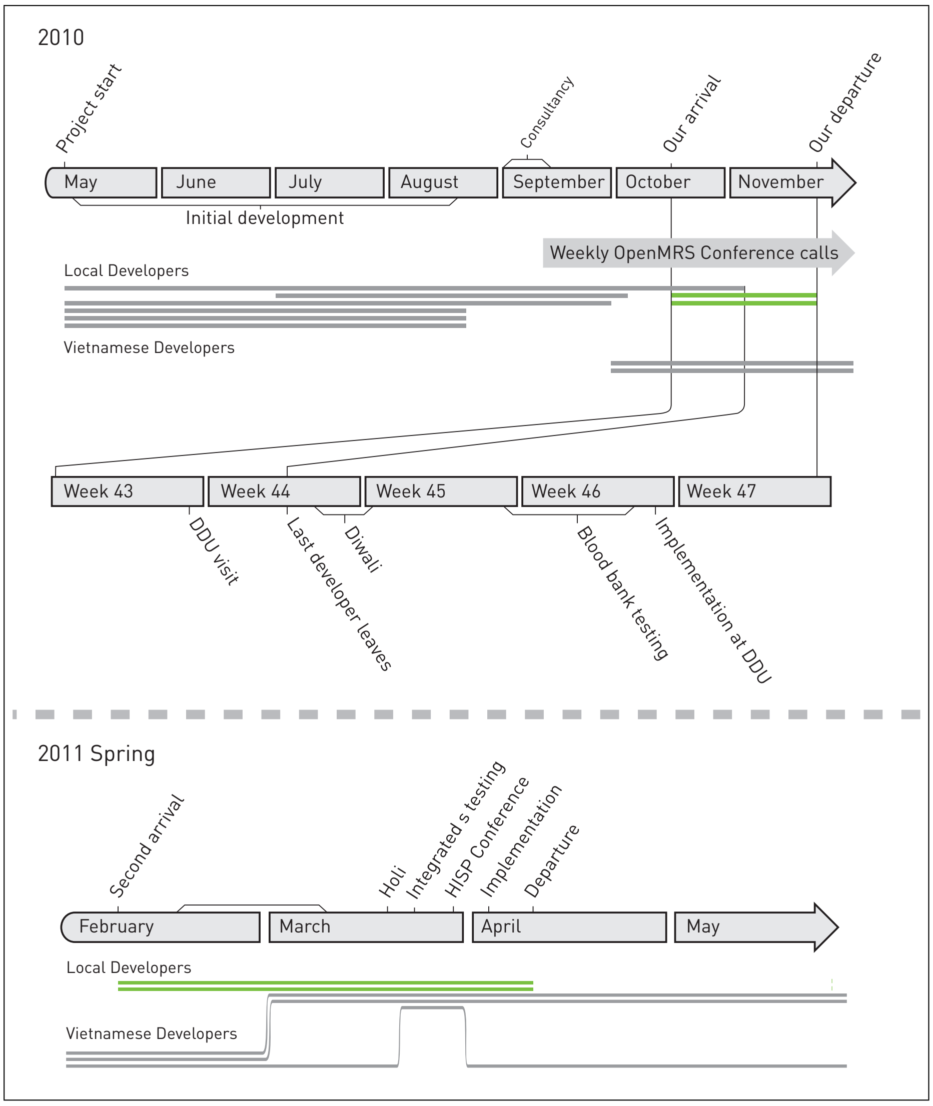


Figure 5.1: Project timeline

this fact. After laying the groundwork they continued producing a number of modules over the following months, amongst them registration, billing, inventory and pharmacy. There were a lot of developer changes throughout this period, with the result being that by the beginning of September only two developers were working on the project. Due to losing the developer with OpenMRS experience, they were also left without a link to the OpenMRS community. In an attempt to alleviate this they contacted OpenMRS centrally and inquired about possibilities for support. They were subsequently put in contact with an Indian company located in Bangalore which agreed to send a consultant to the Shimla office and discuss their design and modules with them. This visit turned out to be very useful, as it was quickly discovered that the team had fundamentally misunderstood the underlying architecture of the OpenMRS model, as well as the complexities and rigour required of a full hospital system. In addition there were found significant gaps between the requirements and the implementations. As a result of this, all modules had to be either scrapped or significantly rewritten.

> *Registration had to be redone completely, billing had to be redone completely, inventory had to be redone, and pharmacy; all had to be redone. (...) It was a huge resource cost... Very huge resource cost.* - Project leader

Development was restarted shortly after by the two developers remaining in Shimla. As a result of the previous problems, it was decided in discussions with the developers at OpenMRS that some guidance was needed. To help with this one, of the OpenMRS core developers offered to host weekly meetings to discuss the architecture and design of the system, and to answer any questions the team might have regarding OpenMRS.

Prior to our arrival, an e-mail was sent out to everyone associated with the project to introduce the people involved. Two Ethiopian developers and a Vietnamese consultant were introduced as starting development on the project along with us. Due to reasons unknown, this never happened, and we ended up being the only new developers on the project.

5.2 Development Handover

When we arrived in Shimla the development was headed by one developer locally, and one developer who had left a week before we arrived and was working remotely. A third developer working remotely from Vietnam had joined the project shortly before. At that point development of registration and billing was wrapping up, while development of inventory, blood bank and the RKS finance module was starting up.

The development of RKS and the blood bank was lead by the local developer, and had been started shortly before our arrival. Since he was the only developer on site we were put under his tutelage. After having stayed six months in Shimla, he was eager to leave, and as such was attempting to wrap up his obligations. He helped us set up the development environments and gave us the documentation he had, but mostly left us to our own devices unless explicitly asked. Being as things usually are, there was a certain amount of learning and setup involved, so while he continued his work on the blood bank we were assigned to familiarise ourselves with the system.

For the remainder of his time in Shimla, the local developer continued working on the RKS module, blood bank module and a security module, primarily coding but also occasionally instructing us to the point where he felt the responsibility for the team could be passed on to us. By the time he left, all modules were more or less operational, though lacking any kind of testing which he considered “the implementer’s job”. There was a lot of skepticism and attempts at discouragement when he finally did leave, which came as little surprise considering he was the only senior developer on site, but assurances were made that we had any and all information we needed, and would be able to finish development.

5.3 Development Process

The development process used can perhaps best be described as a sort of waterfall. Requirements for the different modules were initially collected, discussed, and agreed upon in cooperation with the state government. After requirements were finalised, the design for the system was discussed and agreed upon internally before actual development was started by the responsible developer. While there were internal discussions on each module, a lot of the responsibility for design and planning ended up with the responsible developer, as the implementation team had little experience and were unable to provide much in the way of criticism.

While there was no use of any formal development methods, this had not been the intention from the start. The project leader in an interview revealed that an iterative agile-style development methodology had been both recommended by OpenMRS developers, as well as discussed and agreed upon locally during the planning phase of the project. Despite these plans an iterative process was never used, and the team instead ended up developing systems in solid blocks, with bug and conformance testing being performed before implementation at the hospital. This was a cited as a critical factor in the initial development failure due to significant deficiencies not being discovered until fixing them was too costly. This was less of an issue while we were there, but the testing and implementation period still bore marks of being very hasty.

One notable incident occurred during the development of the blood bank when a security issue was discovered. Both the system architect and the OpenMRS developer agreed that this was a fundamental issue since it would expose confidential blood testing data to non-blood bank personnel. It was quickly decided that the local developer would create a module to fix the issue, and this was done in about a day just before he left the project. During the next meeting when the module was discussed, this was a point of concern for the architect who felt that one day would not be enough to produce a fully secure solution, and that the module could not be trusted without further developer testing. Therefore it was decided that the remote developer and system architect would do testing and report back for next week's meeting. The next week this had not been done due to a lack of a working copy with which they could test. This quickly derailed the conversation onto other issues, and nothing further was decided on the issue of the module. This issue was not revisited until two weeks after we had left Shimla, at which point the blood bank was already operational at the hospital, despite the security issue having been labeled as a showstopper bug.

Another problem during development was the inherent complexity of the system. Due to this, a substantial part of our stay was spent making ourselves familiar with the system and the different components and APIs. Since the trainee developer was in the same situation, she was assigned to work with us. This proved difficult as she was far less experienced than we had initially assumed from her introduction mail:

From: trainee software developer
(...)I am just new with Hibernate and Spring so taking training from [lead developer]. So this shows my weekness and need more experience on this. I am trying that I will cover these in few days and start working on it ASP.

After discussing this with the senior developer who had attempted to mentor her for the past month, it quickly became clear that she was lacking fundamental programming experience, and the lack of experience would make it very hard for her to keep up. Considering we were struggling quite a bit ourselves, we ended up unable to provide much in the way of guidance, and she was not able to help with actual programming while we were there.

5.4 Hospital Core Module

Right before we left Shimla the first time, concerns about modules' need to communicate with each other, especially between Billing and others that were going to need Billing's services in order to properly work. The Core Developer revealed under one of the weekly Skype meetings that OpenMRS had poor support for this, and that it would be harder than the local developers had first thought. It turned out that modules could not use each others' Application Programming Interface (API) without an explicitly declared dependency, which made many of the planned features impossible as the modules were cross-dependant on each other. The system architect did not like these news, as he expressed himself in an email about this addressing the core developer:

> *At one point during the conversation last week you mentioned that modules only had access to one another's api if they had an explicit dependency declared. This is a an important piece of architectural information which should be pretty obvious but which I hadn't been aware of.*

Soon after the issue became apparent, the architect came up with a solution by making a module that would "wrap" and combine the module's APIs, much like an extension to OpenMRS' core. Adapted from his proposal we have made the following two figures(5.2) to illustrate how it should work (though simplified, and not correctly in respect to the real system, they show the concept):

All the team members thought this seemed like a good idea, and even the OpenMRS core developer was positive. It was added to the weekly OpenMRS core developer meeting (not the same as the project's weekly meeting), which the system architect and the senior vietnamese developer attended, to

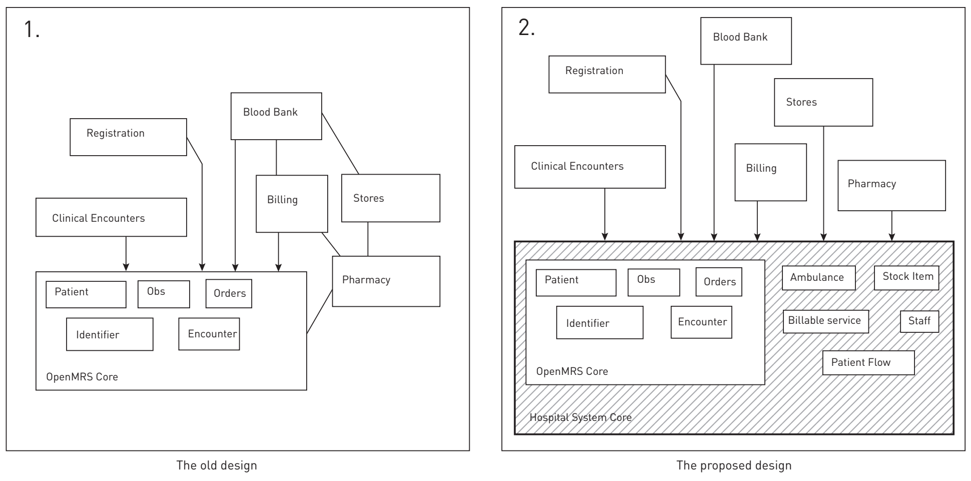


Figure 5.2: Proposed solution

explain their plan. Under the meeting the rest of the core developers seem to have agreed, and they came up with a reasonable approach which was to start by bundling the relevant modules' data module into a single module, and later on split out services and methods as it matured. The Vietnamese developer became the responsible developer, and started the development in late December. About mid February the first version was tested in Shimla. The hospital core module made the other dependent modules useless without all of them being deployed as it contained all the business logic for the modules. They also had to be rewritten substantially since the whole data model moved out, so it had been a quite big refactoring of the whole system. Around end March the first version of the system with the hospital core module was installed at DDU.

When discussing the module in May at the conference, there was a lot of concern about an overly large amount of logic being moved into the hospital core module. The feeling among many of the team members was that it destroyed modularisation and made the core module too volatile as it had to be updated every time a new database function was required. No decision regarding this was reached at the time, but in a later interview with the responsible developer he could reveal that rewrites for the core were planned, and that a lot of the business logic would be moved back to the original modules, thereby making the core less volatile. The hope is that by retaining only the methods that are strictly necessary for communication between modules, a much more flexible architecture can be achieved.

5.5 The Blood Bank Module

A blood bank is a storage compartment for blood, normally at a hospital, where blood is collected and preserved for later transfusion. The blood comes from voluntary donations or by other collection methods, such as blood donation camps. There are eight different types of blood, with different characteristics that make many of them incompatible with each other, meaning that you can only receive certain blood type(s), depending on your own. A number of deadly or serious injurious diseases can be transmitted via blood, the most well-known being HIV and viral hepatitis. Blood can also only be used within a limited period of time before it expires. Because of these reasons, it is crucial that the donation, storage and transfusion of blood is highly controlled. An electronic blood bank system can be of good help in minimising these risks.

At the time of our first arrival, requirements and use cases for the Blood Bank had been gathered from the hospital. The documentation consisted of requirements and use cases, and was seemingly complete and highly detailed. The functionalities proposed for the blood bank system can be summed up as:

- Add/register blood donor
- Add donor data from questionnaire

- Assign blood tests of collected blood unit
- Register blood test results
- Update the blood bank stock
- Issue blood units from the stock
- Pre-generated blood donor IDs

Blood donors are treated as patients in the system, but they are also given a unique blood donor ID in addition to the patient ID. A requirement related to this, is the ability to pre-generate donor IDs before registering the donor. These pre-generated IDs are to be used in blood camps, which is the most common method for collecting blood at the hospital. The blood camps take place off-site, thus they will not have access to the system or any possibility to register donors on the fly.

As the basic functionality of the module had been implemented by the senior developer, bringing a first version to completion was done without any major issues. It was successfully tested and implemented at the hospital before we left Shimla in November. There were no available developers to take over the development, and a few weeks after we left, the module stopped working at the hospital. We tried to help them through e-mail, but the technical complexity of the problem made us unable to fix the problem remotely back in Norway. Therefore, the module was given a low priority by the team and left unfixed. There was a lot of fixing to be done when we returned in February, which led the module to stay unimplemented until the end of our second stay. A lot of technical and architectural changes were done to the overall system, and a substantial part of the time during our second stay was spent refactoring, instead of further development and implementation of new functionalities. As expected, due to the short uptime, there waas no additional feedback from the hospital workers since last time. If a local developer had had the time to take on the development, and fixed pressing issues between our stays, a lot of time would probably have been saved.

5.6 The RKS finance module

RKS, or “Patient Welfare Committee” in English, is a management structure used in many states in India to increase the financial self governance of local hospitals. These committees, consisting of government officials, members of associated NGOs, elected representatives, and others, are free to set user charges for patients and use the resulting revenue to buy supplies, equipment, pay salaries, and conduct repairs. In practical terms the DDU collects money from user charges relating to patient procedures, rent for hospital buildings like the pharmacy, interest payments, ambulance charges, etc.

At its heart, the RKS module is a financial planning and analysis system for these committees, as well as the state government. Currently the hospital’s daily and yearly income and expenses are tallied on paper, and transferred into Excel for calculation. The goal of the module is to transition away from this paper-and-excel hybrid system, and replace it with a fully automated system which can give immediate breakdowns of earnings, in-depth overviews of expenses in the different departments of the hospital, as well as budgeting tools. In other words, a very lightweight accounting system tailored for RKS. The use of the billing and inventory modules means a large amount of financial data is already available in the system, ready to be used.

The plans for the module were already laid down by the time we arrived in Shimla. The lead programmer had designed a solution using Palo Server, an OLAP-database with the ability to pivot, aggregate and select numbers in different dimensions, allowing users to easily present breakdowns in terms of days, months, departments, or other relevant data. Palo uses heavily programmed Excel sheets as input and output front-ends, which was part of the reason for choosing to use it. Currently the clerks at DDU create and use Excel sheets to make accounting sheets, and budgets.

After evaluating the requirements that could be found, we concluded that

Palo was needlessly large and complex, and that a solution made by hand could achieve the same functionality, while being better suited for the hospital environment. This work was, however, started fairly late due to planning issues, and was not finished in time for our departure from Shimla.

After returning to Shimla in February, it was decided that since the previous version had not been completed, it would be wiser to continue ahead with the originally planned Palo version. Work on this version continued for the duration of the stay, but due to the complexity of communcation between OpenMRS and Palo, neither this version was completed. This proved to be a big disappointment both for us and the team, as HISP had promised its delivery along with the other modules.

During our final interview with the senior developer, the responsiblity for the RKS module was planned for some newly hired developers. As none of the other developers were particularly knowledgable about Palo and RKS, they were unsure about which path to choose. The feeling from the senior developer was that Palo might be too heavy, and that a lighter homemade solution might be easier to create as it would be quite similar to the other modules created, and not involve a seperate system and API. This was, however, not decided upon at that point, and discussions will likely continue.

Chapter 6

Implementation

In this chapter we will give an account of the implementation work during our two stays in Shimla. During both visits the majority of team members were working on implementing the system at DDU. Although we have not taken part in the implementation ourselves, we will try to give the best possible picture of it based on our interaction with the implementers and hospital workers, as it comprises a part of the project that in many ways is as important as the development.

6.1 Progress

Since the system is module-based, the implementation could start gradually by one module at a time. The plan has been to do the implementation of modules incrementally along with the development, starting with the parts that seemed easiest both technically and implementation-wise. In september they started installing registration and billing at each respective counter. Both modules were among a few that the team had started development of in May, but these two were the only that turned out to be usable. The rest had to be completely remade. Technically, the registration module was an extension of the patient registration functionality OpenMRS already had, but the billing module was built from scratch. They were also good starting points seen from an implementation perspective. Both departments were simple with one or two clerks at each counter registering patients or handling payments. This meant that you did not have to train a great number of personnel, and the tasks done were easy with few possible use cases. Starting with patient registration in this early stage was also helpful, testing how the system's performance would develop over time as well as getting a patient data foundation to use in relation to other modules' development.

After these two, implementation of blood bank, laboratory and OPD followed, and by the end of November these were installed and the training of personnel at had started. When we returned in February, IPD, inventory and pharmacy modules were also implemented. Except from blood bank, the first implemented modules had been altered since last time. The implementers seemed to have put in place good routines for training. Depending on a module's complexity and the number of workers that were going to use it, the implementers trained them differently. For billing and registration there were only a couple of clerks that would need training, while for IPD a number of nurses were the users. For larger numbers of users they would divide them into groups and conduct intensive training lessons. By now all the workers at DDU have received between 20 and 30 hours of training in total.

6.2 Reception

During our stays we have visited the hospital a couple of times and had the opportunity to see the implemented modules in use at the actual work places.

6.2.1 OPD

The most complex module to implement seems to be the OPD module. Even though all the physicians been given a substantial amount of training by now, and are proven capable of using the module, most of them are reluctant. Time is a key factor here. As all the out patient-departments are crowded with patients, especially in the morning, they see it as a big hinderance to waste time by registering patient data on a computer. Especially as they still have to write symptoms and drug prescriptions on patients' paper slips in addition. Many physicians think that a clerk should be there to do the computer work. Many of the team's implementation workers have been spending a lot of time during work hours doing so. Partly as a testing effort, partly for additional demonstration with the aim of teaching the physicians how it is done. The physicians are nevertheless positive to the system, and think it can be helpful and benefit their work.

There is one OPD unit where implementation has gone better than the rest, at the medical OPD. This seems to be due to its lower patient load, and the fact that there is only one physician working here. Another small, but noteworthy difference is that the physician here has placed the computer right in front of himself on the desk. Like in picture 6.2, the other OPDs have turned the computer away from their working area. One can wonder if the computer has been moved after attempting to use the system, or if the system is not used because the computer is turned away. The answer is probably a combination.

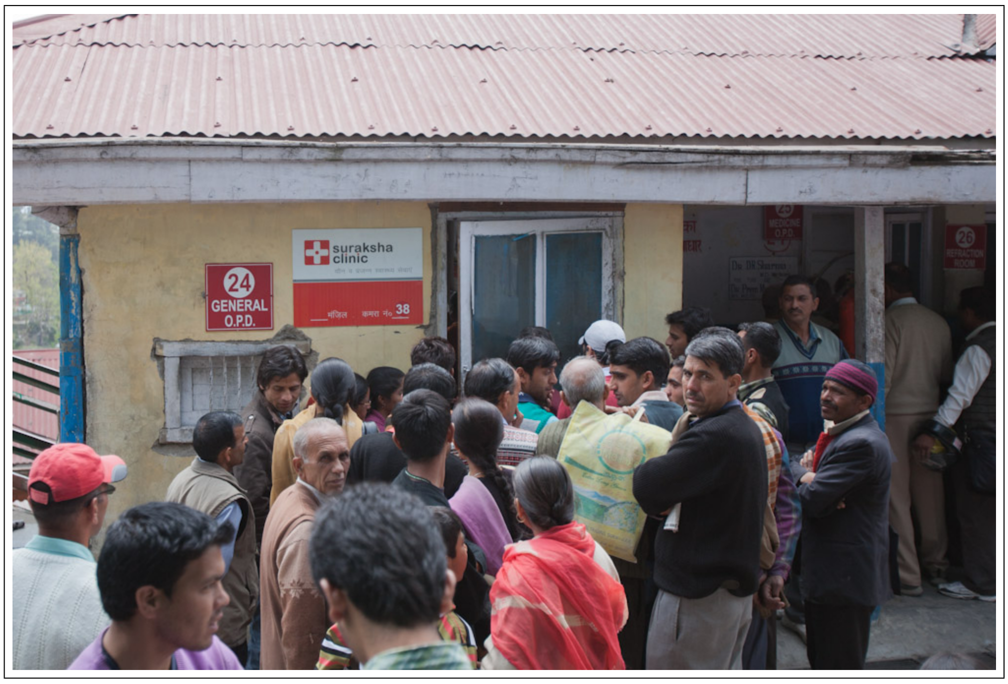


Figure 6.1: A normal queue of patients waiting to get in to OPD

6.2.2 IPD

The IPD module is one of the newest and smallest modules, at least in a functionality perspective. The nurses who will use it seem unsatisfied with its capabilities. They feel limited, they "can't even see the numbers of bed available at the ward". Before the module gets more functionality, it is unlikely to be used extensively. The upshot is that they seem to have a good understanding of how to use it, and will probably adapt easily to newer versions as the module mellows.

6.2.3 Blood bank

On our visit to the blood bank at the end of March, the system was not in use, and from our observations it seems doubtful that it has ever been used

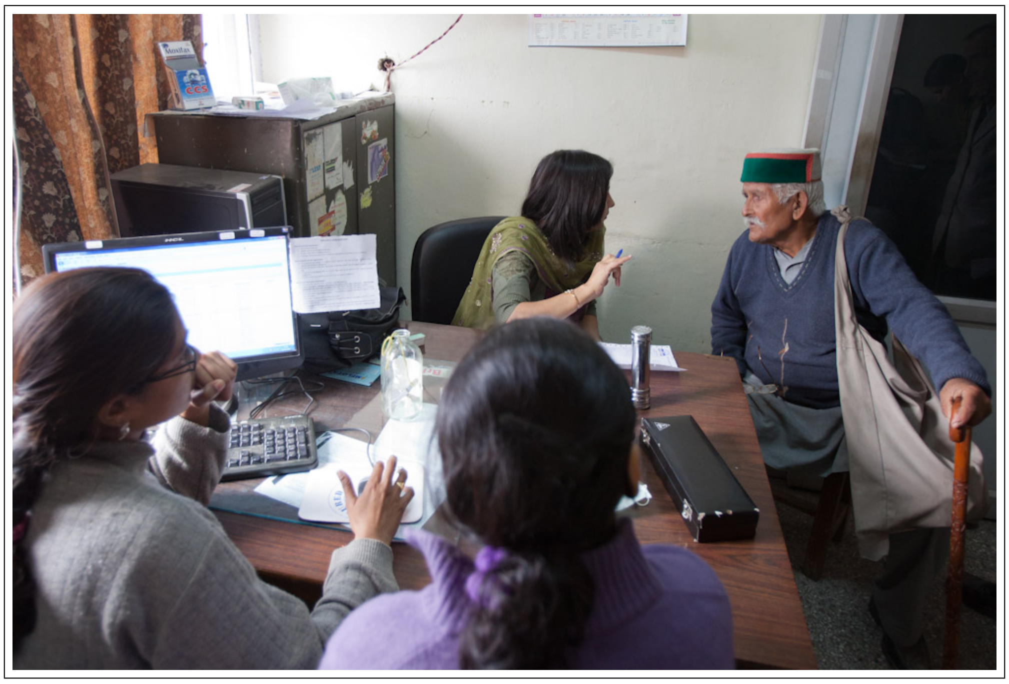


Figure 6.2: Two team members using the system while the physician is helping the patient at the eye OPD clinic.

after the initial installation. The workers are obligated to use a paper-based record for both donation and issuing of blood. As long as they have to do this, they wont use the module since it means twice as much work. Apart from this, there does not seem to be much else that keeps them from using it. This module has seemingly not been much prioritised by the implementation team so far. As were the responsible developers of it, the module have not been altered since in the fall It is also not connected to the rest of the system for the moment. [KOmmentar: SKRIVE LITT MEIR HER]

Figure 6.3: A nurse at one of the wards talking about the IPD module

Chapter 7

Arenas of Communication

One of the biggest challenges in a distributed development project, is communication. Ensuring that information is shared equally, and that development is properly coordinated can be difficult under the best of circumstances. When you are dealing with actors spanning five different locations with up to 14 hour time differences, however, this takes on a completely different dimension.

In this chapter we will look at the different forms and arenas of communication used within the project, and their impact on the development process, both negative and positive.

7.1 Face to Face

As we and the entire implementation team was located in Shimla, face-to-face communication was a big part of how we learnt about the project. Especially the senior developer who was on site for the two first weeks, was a big help in getting us acquainted with OpenMRS and the system as a whole.

The work environment and structure, while quite friendly and informal, was very different from what we were used to. Shouting was frequently used when talking to subordinates or peers, whether in person or on the phone. This was very alien and awkward for us at first, though nobody seemed to harbour any grudges and usually settled their arguments quite quickly. There were also a lot of complaints, even from the project manager, though it was rare for anything to be done about it.

Towards the end of our stay there was a more serious conflict about the distinction between patients and donors in the blood bank. It had at one point, before our arrival, been decided that donors should also be patients to simplify the development. During testing the health information officer discovered this, and in discussions with the other implementers rather vocally disagreed with it. He was concerned that on semantic grounds this was nonsensical, and would cause bloat in the database as donors who never set foot within the hospital would permanently be entered into the patient records. Due to the development process this was discussed shortly before implementation was to take place, meaning there was no time to make the change even if one had wanted to. The project leader disagreed however, and the argument continued back and forth for days. In the end nothing came out of it, though the health information officer shared with us his grave disappointment at not being listened to in what was his area of expertise.

The situation was quite different when it came to technical issues. Due to a low understanding from the implementers, what the developers decided to do was usually accepted without complaints. This proved somewhat frustrating since by the time the senior developer had left we were left without any local 'community-of-practice' [Walsham, 2001a] and, by this, anyone with whom we could easily confer when we were stuck or had questions about the system. We were advised to keep in close touch with the remote developers, but this proved less than satisfying due to the latency and the differing work hours.

Another problematic issue which occurred was the decision of how to implement the RKS module. Both we and many of the senior HISP staff were sceptical of the senior developer's choice of using Palo. It required a separate server instance and the use of Excel, and it was unclear why this solution was especially well suited. He happily demoed the application, but when queried whether it really was the best solution, he would often simply say "It's used by lots of fortune five hundred companies, just check their homepage.". While the project leader trusted his word that it could be done, she often jokingly pleaded to us to "please don't use the Palo". We attempted multiple times to discuss the issue with the senior developer, but were usually met with tired exasperation as he simply wanted to follow his plans and finish up.

In the end, after the senior developer left, we convinced the HISP and project leaders that an equally viable solution could be produced in a simpler fashion, and got the go-ahead to plan such an implementation. However, after developing a model and starting discussion of it with the system architect, it was quickly realised that fundamental requirements documents existed which we had never been provided with or seen. These had been placed on Redmine, but had gone unnoticed. After reading the full requirements it quickly became evident why the senior developer had chosen to go with Palo, as the requirements would be a lot more work to implement manually than initially assumed. This fundamental breakdown in communication caused the module development to go way over schedule, and we were unable to finish it by the time we left Shimla.

7.2 Teleconferencing

Due to the distributed nature of the project, Skype, as well as mobile phones, was often used when discussing issues with remote actors. This included everything from regularly scheduled meetings, to bug reports and status updates. Skype was used most of the time internally in the project as many of the actors were located in different countries, making regular phone calls ill suited.

Perhaps the most important use of Skype was for the weekly developer call between the development team, the lead implementers, the head of HISP India, and a member of the OpenMRS core developer team. This meeting was used to discuss progress of module development, any issues that might need resolving, and planning for future development. Due to the large time differences, the meeting was usually held during the evening Indian time, which was early morning US. This in itself was a source of some annoyance, though nothing much could be done about it.

The meetings were useful in many ways. It was more or less the only time the entire team was "together", and able to discuss matters and progress, not to mention that the OpenMRS developer was able to give valuable input on plans and decisions. Because of this, most development decisions were made at these meetings. There were, however, a lot of problems running these meetings efficiently. A frequent problem was the lack of a meeting agenda, with the side effect that tangential, and often inconsequential, issues took up a lot of time. This was a frequent complaint from most of the people in the call, but in the end little was done about it. This lack of planning also resulted in one meeting having to change the topics significantly due to one of the key people unexpectedly not joining, as well as confusion around the starting time of another meeting with the result that the Shimla office had to sit around for an hour waiting for the meeting.

Besides these organisational issues there were also problems related to the audio quality, as well as some of the participants' English. Audio quality was often bad due to the conference service used. Some of the participants were also not great English speakers. This resulted in misunderstandings and often hesitance on their part. In one meeting, while we were attempting to explain an issue to the architect and the OpenMRS developer, we were interrupted by the project leader who wanted to explain “what I think [he] is trying to say is...”, which turned out to be completely mistaken, and thus derailed the conversation.

In the end, the result was that the majority of the talking was mostly done by three people, with occasional comments by others when they were explicitly asked.

7.3 E-mail

E-mail was in many ways the primary way of keeping in touch. Project announcements, milestones, and development builds were usually distributed in this fashion. When we first arrived in Shimla, most information was distributed by lengthy CC lists in email. As a result, potentially important mails were as a result frequently only sent to a handful of project members, with subsequent confusion when some people had quite literally not received the memo. As an example, the initial introduction mail to the project started out with five recipients and five copies. Over the course of two days, an additional six copies were added which ended up in different branches of the thread, resulting in later copies not getting the rest of the thread as people replied “incorrectly”. This was to some extent alleviated when a mailing list was set up for the project after a few weeks. This list was used extensively throughout our stay, though primarily by and for the development team. There were, however, still frequent misunderstandings:

From: Vietnam developer

Hi, I uploaded the database dump file which I got from [sysadmin] on 13 Nov http://.... This is the database that I am using for development. If we have a newer version please upload it there.

From: Samson

After a quick look it seems to be dumped for patient data, which is good. Nevertheless, I think it is not a good idea to upload the whole db onto redmine, especially with no user restriction(I downloaded the file without logging in).

From: Project leader

Why on earth have we uploaded the patient database on redmine who needs the database and for what? if anyone needs database pls get in touch with (sysadmin). and only database we can share is the testing database (used for application testing in our office). please do not ask for hospital db... we are not authorised to take or share it have removed the db from redmine. and (sysadmin) pls cross check before sending db

From: sysadmin

Please do not upload database on redmine if anyone wants anythings let me know.

From: Vietnam developer

That is a test database, no real patient data, which i am using for development, I see no issue of uploading a development database for other developer to work on it. Samson did you really find real patient data in that database

From: System architect

I asked [developer] to upload the database to redmine. And of course without patient data. In fact I do hope that (developer) doesn't have any patient data. We HAVE to be able to share this database. (...)
Now it is gone! Will you please check that there is no real patient data and put it back soonest. I have been asking for this for some weeks now.

The style of writing emails was quite different within the project. Especially the system architect had a habit of writing long mails, which caused some friction with the local team. There was a feeling that the emails were often both unnecessary and disruptive, and while they were read, the content was frequently ignored even though the architect was technically in charge. There was also a certain amount of indignation involved due to what was considered nitpicking from someone who didn't do any coding themselves. While this was never explicitly voiced in writing, it did result in some passive aggressive replies and snide comments from the senior developer: "Ah, [architect] has sent a long email again! That's the only thing he can do. He know shit!". While the project manager didn't directly approve of this, she too would often comment on this when they were on Skype. Much the same indignation was directed to the OpenMRS developers.

> From: OpenMRS core developer
>
> (System architect), your emails are long. :-) Here are my responses–I hope I cover everything. [...] 4. Restrict by Encounter module (i.e. what to do) I agree that building a Restrict by Encounter module is a fundamental key step. [Local developer], I would recommend that you sketch out the design you're proposing, and email this to the developers@openmrs.org mailing list for feedback. We also have a "Design Review" call on Wednesday 9am Eastern where it would be appropriate to get feedback on this module from others (especially including [Main OpenMRS developers]).

The senior developer never replied to this, but was very clear that he was unwilling to *"waste my time listening to them for two hours only to talk for ten"* on the OpenMRS call, despite this being the head developers of OpenMRS who would be able to provide valuable input.

7.4 Redmine

Redmine is one of the tools deployed to ease collaboration in the team. Redmine is "a flexible project management web application" [1], with support for wikis, forums, roadmaps, source code management, and bug tracking, among other things. When we arrived in Shimla, Redmine was a relatively newly implemented system, intended to clear up a lot of confusion around documents, and make it easier to distribute and disseminate information. The transfer of information was in many ways in its infancy, with only a small number of documents having been uploaded to the repository.

[1]www.redmine.org, (June 11, 2011)

One of the main problems with Redmine was that the installation had performance issues which rendered it more or less useless. It could take minutes for the page to load, if it did not time out. Everybody knew about this, but nobody did anything further about it than complain informally. As time went by and the performance issues did not disappear, people started ignoring it. When they were told to put documents up on the site, they did not answer, or circulated the files on email or Skype. In the cases where things were uploaded to Redmine, they often went unnoticed by the people for whom they were intended, with similar results of bewilderment when the documents were referred to. It was also the intention to use Redmine for bug tracking, but a shared Google Docs spreadsheet was used instead.

Nothing was done about the performance issues until the system architect expressed his concerns by e-email:

> From: System architect
>
> I am aware that the redmine server seems still to be very slow. [sysadmin], can you please advise what to do about this? The issue has been raised repeatedly. If it is for reasons beyond your control (eg. bandwidth of link to your server) then we must consider moving this to a commercial hosting. It's really not that expensive. [project manager]? US$9 per month?
> Meanwhile we must persist ..

This got the project manager's attention, which shortly thereafter led to the system administrator fixing the problem.

> From: Project Manager
>
> (. . .)if we need new site to host this redmine, even if it means more money, let us do it..as is crucial for us

When the performance problems were fixed, people started using it more frequently, but for bug tracking people continued to use the Google spreadsheet as a draft for the bugs, and when the testing was done they moved the findings over to the Redmine bug tracking system. Despite the cumbersome process of registering bugs both places, people tolerated this. They seemed to be very satisfied with the Google spreadsheet, as they felt this had less formality than Redmine. On the other hand, the formality was one of the benefits of Redmine. If you reported a bug on Redmine, it was assured that it was properly tested. The system also sent out a mail to every member attached to that project.

This confusion around document management did unfortunately not change in a significant degree while we were on site.

Chapter 8

Discussion

In this chapter we will analyse some of the challenges faced throughout the development and implementation process of the project, and attmept to answer the research questions we posed in the beginning of the thesis.

8.1 Distributed Development

The challenges of distributed development present themselves in many ways. In our case large parts of the development process was conducted in different countries, and coordinated via mail and telephone. In such a process there are a number of things which can go wrong or be confused, both in terms of implementation, testing and the actual development. It places a large requirement on both the diligence and the communication skills of the different parties involved, and can, if underestimated, result in both slowdowns and direct failures. The gains are, however, often significant, in that you can leverage resources and expertise which would otherwise be unavailable, and as such it can often worth the risk. In our case, an already tricky situation was made even more difficult due to deadlines and a limited amount of experience with the software products used, and as such the complexity was raised significantly.

8.1.1 Project Obstacles

One of the earliest and biggest obstacles we encountered was difficulty of coordinating work across timezones. The project architect, who was intended to be the technical lead, was working remotely and often not able to keep up what the other developers were doing on a daily basis. Development for each module was delegated to one developer who was in charge of making sure that it proceeded smoothly. Information about the development progress was largely restricted to the developer responsible, and as such it was difficult for others to keep updated and provide feedback on the development process. This problem became very evident during our first stay in Shimla, during the development of a security module for the Blood Bank. Due to the inherent structure of the OpenMRS system, information about blood tests entered into the blood bank, which could potentially contain classified information such as the result of AIDS tests, would become available to a large amount of staff with generic system privileges. This was seen as a show-stopping problem for blood bank module, and responsibility was assigned to the lead developer. The development of this module proceeded swiftly before his departure from project, and what he assured was a working solution was delivered in code form. In subsequent meetings after his departure, the subject of this module was discussed, and it quickly became evident that nobody besides the original developer had verified its functionality, or indeed attempted to run it. The architect also expressed significant doubts as to whether the module would provide secure enough for the purpose.

> *I have serious doubts about whether [the developer] would have had time to write a fully secure module in only one day. [...] I*

think we need to do a thorough code review before we can use it.

Later attempts to run it showed that the module did indeed seemingly work, but code-level inspection of the robustness of the module was never performed, despite stated plans to do so at the weekly meeting. Prior to the development of the module, there were discussions with the OpenMRS developers about the feasibility of such a module, and a general approach was more or less agreed on. The lack of documentation, however, more or less doomed the module since it proved hard for others to both use and understand the module, and the other developers were hesitant to take responsibility of it. In the end the module was never deployed, despite it initially having been labeled as critical.

This particular case, which was in no way unique, highlights some of the intra-team problems of communication that arose during our initial stay in Shimla. The architect had significant problems communicating with the other developers, and was by many seen as both overly concerned with design, as well as insignificant due to the fact that he did not participate in the actual development of the system. The architect later ended up leaving the project, and while we can only speculate as to his reasons for doing so, there is little doubt that he had significant problems communicating with the rest of the team. Such a problem is in no way uncommon in distributed situations, and especially for leaders it can be challenging to retain influence and respect when you are not able to interact with the team directly.

For us as junior developers on the team the distance also proved to be problematic in that after the lead developers stay in Shimla ended, we were left without a "community of practice" or any other developers to confer with. This proved to be highly problematic as we were in many cases not sufficiently comfortable with the system to make decisions about ways to develop our modules, as well as ending up spending significant amounts of time on relatively trivial problems due to a lack of mentors with with to discuss.

Attempts were made to improve communication with both the vietnamese developers as well as the architect, but in the end the latency of such communication slowed us down greatly. Especially the initial development of RKS was very delayed due to the complexity of the system, and our inexperience with the plans the lead developers had laid out.

Documentation was also a significant problem throughout our stay. While Redmine had been planned as a collaboration platform initially, this did not happen for a long while due to server issues which rendered it exceedingly slow. After complaints during one weekly meeting, and a somewhat strained email thread, the problem was resolved. Despite this, however, Redmine did not see significant use, and a lot of documents were distributed solely by mail. The architect made repeated calls to use Redmine, but there were few changes in the use among the implementors. By the end of our stay is was more or less only used as a repository for builds of the different modules. This lack of knowledge sharing was especially evident during the RKS handover during which a lot of time was wasted on what turned out to be only parts of the actual requirements. The actual requirements document had been uploaded to Redmine, but despite asking for it were not made aware of its existence until later discussions with the architect regarding module features.

8.1.2 Theoretical Analysis

To understand the project actors it is perhaps more useful to look at them from a background and point-of-view perspective. Without a doubt the biggest differences between them lie in the differing national backgrounds, as well as technical experience, and project goals. With this in mind some very distinct groupings are visible.

First and foremost the different actors involved in the project during our stay came from very different backgrounds. While all of the local implementors were native Indians, who had previously worked with HIS in academia or in government, all members of the development team were foreigners, both from other parts of Asia as well as Europe. While this isn't in and of itself a problem, it does cause very different views on how the development and implementation process should best be conducted. The Irish architect for instance, when visiting the local premises, was shocked to see that what was termed the "server-room" was little more than a padlocked closet with a window. Without such local background knowledge it is impossible to make well-informed decisions about the directions to take during the developent process. This in turn means that a close dialog with the local implementors and the rest of the team is necessary to ensure that all participants of the process are on the same page with regards to the premises and the goals of the project.

This also applies to the technical backgrounds of the actors. The implementation team was largely from Health Information backgrounds, and with some exceptions had little technical background in the field of IS development. On the other end of the spectrum, the architect, and in many ways us, come from an academic background and have a broader interest in the product beyond just the end product. This difference in views became very evident throughout the development process in that the architect would write long mails on the subject of software modularity and security, which would largely be ignored by the local team because they often either didn't understand the problem, or see it as a problem. One particular example is the discussion around data sharing. The practice prior to our arrival in Shimla was distribution of a database which contained all the data that was required to run the system. This database was both brittle and confusing as it contained snapshots of a running system, and as such was dependant on running the correct versions of the modules, whichever version that may be. Many of these databases also contained live data from the hospital, in strict violation of guidelines. The data issue was quickly solved through the use an OpenMRS data export module which could efficiently export and import specific chunks of data, streamlining the process of distributing necessary data.

The architect was very enthusiastic about this since he was having problems keeping a working and stable copy of the system running for his own testing purposes. The implementors, however, didn't really see the use of this, and were fairly dismissive. The other developers, while more understanding of the problem, had little use for such a solution since they already had working systems up and running, and had no problems with the current deployment process. Because of this, the alternative method of exporting data quickly fell by the roadside, and was never used beyond some minor testing done by us.

In many ways, the most direct result of the differing backgrounds was evident in the views on the process and the goal. While it for us was an opportunity to both learn and to contribute to OSS, the prevailing view among the implementors and many of the developers was very focused on "getting it done", in many cases with little regard for the maintainability and sustainability of the solution. They were under strict contractual obligations to deliver a certain amount of modules by a certain date, and failures to do so would cause problems both within the organization and with the relationship to the state and the hospital. Considering these diverging views on what the purpose and the goal of the project would be, it comes as little surprise that there were significant issues with the communication between the different groups. Ensuring that everyone involved are on the same page with regards to project goals is important to make sure everyone pulls in the same direction, but can at the same time be quite difficult when the cultural gap is similarly large.

According to Orlikowski and Gash[Orlikowski and Gash, 1994]:

> *The frames of reference held by organizational members are implicit guidelines that serve to organize and shape their interpretations of events and organizational phenomena and give these meaning.*

Ågerfalk et.al. [Agerfalk et al., 2005] add:

> *Culture can have a huge effect on how people interpret a certain situation, and how they react to it. Hence, having shared (or overlapping) frames of reference is a precondition for people to succeed in communication and collaboration.*

This view correlates with a lot of the experiences we've had while working on the project. The cultural differences in ways of discussing were immediately evident to us when we came there, a much stronger hierarchy and more confrontational style of communicating than we were used to. In addition to the strong divide between developers and implementors, the practice of assigning modules to individual developers made planning far more difficult than it should have been. Many of the things discussed require relatively high-bandwith communication to get across safely, and in many cases e-mail is not sufficient to fill this role. In the end, parts of the project failed, and a lot of time was wasted.

Practically all of issues encountered are known issues associated with distributed development [Sengupta et al., 2006]. It seems evident from our case, that these issues are a clear and present danger to any distributed project, and if not recognized early and handled properly, can cause significant problems.

8.2 Implementation

The primary motivation for the implementation of a new HIS from the state's side has always been an increased capability to track patients and diseases at a local level. Taking this a step further, and going for a full HospMIS in addition to Health Management Information System (HMIS) efforts (like the ongoing implementation of DHIS) might turn out to be a burger too big to handle. Arguably, hospital systems based on EMRs are, at least technically, more complex than HMIS. Since success in the HMIS domain has also been rather limited across the developing world, some may argue that maybe the district hospitals are not ready for a project like this. There are reasons to believe that there is much to learn from the literature and research on this subject. HospMIS' have been around in the west for quite a time, and the primary obstacle is not necessarily the technical part related to development, but largely in the socio-technical implementation challenges. In this chapter we will try to summarise what we believe are the biggest obstacles related to the implementation effort.

8.2.1 Patient load

One of the biggest differences between Indian and Norwegian hospitals is, as one might expect from the population density, the patient load. An Indian doctor can on an average day attend up to as many as 150 patients in a 6 hour day, many times the number that any western doctor would expect. This also means a proportionally shorter time with each patient (2.5 minutes per patient on average!), the direct consequence of which is that any extra time spent with a computer system compared to a paper based system, will be a very hard sell. Most of the doctors we talked to after the final installation were very clear on that they would rarely have time to use a computer system. Most of this was related to pure patient overload. There was, however, one exception to this. The doctor in the medical OPD had a lighter load, and was very positive to the use of the system.

While it's easy to claim that the patient load is too high to make a computer system viable, it should, however, be mentioned that the current implementation calls for a paper based system in addition to the computer based system. It is in out opinion likely that a full conversion to a computer based system would make it more viable. This is, however, easier said than done.

8.2.2 Inexperience

Another important difference between the west and India, though decreasing over the last years, is the so-called “digital divide”. While in Norway there is practically one computer per capita, there are in India about .05[1]. The majority of the health workers have very sparse experience with computers, especially the older ones, and can have a hard time understanding the basic concepts of how to use a computer, never mind use the computer system. This adds to the implementation challenge. The implementation of a Hosp-MIS affects most parts of a hospital. Maintaining an electronic record of a patient necessitates input in all stages of patient interaction, from check in, lab tests, medications, and to discharge. DDU is no exception to this. As described earlier, the patient goes from registration, through clinic checkups, and, if necessary, is checked in to a ward, or proceeds to billing to pay for procedures and lab tests. Both registration and billing have been computerised for some time, and while these were both part of the new implementation, the presence of trained clerks helps the further transition go smoother. Any hope that a successful implementation is to have, needs to be based on thorough training of the employees at the hospital.

8.2.3 Lack of motivation

Many of the health workers we talked to complained of a lack of functionality, as well as a lack of advantages for their working day. Much of this is related to the fact that many of the modules are in prototype stages, but there is little doubt that for some a computer system might not be a huge advantage.

[1]According to International Telecommunication Union, as of 2005 India had about 1.5 computers per 100 capita.

The blood bank is one of the less patient-congested departments, and one could believe that this lead them to be more adaptive to change. So far this has not been the case. This seems mainly to be due to their well-working paper-based routines that management seemingly force them to continue using. So far there also are very few benefits with the computer system compared to this. Before there is to be any hope that the new system will be used, it a) needs to fulfil all the criteria the management has, so they can cut out the paper record, and b) must have additional user friendly features that make it more attractive than the current paper-based system for the workers. The latter relates not only to this module - the nurses using the IPD module, for instance, were clearly dissatisfied with this module's present functionality. They missed everything from simple things like it being able to show information about beds available, to more complex things like administering patients and change patient data.

8.3 Standardisation

One of the most fundamental principles in the development of large IS projects is the use of standards. For different actors to be able to communicate efficiently and accurately, a set of shared standards need to be present at the bottom. [Braa et al., 2007] state that *"HIS standards have national importance and the role and involvement of health authorities will always be significant"*. It seems evident that HIS by their very nature are even more dependant on such a set of well-established standards to ensure thorough and consistent processes. Without going further into discussing why this is, we would instead like to focus on how to approach and establish standards in large and complex projects such as our case. We will attempt to analyse what has been done with respect to standardisation, and what the result has been.

8.3.1 Designing Standards

A number of papers about standardisation, especially in the healthcare domain, suggest that standards should be approached bottom-up. [Berg and Timmermans, 2000] say *"the phoenix of universality rises from the ashes of local chaos"*, which can be interpreted as: To have the best chances to successfully implement a health information system, one should not enforce standards upon the users, but let standards "evolve" throughout development and implementation. From the start of the project, this has been the intended course of action. The development has been conducted in a participatory manner, with the team working closely with the hospital employees.

Through meetings and feedback the team gathers requirements and insight in the work situation. After analysing the current practices, an optimised workflow can be created with the help of computer systems. Throughout this process interaction and training with the hospital employees is conducted, preparing them for the new way of conducting their day-to-day activitie. This step is in many ways one of the most difficult and critical in the implementation. While the goal of a bottom-up approach is to ensure that local procedures and processes are taken into consideration when building the system, part of the goal of such a system is also to optimise bottle-necks and provide new functionality. This requires very close cooperation with hospital actors to ensure smooth transition.

The project is not only trying to achieve HIS standards, but work standards as well. When the state health department released the RFP, they received a lot more proposals than anticipated, however, most of them were interpreted as *"unrealistic and utopian, even for western-world hospital standards!"* (HISP Project Manager). Some of the proposals had previously been bought and implemented at other hospitals in India, but had proven dissatisfying. The majority were also proprietary and prohibitively expensive, which caused the state health department to reconsider the wisdom of springing for such a system. They questioned whether spending money on technology and unproven IS was the best idea compared to hiring more staff, improving infrastructure and purchasing equipment. After a lengthy process, this led them to reject all the proposals, opting instead to contact the national government for advice. Through the government they were put in touch with HISP India who helped them create plans for developing a system partly from scratch, alongside plans to improve the health working standards through the implementation.

Unlike in the western world, where Hospital Information System (HospIS) and HospMIS based on EMR have been of great interest for decades, and are becoming increasingly common, there have not yet been any great ICT initiatives of this kind in India, or in any other developing countries. Most initiatives have been focusing on PHC, and because of this development efforts have been on concentrated around HMIS solutions for aggregated facility based statistics. Throughout a project's duration there are many decisions one has to face that it can be hard to predit the outcome of, and which can later have unforeseen impact. The initial choice of software is a good example of this. HISP India had two choices when they started this project, a) implement a HospMIS built for and on western standards or b) build their own system from scratch. They chose the latter, in accordance with the state health department (who in many ways had already had the first alternative available). Based on this, and that they wanted the system to be open-source, they landed on the decision to base the system on OpenMRS, an already successful and established product in the HIS world. While it's hard to speculate what the result would ultimately have been if a more classical HospMIS product had been chosen, and even though it's too early to say the outcome of the current system, our observations during our stay points to the state being well served by the system in its current form. Basing a system on local practices and processes is critical when attempting to introduce HIS in an environment where both resources and experience are in short supply. Any system based around idealised standards will inevitably fail.

8.3.2 The Hospital Core Module

Because of the fact that the system is still in its infancy, and currently has not been deployed in any other sites beyond DDU, it's hard to say how successful the standards created will be. The great litmus test will be when the implementation at DDU is finalised, and the distribution to other hospitals begins. Due to the nature of the nineteen other planned implementations, there will be far less time and resources dedicated, which necessitates a solid base system which requires a minimum of customization, but which at the same time meets the local requirements and is easily adapted to local processes. To what degree the system will be generic or possible to easily customise at each hospital, can be crucial for success. The hospital core module will play an important role here as it both creates a framework for controlled customisation, as well as restricts more direct customisation of individual modules. It is an example of how the need for standards can emerge, but also a good example of what the consequences of your choices can be further down the road. It also demonstrates that OpenMRS is missing fundamental pieces for the core of a complete HospMIS system, and might not have been a perfect match for the task.

In order to further investigate the standardisation effort, we have chosen to look at the pros and cons of this module to get a clearer picture of its meaning and importance.

Pros

1. **Standardised module communication**
   The initial issue which prompted the creation of the core module was intra-system communication between the different modules. Cross-dependencies between the modules is something that isn't supported in OpenMRS, and which necessitated a workaround. The prevailing

opinion after discussions with core OpenMRS developers and within the team, was that a third overarching module had to be created to facilitate communication. This proved to work fairly well, and in terms of the initial goal, the core module was a success.

2. **Customisability**
   Since the module is centralised and able to communicate with the other running modules, it can serve as a configuration point for different aspects of the system, and as such make the system more understandable and user-friendly.

3. **Generification**
   The initial plans for the module also included plans for dashboard functionality which would provide overviews of the the different departments on a day-to-day basis. Such functionality would, on the long term, likely be required in some form or another to create a more unified administrative interface for a hospital. Such functionality would be difficult to implement in a standalone module.

Cons

1. **Less Flexibility**
   The module lead to a more rigid system in total, as all the business logic of the different modules was moved into the core to facilitate communication. In practice this makes it impossible to pick and choose modules as they are no longer standalone.

2. **Lock-in**

   > “Lock-in is not only created by hardware and software. Information itself, its structures in databases as well as the semantics of the individual data elements, is linked together

into huge and complex networks that create lock-ins."
([Hanseth, 2000])

The core module creates lock-in in that it contains most of the business logic of the system, and can not easily be switched out or upgraded without affecting all the other running modules. This makes customisation very difficult.

3. **Time Consuming**
   At the time of the development of the core module, there was a shortage of developers on the team. Given the time constraints, the uncertainty of the core module and the necessary development, it may have been wiser to focus on developing the scheduled modules.

While there is little doubt that the core module will go a long way towards standardising and bringing the system together, this has a definite cost in terms of potential customisation and maintainability at later points. It's impossible to say at this point whether this tradeoff will be worth it, though so far it has solved the problems it was created to fix.

8.4 Improvements

While the project so far has been a success in terms of their stated goals, there has throughout the process been quite a few obstacles and challenges which has caused both delays and redesigns of the different modules. In this section we'll look at some of the points we feel it would be worthwhile to investigate for future projects.

8.4.1 Improved Documentation and Communication

One of the largest challenges of any organization is management of experience and knowledge. Even though a lot of resources are explicit in the form of paper or computer data, ensuring that the right people see these at the right time, can be a point of difficulty. In addition, a lot of experience is tacit, or hidden, often due to employees not realizing its importance, which further complicates efficient knowledge transfer. A good knowledge management strategy can in many cases mean the difference between a smooth transition process, and a costly redevelopment.

Knowledge manifests itself in all parts of a modern organization; in the minds of its employees, in the tools and technologies used, and in the structure of the organisation itself. In all levels there is both explicit and tacit knowledge about how best to operate and adjust, which has been built up over a long period of time. Especially in the IT-industry, but also in many other service-oriented businesses, this knowledge is the bread and butter of the organisation. Without the know-how of your employees, and the business practices to best employ those skillsets, you are left without a product.

[Rus and Lindvall, 2002] have described some of the risks involved with poor knowledge management:

- Loss of knowledge due to attrition
- Lack of knowledge, and an overly long time to acquire it due to steep learning curves.
- People repeating mistakes and performing rework because they forgot what they learned from previous projects.
- Individuals who own key knowledge becoming unavailable.

In a project where a large part of the actors involved, such as ourselves, are transient, this is an even greater risk. Indeed one of the earliest obstacles we encountered was the work left behind by the senior developer who left the project shortly after our arrival. Despite a conscious effort in advance to ensure that we had all the information necessary to continue his work, it quickly became evident that this hadn't been the case. While there was process in place to facilitate knowledge transfer, the simple fact was that omission of tacit knowledge, either through forgetting or not considering it important, became a problem for us later on. There was little in the way of documentation to help us understand the problems we were attempting to solve, and a large effort had to be done to uncover why certain decisions were made, and what the best way to proceed would be.

Such problems are common, and particularly in a distributed project it's important to ensure that documentation is taken seriously by everyone involved. As [Szulanski, 1996] points out, "Individuals who do not understand why particular practices are effective may not be adept at communicating their knowledge to others." It is our belief that a large amount of time and effort could have been saved if more time had been dedicated to planning and documenting the decisions taken throughout the development process, and ensuring that everyone made the necessary steps to document their work. It is far cheaper to create documentation that isn't needed than to need documentation and not have it.

8.4.2 Increase Focus on Development Processes

From the start of the project the development process has proven problematic. Despite plans for an iterative process, the team ended up assigning the modules to individual developers, and developing in what can be called a waterfall model. While this in itself isn't necessarily a problem, it turned out to be devastating in this particular case, as there were fundamental misunderstandings in the way the team had developed their modules. Because of the architectural problems inherent in the design, all but one of them had to be scrapped, and development was severely set back. In the words of the project leader:

> *What happened is after the module was done by the developer, as per him, it was given for testing, and then realizes there's so many drawbacks into it, because you know, the requirements were not [followed].*

Despite the obvious disadvantages the development process had, no particular changes were implemented after development was restarted, and during the time we spent on the project this continued in more or less the same fashion. While there were no problems of a similar magnitude at a later stage, there were nevertheless a lot of hitches throughout the project. During our second stay, the entire implementation team had to spend over a week redoing data imports of the entire hospital inventory due to a database change in the latest version of the inventory module. While this was in itself not catastrophic, it was very costly in terms of man-hours.

A similar problem occurred after our first stay in Shimla. Some time after we had left, we received a mail from the project leader regarding the blood bank module we had developed.

From: Project leader
pls this is urgent, blood bank module is not working. if not done urgently we will have to remove from there re-develop blood bank. pls consider this urgent. (..)

It's hard to say what the cause of these specific problems were, but it seems clear that the amount of testing performed before implementation was insufficient. The question then becomes, what method can be employed to improve the development process and the system quality? While the project leader wasn't familiar with Agile, she was quite clear on the problems they had faced:

> *"Had we followed the shorter releases I think it would've been much better, and as an implementer I know then what has been built, and as a developer he knows, you know, if it actually matches the requirements."*

Unfortunately, even in the case of blood bank where there was followup from the implementer, there was still very little in the way of testing during development. Anything beyond superficial testing was scheduled right before implementation, leaving virtually no time to actually fix severe bugs. Such bugs did not occur at the time, which may have seemed like a blessing, but it didn't take many weeks after our departure before lists of bugs started appearing in our mailboxes despite repeated warnings that we would not have time to fix them later.

While it's not surprising that implementers did not have time to actually test the modules for conformance at all times, it seems very short-sighted to postpone all such testing until the very last minute. Especially due to the initial requirements failure of the first four modules one would imagine that such a thing would be prioritised.

Mapping a method like XP or Scrum to this case is, however, not necessarily easy as agile methods were created for smaller co-located teams. [Ramesh et al., 2006] noted that many changes had to be made to fit with the traditional agile model. Much like those cases, the OpenMRS project has a very real need for an upfront and somewhat finalised requirements list due to contractual obligations, as well as synchronisation issues due to time differences. It is however, our belief, that with flexible planning, and through the use of collaborative tools, an agile process can be achieved that would greatly benefit the project.

The question remains whether or not the resource restrictions of the project, and the level of skill of the development team would be sufficient to accommodate a high-maintenance agile process. This is, however, beyond the scope of our study.

8.4.3 Risk Estimation

A lot of the problems we witnessed during our time on the project can best be described as planning issues. Many of the software solutions used have later turned out to poor fits, and require more work than previously assumed. While it's impossible to have perfect knowledge of a solution in advance, a pattern does seem to emerge in terms of a distinct lack of risk estimation.

When OpenMRS was first picked, very few on the team had any familiarity with the project. Only one of the developers had previously worked with. This proved to be costly, as there in the first months of the development process were fundamental misunderstandings in terms of the OpenMRS Data model, and how the planned modules should best interact with the OpenMRS core. Even well after this, when the redevelopment of the modules was well underway, there were still a lot of confusion with regards to OpenMRS. What has later become obvious is that OpenMRS, while a promising system, has a quite different direction and focus than the HISP project does. This has proven to create a lot of problems in terms of module architecture. As the health information officer put it on the questionnaire: *"On the project front both implementers and developers should educate/be educated as to the functionalities and limitations of OpenMRS."* If a common understanding of the capabilites of the software you're using is not established prior to the project planning, you run the risk of wasting significant amounts of time doing things the wrong way.

The choice of Palo was perhaps even more haphazard than the choice of

OpenMRS. The plans for RKS were laid out by one developer, who had investigated the problem and the options. As it turns out he did not end up creating the actual module, instead passing it on to us. At the time we started this work, there was virtually nobody on the team with a complete understanding of the planned solution, or the feasability of the solution. As it turns out, the solution was feasible, albeit far from simple. In the end, it took significantly longer to create than planned, and was ultimately scrapped, with great cost.

Similarly, the choice of creating the hospital core module, was a time investment which proved to have more drawbacks than anticipated. While there was a definite need for the core module, it later had to be scaled back due to the tight coupling it created between the modules. This kind of risk, while impossible to completely eliminate, can be greatly minimised through more thorough planning and risk estimation. While doing so will necessarily require more work which isn't directly productive, it will in most cases be a small price to pay when compared to the potential fallout of a project gone wrong.

8.4.4 Improved Cooperation With the OSS Commmunity

Any project based on an OSS system, is dependant on either good contact with the core developers, or strong project developers who themselves can solve deficiences and problems. Unlike a commercial system where there is a contract in place with regards to support, the creators of a Free and Open Source Software (FOSS) system bear no obligations towards its users. It is a hierarchy based around sharing, with those who choose not to participate in the process often having to sit on the sidelines. When deciding to base your software on a system like OpenMRS you are essentially staking a bet on the direction that the core developers are taking their system, and have to either cooperate with the developers to ensure that your needs are met in the development process, or hope that they have the same goals in mind as you.

With regards to OpenMRS, it initially seemed like the goals were unifiable, though it after a while became evident that the leadership of OpenMRS, while very interested in the HISP effort, were not overly keen on taking OpenMRS in the same direction. Despite this there was initially a lot of cooperation between the two teams, with OpenMRS providing the team with direct weekly support from one of the core developers. Shortly after we left Shimla the first time, this weekly meeting was changed to a public meeting instead of one dedicated to the HISP project, and cooperation between the two teams quickly dwindled.

When we later asked the HISP leader whether he considered contributing any of the work back to OpenMRS, his response was one of skepticism. Due to what he perceived as a lack of interest on OpenMRS' part, he considered it a waste of time to contribute back, and instead wanted the team to focus on their own project. Such a sentiment is understandable given the divergent goals of the two projects, though in the long run it's questionable whether it will be productive. Given that the system is based on OpenMRS, they either have the choice of continuing down the path that OpenMRS developers are paving in new version, creating their own version of OpenMRS through forking, or abandoning the system altogether and create their own core on which to base the modules. For the time being they have chosen to continue with OpenMRS. In a later interview the senior developer noted that he was preparing the modules for the latest version of OpenMRS. He did, however, express doubt about the future. The prospect of OpenMRS being able to break their modules in new version, is, understandably, worrying.

While it's hard to make predictions about which direction the project leaders should take with regards to OpenMRS, it seems to us that their best bet is to continue cooperating closely with the OpenMRS project. OSS is by and large a meritocracy, and if HISP is to have any say in the direction that OpenMRS is taking, it would be wise of them to maintain a presence, and to contribute back to the project to gain support for their views. Otherwise, as passive observers, they risk the project moving ahead without them, and being left with a dying platform which they are ill equipped to maintain.

Chapter 9

Conclusion

Throughout our time working on this project we have seen the system go from a fledgling set of independent modules, to a complex system implemented and used on a daily basis at a major Indian district hospital. By all accounts, the system has so far been a success. Much of the reason for this is the solid base that OpenMRS has provided, in addition to the hard work laid down by the HISP developers and implementors over the past year. While it's heartening to see that a fairly generic system like OpenMRS can be expanded to work in a hospital setting, our work nevertheless shows that care should be taken when planning and choosing such a system. Despite the success, there have been a lot of problems which could likely have been avoided through better planning and communication with the OpenMRS community. One should not be blinded by the prospect of free software; there are definite costs attached to adapting such a system, and skilled developers and implementors are no doubt needed.

Our research also shows that working in a distributed development comes with significant risks and pitfalls, which it can be very hard to avoid if not carefully planned for. Especially in a research project like this where many of the actors are transient participants, it is extremely important to ensure that knowledge is retained within the project. Many of the problems we faced throughout our stay were sustained from a lack of proper documentation and planning. The failure to complete the RKS module can to some degree be attributed to this, though there were many other causes which contributed as well. It is also important to consider the risks involved when planning the structure of the system. While things have turned out for the best so far, OpenMRS and the hospital core module still pose risks for the future development and expansion of the system.

While it's too early to tell what the result will be when the system is moved to the nineteen other district hospitals in Himachal Pradesh, it is our opinion that the idea of an open and free HospMIS has proven itself sound, and that the future bears a lot of promise for such systems. We also think that there is a lot of potential for further research on the future development and expansion of the system, particularly in the fields of software standardisation.

Bibliography

[Agerfalk et al., 2005] Agerfalk, P., Fitzgerald, B., Holmstrom, H., Lings, B., Lundell, B., and Conch'uir, E. (2005). A framework for considering opportunities and threats in distributed software development. In *International Workshop on Distributed Software Development*, pages 47–61. Citeseer.

[Berg and Timmermans, 2000] Berg, M. and Timmermans, S. (2000). Orders and their others: on the constitution of universalities in medical work. *Configurations*, 8(1):31–61.

[Braa et al., 2007] Braa, J., Hanseth, O., Heywood, A., Mohammed, W., and Shaw, V. (2007). Developing health information systems in developing countries: the flexible standards strategy. *Management Information Systems Quarterly*, 31(2):9.

[Braa et al., 2004] Braa, J., Monteiro, E., and Sahay, S. (2004). Networks of action: sustainable health information systems across developing countries. *Mis Quarterly*, 28(3):337–362.

[Carmel and Agarwal, 2000] Carmel, E. and Agarwal, R. (2000). Offshore Sourcing of Information Technology Work by America's Largest Firms. *American University, Washington DC.*

[Coiera, 2009] Coiera, E. (2009). Building a national health it system from the middle out. *Journal of the American Medical Informatics Association*, 16(3):271.

[Dybå and Dingsoyr, 2008] Dybå, T. and Dingsoyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859.

[Erickson et al., 2005] Erickson, J., Lyytinen, K., and Siau, K. (2005). Agile modeling, agile software development, and extreme programming: the state of research. *Research," Journal of Database Management*, 16(4):88–100.

[Flink, 1976] Flink, J. (1976). The car culture.

[Gates, 1976] Gates, B. (1976). An open letter to hobbyists. *Homebrew Computer Club Newsletter*, 2(1):2.

[Hanseth, 2000] Hanseth, O. (2000). The economics of standards. *From control to drift: The dynamics of corporate information infrastructures*, pages 56–70.

[Heeks et al., 2002] Heeks, R. et al. (2002). Information systems for public sector management. *Institute for Development Policy and Management, Manchester, UK*.

[Helsedepartementet, 2004] Helsedepartementet (2004). S@mspill 2007. Elektronisk samarbeid i helse-og sosialsektoren. Oslo, 2004.

[Kiel and Eng, 2003] Kiel, L. and Eng, P. (2003). Experiences in distributed development: A case study. In *GSD'03 The International Workshop on Global Software Development*, page 44. Citeseer.

[Klein and Myers, 1999] Klein, H. and Myers, M. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS quarterly*, 23(1):67–93.

[Lerner and Tirole, 2002] Lerner, J. and Tirole, J. (2002). Some simple economics of open source. *The journal of industrial economics*, 50(2):197–234.

[Lippeveld et al., 2000] Lippeveld, T., Sauerborn, R., and Bodart, C. (2000). *Design and implementation of health information systems.* World Health Organization Washington, DC.

[Manifesto, 2001] Manifesto, A. (2001). Manifesto for agile software development. *Retrieved November*, 29:2006.

[Orlikowski and Gash, 1994] Orlikowski, W. and Gash, D. (1994). Technological frames: making sense of information technology in organizations. *ACM Transactions on Information Systems (TOIS)*, 12(2):174–207.

[Perry et al., 2002] Perry, D., Staudenmayer, N., and Votta, L. (2002). People, organizations, and process improvement. *Software, IEEE*, 11(4):36–45.

[Ramesh et al., 2006] Ramesh, B., Cao, L., Mohan, K., and Xu, P. (2006). Can distributed software development be agile? *Commun. ACM*, 49:41–46.

[Raymond et al., 2001] Raymond, E. et al. (2001). *The cathedral and the bazaar: musings on Linux and open source by an accidental revolutionary.* O'Reilly & Associates, Inc.

[Rus and Lindvall, 2002] Rus, I. and Lindvall, M. (2002). Introduction: Knowledge management in software engineering. *IEEE Software*, 19(3):26–38.

[Scardino et al., 2006] Scardino, L., Potter, K., Young, A., Stone, L., Da Rold, C., Huntley, H., Dreyfuss, C., Longwood, J., Tramacere, G., and Maurer, W. (2006). Gartner on Outsourcing, 2006-2007. *Gartner Research Report G*, 144477.

[Seebregts et al., 2009] Seebregts, C., Mamlin, B., Biondich, P., Fraser, H., Wolfe, B., Jazayeri, D., Allen, C., Miranda, J., Baker, E., Musinguzi, N., et al. (2009). The OpenMRS implementers network. *International Journal of Medical Informatics*, 78(11):711–720.

[Sengupta et al., 2006] Sengupta, B., Chandra, S., and Sinha, V. (2006). A research agenda for distributed software development. In *Proceedings of the 28th international conference on Software engineering*, ICSE '06, pages 731–740, New York, NY, USA. ACM.

[Silber, 2003] Silber, D. (2003). *The case for eHealth*. European Institute of Public Administration.

[Silverman, 2005] Silverman, D. (2005). *Doing qualitative research: A practical handbook*. Sage Publications Ltd.

[Stallman, 1985] Stallman, R. (1985). The gnu manifesto. *Dr. Dobb's Journal of Software Tools*, 10(3):30–35.

[Susman and Evered, 1978] Susman, G. and Evered, R. (1978). An assessment of the scientific merits of action research. *Administrative science quarterly*, 23(4):582–603.

[Szulanski, 1996] Szulanski, G. (1996). Exploring Internal Stickiness: Impediments to the Transfer of Best Practice Within the Firm. *Strategic Management Journal*, 17:27–43.

[Von Hippel, 2005] Von Hippel, E. (2005). Open source software projects as user innovation networks. *Perspectives on free and open source software*, pages 267–278.

[Walsham, 2001a] Walsham, G. (2001a). Knowledge Management::: The Benefits and Limitations of Computer Systems. *European Management Journal*, 19(6):599–608.

[Walsham, 2001b] Walsham, G. (2001b). *Making a World of Difference: It in a Global Context*. John Wiley & Sons, Inc., New York, NY, USA.

[Walsham and Sahay, 2006] Walsham, G. and Sahay, S. (2006). Research on information systems in developing countries: Current landscape and future prospects. *Information Technology for Development*, 12(1):7–24.