Traceability
- A key to software success

David Olsson
This thesis is submitted to the Department of Software Engineering and Computer Science at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

**Contact Information:**

Author:
David Olsson
E-mail: David.Olsson@epk.ericsson.se

University advisor:
Conny Johansson
Department of Software Engineering and Computer Science

Department of Software Engineering and Computer Science
Blekinge Institute of Technology
SE - 372 25 Ronneby
Sweden

Internet : www.ipd.bth.se
Phone : +46 457 38 50 00
Fax : + 46 457 271 25
Abstract

From one year to another an intuitive feeling has grown stronger. That feeling tells us that poor traceability is the underlying reason for many of the problems that the software industry struggles with today. This thesis was carried out to see if this was true and if so investigate how traceability was related to the problems in today’s software industry.

In order to do this we have taken two different approaches. The first approach has been to try to establish if there exist support for this claim in existing literature. In the second approach we have tried to do the same thing by performing our own empirical study. Within this empirical study we have interviewed some project managers and some developers in order to find out which problems that are perceived by the software community as the most troublesome ones when it comes to software development.

Finally we have created a conceptual framework for increased traceability and then investigated if whether or not this framework could provide a good foundation for tackling the problems identified in literature as well as the problems discovered within the empirical study.

Keywords: traceability, communication, information models
Acknowledgements

I would like to thank all people that have contributed to this thesis especially...

- the Project managers and Developers at Ericsson Software Technology (EPK) who participated in the interviews.
- my advisor Conny Johansson for his straightforward and very constructive feedback.
- Ericsson Software Technology (EPK) for putting personnel as well as office space at my disposal.
Preface

I started to work on this thesis in the fall of 1999. Many of the ideas and issues discussed in this paper have emerged gradually over the last six years. During my studies in software engineering at the university of Karlskrona/Ronneby and later while working in the software business I have come to suspect that almost all problems in software development have to do with poor communication caused by lack of traceability. It has become more and more obvious to me, that in order for any idea to have a real impact on the success rates of software development projects it must have a strong focus on communication and the way we handle information in our organizations. Since 1997 I have worked, part time or full time, at EPK (Ericsson Software Technology) with the development of a number of tele communication systems.

In 1996 during the third year of my software engineering studies I got an opportunity to work as a project manager for a six month long student project involving 22 persons. The purpose of the project was to see if the NT-platform was a suitable platform for a telecom positioning system. The assignment was given by a unit at an Ericsson company called Ericsson Software Technology (EPK). Meanwhile the student project went on this unit begun developing basically the same system on a UNIX-platform. After the student project I more or less just continued with the work on the UNIX-version and that is how I started to work for Ericsson.

This UNIX-based positioning system was the first ‘real’ product I worked with at Ericsson Software Technology. The system allowed a mobile-operator to provide its customers with the possibilities of positioning their mobile stations. This was done using the existing GSM-network (with some updates), so no investments in new mobile stations (phones) were necessary. The project included about 10 developers and the assignment was to design and implement the heart of this system; a separate network-node providing the interface for, and the authorization of, the potential customers. Even though the core project only included 10 developers a number of additional developers were involved in the design and implementation of the necessary changes in the existing network nodes. There was also a project with the sole purpose of performing the system test, so the main project was a lot bigger.

From there I went on to the development of a Prepaid System for Ericsson, which as the name suggests allows customers to pay in advance for the usage of their mobile phones. This grant customers better control over their finances, and the operator of course gets money in advance. This was a somewhat bigger project an involved about 20 developers for the core system not including the people involved in system test.

Since then I have been involved in the second generation of the mobile position system and I am currently working with the third generation of it. The second one was no small upgrade but a completely new system built on top of a component server that we developed within the earlier mentioned Prepaid project.

The reason for why I write about my previous experiences is just so that you get a chance to reflect over the applicability and the trustworthiness of the contents in this thesis. It has never been my intention to discuss only problems and solutions applicable to the field of tele communications, but I think it is only fare to let you know that the majority of my experience comes from this particular field within software engineering. In an ideal world this would not influence the findings in this thesis but I believe that to be a utopia.
Contents

1 Introduction ................................................................. 9
  1.1 Background ................................................................................................................................. 9
  1.2 Common ground .......................................................................................................................... 10

2 Reading guidelines ......................................................... 11
  2.1 Required previous knowledge ...................................................................................................... 11
  2.2 How this thesis is structured ...................................................................................................... 11
  2.3 The essence of this thesis .......................................................................................................... 11
  2.4 Outline ....................................................................................................................................... 12

3 Scope and purpose .......................................................... 13

4 The software industry today ............................................... 14
  4.1 Introduction .................................................................................................................................. 14
  4.2 The software industry today ...................................................................................................... 14
  4.3 Summary ..................................................................................................................................... 15

5 An increased need for flexibility .............................................. 16
  5.1 Summary ..................................................................................................................................... 17

6 Communication ................................................................. 18
  6.1 An increased need for communication ........................................................................................ 18
  6.2 Communication a key factor for success ..................................................................................... 18
  6.3 Problems in communication ..................................................................................................... 19
    6.3.1 The message does not get through .......................................................................................... 19
    6.3.2 The message is rejected .......................................................................................................... 19
    6.3.3 The message changes ............................................................................................................. 20
  6.4 Summary ..................................................................................................................................... 20

7 Traceability ............................................................................. 21
  7.1 Introduction .................................................................................................................................. 21
  7.2 What is it? ..................................................................................................................................... 21
  7.3 Our definition ............................................................................................................................... 21
  7.4 How does traceability relate to communication? ........................................................................ 22
    7.4.1 The message end up in the wrong place .................................................................................. 22
    7.4.2 The message is not noticed .................................................................................................... 22
    7.4.3 The message is delayed on its way ......................................................................................... 22
    7.4.4 You get too many messages .................................................................................................. 23
    7.4.5 The message is not understood ............................................................................................ 23
    7.4.6 The message does not appear in the proper context ............................................................... 23
    7.4.7 The message is misunderstood ............................................................................................. 24
  7.5 What makes us loose it? ............................................................................................................. 24
    7.5.1 Poor modelling ....................................................................................................................... 24
    7.5.2 Document fixation .................................................................................................................. 25
1 Introduction

1.1 Background

It has been 25 years since Fredrick Brooks wrote his famous book ‘The mythical man-month’ but still the software industry struggle with most of the problems he brought up in his book. In 1986 he wrote an article called ‘No Silver Bullet’ in which he claimed that - “There is no single development, in either technology or management technique, which by itself promises even one order-of-magnitude improvement within a decade in productivity, in reliability, in simplicity.” [5]. Sadly this prophecy of woe came true, but is this true for the next decade as well?

We agree with Brooks that there are some very hard problems to solve, we just have to look at the software industry today to confirm that. In 1996 the Standish Group [27] reported a success rate for American software projects as low as 16.2%. In other words, 83.8% were not successful. In addition to the 16.2% successful projects another 52.7% were finished but were over-budget, over the time estimates, and with less functionality than originally specified. With figures like this it is obvious that the software industry has major problems. Then, how can we improve these figures and what is the underlying problem?

Tom Gilb [10] argues that the underlying problem from which all other problems follow is communication. He expresses it like, “The real problem is communication between people”. If this is true, this must imply that any idea (tool, process etc.) that wants to have a significant impact on the success rate for software development projects must have a strong focus on communication and thereby on issues related to how we handle information in our organizations.
1.2 Common ground

Much of the work within the software engineering business (if not all of it) can quite naturally (at least theoretically) be modelled as one single network. This network can be thought of as an extended hierarchy. The reason why an ordinary hierarchy is not enough is because it only allows for one parent for each entity. In software development work some of the entities depend upon several others, e.g. a software component will probably implement more than one requirement. For example; a component for authority checking will probably check for a valid client as well as for a valid password.

However this network is not flat in nature. All relations have a direction. The direction tells us who is dependent on whom. For instance, we can place high-level requirements in this network, and as children to them, design decisions, and as children in their turn we can have test cases and so on. It has been our experience that the software engineering industry does not take full advantage of this natural way of structuring the software entities. Instead the different types of entities are often separated from each other and thereby the logical connection is lost. If we somehow could preserve and visualize them in for instance a model, that model could then constitute a Common Ground for all people involved in the software development.
2 **Reading guidelines**

2.1 **Required previous knowledge**

This master thesis is intended to be a contribution to the field of Software Engineering. We think that in order to gain anything from this report at least a basic knowledge of software development in projects is required. On the other hand, no special knowledge in the telecommunication domain is necessary.

2.2 **How this thesis is structured**

We begin this thesis with a presentation of the underlying hypothesis around which it is constructed. From there we begin our work of proving or disproving the first part of this hypothesis (e.g., Is traceability really the key issue?). This is done via two different paths. The first path goes through existing literature. The second path goes via an empirical study including a number of interviews. In the last part of the thesis the idea is to move in the opposite direction. This is done via an attempt to answer the question: “How does improvements in traceability address software development problems in literature and the problems received during our interviews?” Within this attempt a conceptual traceability framework is presented.

---

![Diagram: Empirical approach (Traceability a key issue?) → Theoretical approach (Traceability a key issue?) → Semi practical approach]

---

2.3 **The essence of this thesis**

If you are only interested in what is really the contribution of this thesis then it is possible to begin with chapter 3 (**Scope and purpose**) and from there on head directly to chapter 11 (**A traceability framework**). This chapter is really the climax of this thesis. Another chapter that also might be of interest for the quick reader is chapter 9 (**Interviews**).

- Every chapter is concluded with a summary where you can get an idea about its contents.
2.4 Outline

Introduction

Chapter 3 (Scope and purpose) describes the scope and purpose of this thesis. It also includes our hypothesis.

Chapter 4 (The software industry today) tries to give an insight to where the software industry stands today. Is there a problem or has the software industry matured into a discipline with reliable forecasts and where failures are the exceptions. This chapter also presents some recent success-rate figures for the software community.

Theoretical approach

Chapter 5 (An increased need for flexibility) reflects over the effects of the increased change rate in the software industry.

Chapter 6 (Communication) reflects over communication. It describes why the importance of good communication increases with decentralization.

Chapter 7 (Traceability) reflects over traceability. What is it? Why is it important? How does it relate to communication? What can destroy it?

Empirical approach

Chapter 8 (Method) describes the method we use for the empirical study.

Chapter 9 (Interviews) presents the findings from the interviews performed within the empirical study.

Chapter 10 (Analysis conclusion) contains an analysis of the interviews.

Semi-practical approach

Chapter 11 (A traceability framework) presents “A Common Ground”, a proposal for a communication oriented traceability framework.

Summary

Chapter 12 (Summary) tries to summaries the findings from this thesis.

Chapter 13 (Further research) discusses what has been left out from this thesis and what could constitute material for further investigations.

Formalities

Chapter 14 (Appendix) is an appendix containing the questions used during the interviews as well as printouts from the actual interviews.

Chapter 15 (References) contains a list of all references used within this thesis.
3 Scope and purpose

Hypothesis:

The majority of the really troublesome problems within software development has to do with poor traceability, either directly, or via communication, so a framework that drives us to define and preserve the connections between the different information entities such as: requirements, design decisions, test cases, estimation figures, etc. should be able to provide us with help on the majority of problems reported by the software community.

The purpose of this thesis is to:

- ...establish if there exist a path through existing literature that supports that traceability is a key factor for increased success rates.

- ...make an empirical study to find out if the problems within software development that are perceived as the most troublesome ones are in fact related to traceability.

- ...to find out if a traceability framework can be used as a base for addressing most of the problems identified in the empirical study as well as the problems from existing literature.

The chain of success

increased traceability

\[\downarrow\]

more effective communication

\[\downarrow\]

increased flexibility

\[\downarrow\]

higher success rates

leads to
4 The software industry today

4.1 Introduction
In this chapter will we try to get an understanding about where the software industry stands today. Has the industry matured over the last years or does it still produce highly unreliable estimates of time and cost? Are the success rates acceptable? Is success more common than failure?

4.2 The software industry today

Standish Group research shows a staggering 31.1% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates. The costs of these failures and overruns are just the tip of the iceberg. The lost opportunity costs are not measurable, but could easily be in trillions of dollars. One just has to look to the City of Denver to realize the extent of this problem. The failure to produce a reliable software to handle baggage at the new Denver airport is costing the city $1.1 million per day [27].

Standish Group (Chaos)

It seems little has happened in the 25 years since Fredrick Brooks wrote “The mythical man-month” [5]. The failing rate is still alarmingly high. The software industry has not matured into an industry with success rates like those for bridge building. IT executives report status quo if not worse comparing failing rates today and five as well as ten years ago.

Only 16.2% of American software projects are completed on-time and within budget and with the functionality originally specified. As many as 31.1% of the projects are never completed. 52.7% of the projects are completed but over-budget over the time estimate, and with less functionality than originally specified. The average time overrun is 222%! If you look only at large companies the figures are even worse, only 9% of their software projects are successfully completed [27].

The Standish Group are not alone in their believes. Lederer and Prasad [18] report that nearly two-thirds of all major projects substantially overrun their estimates. Their study shows that the number one reason for inaccuracy in the initial estimates is ‘frequent requests for changes by user’. The number two reason is ‘overlooked tasks’.

Examples of software catastrophes abound. An unpublished review of 17 major Department of Defense (DoD) software contracts found that the average 28-month schedule was missed by 20 months. One 4-year project was not delivered for 7 years; no project was on time [6].
A report from the US Government Accounting Office concludes, “The understanding of software as a product and software development as a process is not keeping pace with the growing complexity and software dependence of existing and emerging mission-critical systems” [11].

Hans van Vliet [31] says; “Getting reliable cost and schedule estimates for software development projects is still largely a dream.

4.3 Summary

There seem to be a large consensus about the maturity of the software industry. The industry still has major problems. Complete successes with projects on time and within budget are the exceptions. Standish Group [27] report that the average overrun is 222%! Somehow the software industry must learn how to produce reliable estimates of cost and time. There definitely still exist problems that must be solved.
5 An increased need for flexibility

To compete effectively in this age of revolutions and transformations, the enterprise must adopt new strategies, organizational forms, and practices that support its transformation. To increase enterprise agility, it must initiate a new life cycle based on discontinuous business vision harmoniously coupled with evolutionary business process and technology change. The substantive questions involve how to start and how to do it, and every move becomes an organizational experiment in learning how to change faster and better. [29]

The world is changing more and more rapidly. If you look at the technical evolution it follows almost an exponential curve. An unproportionally large part of the ‘every day changing’ inventions have been made during the last century. Since the world is changing at a greater and greater pace the ability to adapt to changes is becoming more and more important. Today it has reached the level of an absolute necessity. A Director of MIS at a securities firm says: “Changes, changes, changes; they’re the real killers.” [27]. Kotonya and Sommerville says in a discussion about developing software systems: “It is often the case that more than 50% of a system’s requirements will be modified before it is put into service.” [10].

What then is the key for coping with massive changes? Flexibility of course! It is like the new building techniques developed to better sustain earthquakes. Their most important ingredient is flexibility, to allow the building to flex with the movement on the ground. Thus flexibility is a vital characteristic for buildings during earthquakes but the same goes for organizations involved with software development, which exist in a constant earthquake of changes in techniques, necessary competence, and requirements.

In a world where changes accelerate, adaptability and communication effectiveness becomes more and more important, even a midsize software company can not have someone sitting at the top of the organization making all strategic decisions. If an organization wants to keep up it must delegate responsibility among all its employees. Nicholas expresses it like - “When adaptability and rapid response to change are called for, such as in volatile technological or market environments, more “organic” forms of organization and management are required. These forms accommodate the need for high-level technical and managerial competence, and considerably expand the latitude and degree of decentralization.” [23]

In the book “Organization Theory and Design” Richard L. Daft argues that today’s organizations need to build in change and not treat it just as an external stimuli. He says; “...change-not stability-is the natural order of things in today’s global environment.” [26]. He goes on with a discussion about a number of possible change approaches and concludes the section by saying; “All of these approaches typically favour organic conditions that lead to employee participation in decisions, interesting work, and the freedom to initiate ideas to improve their jobs.” [26]
Tom Gilb is another man arguing for more network oriented organizations. He declares the ‘Bill of Rights’ in his book Principles of Software Engineering Management. Especially ‘Right’ 5, 6, and 8 makes way for a more decentralized organization -

1 You have the right to know precisely what is expected of you.
2 You have the right to clarify things with colleagues, anywhere in the organization.
3 You have the right to initiate clearer definitions of objectives and strategies.
4 You have a right to get objectives presented in measurable, quantified formats.
5 You have a right to change your objectives and strategies, for better performance.
6 You have the right to try out new ideas for improving communication.
7 You have the right to fail when trying, but also to kill failures quickly.
8 You have a right to challenge constructively higher-level objectives and strategies.
9 You have a right to be judged objectively on your performance against measurable objectives.
10 You have a right to offer constructive help to colleagues to improve communication.

In order to shift responsibility like Gilb suggests we must adapt our entire organization. This has implications on everything we do, the way we work, the tools we use, how we communicate, etc.

5.1 Summary
In this chapter we have seen that the world is changing more and more rapidly. Some have pointed out changes as the single most important problem to overcome in software development. In order to better sustain changes our organizations must become more flexible. In fact some say that we must change the way we think and begin building in change into our processes and organizations and not treat it as an external stimuli.

We have also seen that many suggest decentralization as a way to make an organization become more flexible. In the next chapter we will take a look at what impact more decentralization has on communication within the organization.
6 Communication

6.1 An increased need for communication

In the previous chapter we saw that today’s software organizations move towards a more network-oriented structure. Then which are the implications of this restructuring? One very important implication is that it makes it more difficult to provide the strategic decision makers (namely everyone) with accurate information. In centralized organizations, this type of information only has to be given to a very limited number of people. So with more organic organizations the need for good communication increases drastically.

Here we can see that the possible communication paths increases exponentially rather than linearly. If we also consider indirect communication paths, the number of different communication paths seem to explode. It is probably safe to assume that changes in how information is communicated have huge implications in a networked organization.

Marlow and Wilson suggest how you can determine if an organization is centralized or network oriented - “In network organizations, horizontal communication not only predominates, it flourishes” [8].

6.2 Communication a key factor for success

The implications of technological change and product obsolescence present a situation in which the various functions of a corporation must work closer together if the firm is to survive, and they must be provided with accurate, succinct, and timely information. Communications effectiveness throughout the corporation will thus become imperative for survival. [8]

Eugene Marlow
Many have identified communication as the key factor for success in the software business. Lindstaedt and Schneider say, “It is widely recognized that groups engaged in the development of complex software systems face a number of critical communication issues” [27]. In a discussion about Software Engineering Management Gilb claims that “The real problem is communication between people” [10]. What he is saying is, that although there might appear to exist an enormous amount of totally diverted problems, they all come down to poor or non-existent communication. Go deeper and eventually it will show its ugly face, miss communication! On the brighter side though; if you solve the communication problem you will have solved almost all other problems as well.

### 6.3 Problems in communication

Communication is not an easy thing. All sorts of problems can distort our communication channels, or even worse, break them down completely. Within this section we will present a number of potential problems brought forward by Wiio [24], a well-known professor and researcher about communication at a trade school in Helsinki. He identifies the following basic problems.

1. The message end up in the wrong place
2. The message is not noticed
3. The message is delayed on its way
4. You get too many messages
5. The message is not understood
6. The message does not appear in the proper context
7. The message is misunderstood

#### 6.3.1 The message does not get through

**The message end up in the wrong place**
This might seem like an obvious problem but according to Wiio it is a rather common problem. Maybe the wrong communication channels have been chosen.

**The message is not noticed**
Very often messages are not noticed by the intended recipient. There is such a strong competition between messages that very often they get lost.

**The message is delayed on its way**
The message might get through but so late that it no longer has an importance. The reasons for the delay could be many: someone has forgotten to tell about it, the message is buried in a pile of other messages.

#### 6.3.2 The message is rejected

**You get too many messages**
The human being has a very limited capacity when it comes to processing data. There is an overflow of information and most of it has to be thrown away in order not to jeopardize the recipient’s psychological health.

**The message is not understood**
The message does not make sense to the recipient and is therefore rejected.
6.3.3 The message changes

The message is misunderstood
When it comes to understanding a message there exist many levels ranging from a complete lack of ability to understand to a complete misunderstanding. When it comes to interpreting a message it is of vital importance that the short-term memory is able to produce meaningful holistic thought units. These though units might not necessarily be correct or like the sender intended them. The recipient might be completely convinced that he has understood the message anyway.

6.4 Summary
In this chapter we have discussed how more decentralized organizations increases the importance of good communication. We have seen that in a network-oriented, or decentralized organization, the possible communication paths quickly reach astronomical numbers. It is probably safe to assume that communication issues are extremely important in decentralized organizations.

Many have pointed out communication as a key area for success or even survival in the software industry. We have also looked closer at the different communication problems that we may encounter within our organizations. We saw that there is a wide range of possible communication problems. They could be divided into three main categories namely, the message does not get through, the message is rejected, or the messages is misunderstood.

In the next chapter we will look at traceability as a concept and try to understand if and how it relates to the communication problems mentioned in this chapter.
In this chapter we will take a look at traceability as a concept. We will try to explain what it is and in that context also give our own traceability definition. Further more, we will try to answer questions like; why is traceability important? How does it relate to communication? What can give us poor traceability?

Every piece of substance is built up from the same basic compounds, namely protons, neutrons, and electrons. Just looking at the compounds all substances looks more or less the same. It is how these compounds are connected together that makes a world of difference. A good orchestra playing a classical piece might sound impressive, but if you are not careful with the order of the different notes, you will probably just end up with a loud noise. The same goes for software development. In order for a project to 'play' beautifully the information have to be structured in a good way. Which requirements have lead to a specific design decision, which piece of functionality is a certain test case supposed to test, etc. Without these connections the project completely falls apart. It becomes impossible for anyone to see the effects of changing requirements. No one knows if all functionality have been tested, how much time has been spent on a specific part of the system etc.

A software system is much more than just its executables. It is also everything else needed to understand, maintain, and operate the system [5]. This is important to keep in mind when trying to define traceability. Our definition of traceability is; the degree to which it is possible and easy to get hold of context information of interest, for all information entities (e.g. requirements, design decisions, test cases, reported bugs, progress figures, code etc.) in the given domain. A piece of context information is in itself an information entity, which in turn has relations to others.

This definition conforms more or less to Lindvall’s definition [19]. The one main difference though is that Lindvall’s definition only speaks about tracing within and between models. This would, if taken to it’s extreme mean that an organization that did not use any models at all could be seen as having great traceability even if it is impossible to navigate from one piece of information to another. If we widen our perception of models to include even documents, this would nevertheless not be enough to make the two definitions equivalent. This is because our definition also includes information yet outside all models e.g. a newly discovered requirement that only exists in a single persons mind. In other words, the traceability definition that we propose implies that good traceability also means that it must be easy to put new things into our models since it must be easy to find the related context information that already exist within our models.

Lindvall’s traceability perception is still more holistic than most’s. The bigger portion of the existing literature devoted to traceability is particularly focusing on just requirements traceability. There exist numerous definitions for requirements traceability [10][24][14]. Definitions of traceability as a whole is a lot more scares [19]. The same thinking seems to have influenced the development of traceability tools. Most of them too only support select aspects of the traceability problem [19].
7.4 How does traceability relate to communication?

We will here go through the communication problems brought up by Wiio [24] one by one to see if they in some way are related to poor traceability, and if so, how?

7.4.1 The message end up in the wrong place

In a discussion about problems with e-mail usage, Rose, Bornstein, and Tiene [7] say “The point-to-point addressing scheme means that the burden is on the sender to not only be aware of anyone who should receive the message, but also to know those users’ e-mail addresses”. If we apply our definition of traceability which implies that the easier it is to find context information for different information entities the more traceability we got, then we can see that traceability in this context means the easiness to find information about who is interested in what and also e-mail addresses to those individuals. Any solution to this problem must therefore improve traceability according to our definition.

7.4.2 The message is not noticed

One reason for this suggested by Wiio[24] is the strong competition between messages. This problem is of course strongly related to the problem of too many messages discussed in section 7.4.4. Rose, Bornstein, and Tiene [7] strengthens Wiio’s claim with a hypothesis which they imply is true; “Many people will have access to far more RELEVANT information than they can reasonably hope to process”. They continue by saying; “If this hypothesis proves to be true, then it is not enough to separate the relevant items from the non relevant ones; users are drowning in wheat. Instead, our task is to figure out which items to present to each user at a particular time, given his or her specific resource and task constraints”. This relates to the second half of our definition which speaks about context information of interest. In order to fulfil this it seems we have to structure our information so that it is possible to get to it from different starting points depending on who wants it or when someone wants it.

7.4.3 The message is delayed on its way

The traceability relation here might not be obvious at first. If we start by looking at the case where someone forgets to tell about it then we have to ask ourselves, what is the underlying reason? Of course it could be a lot of things. For instance, the person with the message can not in an easy fashion get hold of information about who is interested in the message so he passes the message on to a person who he believes knows who to inform. The longer these chains get the greater the risk for someone forgetting to pass the information along.

We can also ask ourselves why is this person given the possibility to forget about the message. This could be because there is a lack of an automated information service. The reason for this in turn, could be lack of traceability. If messages were created with an automated information service then the sender could be releaved from the responsibility of sending the information. This could be handled by the service itself.

If we go on with the case when the message is delayed because it has been buried in a pile of messages this is definitely a case of poor traceability according to our definition. If messages are stored temporary or permanently in a large pile it is obviously hard to come by this information for all persons that might have an interest in them.
7.4.4 You get too many messages

Never has so much data been fed to us. On top of the direct contacts with other people that has always been there we have radio, television, telephones, e-mail, cd:s, the internet, etc. Today we are more or less drowning in this data flow. Turner comments on this by saying - “Surprisingly, poor communication on projects is often caused by too much rather than too little” [22].

One reason for why people get so many messages has to do with the need of providing all coo-workers with proper information. Often information broadcasting is used. According to our own experience and during the empirical study we have seen that the reason for why information broadcast is so widely used is that it is often hard to identify who should receive specific pieces of information. One example would be an e-mail saying that a requirement specification has been updated. This mail is sent to the whole development project. Some people that are unaffected reads it while the affected people forgets to do so. This whole situation is due to the fact that it is hard to identify who should receive the information. In other words lack of traceability. Almost every time information broadcasting is used it shows a lack of traceability. This might appear harmless because ‘- at least the people who should have the information gets it’, so what if it is read by some more people, ‘- the more they know the more they understand’. Unfortunately this is not the case. Like in our example some people might forget or chose not to read it because ‘- I get so much junk’. Turner says - “The project manager must define those who need the information, so that when people receive something they know they ought to read it” [22].

7.4.5 The message is not understood

The reasons for why a message is not understood at all can be divided into three main categories.

- The message is sent to the wrong person who has no idea of what it is about.
- The message in itself is not accurate.
- Context information needed to understand the message is missing.

We have already talked about the first category in section 7.4.1 so we will go onto the second one. A reason for why the message is not accurate could be that the sender did not write what he though. Sometimes people think something and say something else. This problem does not seem to have any clear traceability relation. It is our experience though that the third reason is by far the most common one. The sender produces the message in a very specific context but often that context is not as easily available to the recipient as the sender assumes.

7.4.6 The message does not appear in the proper context

In order to gain anything from a bunch of data we must understand its context. If for example you get a mail saying that a car is being towed of you might do nothing, but if the mail also says a red BMW with license plate AKC760 you might react quite differently because you understand that it is your car. Because you are able to connect the information, in this case to yourself, you can also understand the implication that you might have to go the 10 miles from work to home. Data only becomes information when you can understand the context.
The traceability relation to the problem when a message does not appear in the proper context is clear as a bell. In accordance with our traceability definition, context information has everything to do with traceability.

7.4.7 The message is misunderstood

Even when people in the organization get hold of the proper information they risk misinterpreting it [24]. Of course this risk can be greatly decreased if the information is well structured and succinct (we will come back to this later). But even if the person that has put together the information has done a superb job an innegligible risk of misinterpretation still exist. In our experience the best thing you can do to decrease this risk even further is to provide the reader with proper and easy accessible context information so that he or she can really understand it. This is in our opinion an extremely powerful technique. It might seem scary to some people because everyone can begin to think on their own because when you provide context information you risk that persons reading the information that you have put together actually checks the context information and might even come up with a different conclusion than you.

When it comes to structuring of information, Parnas ideas about data hiding is still one of the most powerful techniques available. His ideas can be applied to any information not just software code. The key is to hide the right information. If too much information is presented right away there is a big risk that the most important information will drown in the total information sea [7]. When we say hide information we do not suggest that it is moved out of reach for the reader but merely that accessing that information should require a deliberate action (i.e. to ask for it, e.g. via a mouse click).

7.5 What makes us loose it?

As we have said before, traceability is really about identification and preservation of the endless number of logical connections between information entities. In the following sections we will bring up some of the reasons for why we might miss or lose the valuable connections.

How we represent things is of utter most importance. Brooks [5] (p.103) claims that *representation is the essence of programming.*. Representation/Modelling is discussed rather frequent in literature. The connection between document fixation (which we will talk about later) and modelling though is to our knowledge not explicitly mentioned in existing literature. We claim that such a connection in fact does exist mostly based on own experience even though there do exist some indirect support for this claim in existing literature as well.

7.5.1 Poor modelling

*Most real-word decisions are multidimensional in nature, and the range of considerations they involve is confusingly wide. We tend to try to express problems in a simple form in order to make them intellectually manageable. This is not in itself a bad thing, but a necessary process in understanding the real world. However there is a danger that if carried too far, it becomes over-simplification. The consequence of over-simplification is that we often don’t find an appropriate solution for what is, in reality, a very complex problem.*
Gilb does here apply and adapt Einstein’s over-simplification principal “Things should be as simple as possible, but not simpler” to how we should represent problems. We believe that this principal is true not only for problem representation but for the entire range of representations that we use in software development. The representations and models we use should of course be as simple as possible, but if they cut away too much of the complexity and the richness of the information they risk becoming blinders instead of useful tools.

Many of the simpler traceability tools have traceability representations that are decoupled from the development environment which make them very limited when it comes to capturing dynamic traceability information [2]. Unfortunately it is not only the simpler tools that support traceability poorly. A lot of the popular requirements traceability tools although embedded with the development environment only address limited aspects of the system development life cycle [2].

Today there exist a lot of tools that represent their part of the work quite well but the different models must come together to really improve software development. The connections between the representations are equally important as the representations themselves. It is the relations to other representations that make a representation interesting.

### 7.5.2 Document fixation

To traverse a famous statement by John Donne [8], no single piece of information is an island. Everything exists in a context. A change in some part will have impact on some other part. Everything can be said to be interwoven with an enormous number of connections. These connections will affect us whether or not we are aware of their existence. We therefore think it is crucial to identify and preserve these connections when we model and structure our work.

What can keep us from identifying or make us lose identified connections then? Well, this is a complex issue but in our opinion it holds some important keys for improving software development.

One way to improve traceability would be to start exchanging our old information models (e.g. documents used as information holders) that destroys the connections between the information entities and instead use new ones that preserve the connections and even better urges us to model them.

You can always hear endless discussions about whether we ought to document more or less. One of the reasons for why an agreement never is reached on this subject we believe comes from a mix-up of issues. Most people associate more documentation with more documents. This might seem logical but is an unfortunate mix-up due to a several thousands year old inheritance. Documents were invented long before computers and for a long time documents have been by far the best way to store and distribute information. This has lead to that the word for writing descriptions about different things and the word for the media used merged into one and the same. This is probably an important reason for why people have such difficulties to imagine using anything else than documents for documenta-
tation. People arguing for less documentation has usually discovered that documents fulfil their role poorly in today’s software development and therefore quite rightly believes a lot of them to be a waste of time and resources. On the other hand, the people arguing for more documentation have discovered a shortage for explanations and information about how and why. This is sadly not true for all ardent advocates of more documentation. Some are just interested to use a massive amount of documents to cover up their incompetence and the fact that they do not know how to get to the goal or even where it is located.

The first thing we will talk about is the way we use documents in today’s software development. Documents have some important characteristics and advantages. The restrictions we find in a document when we look at it on a PC monitor has almost all to do with the fact that it must be possible to print it on paper. As we said earlier, documents were for a long time by far the best way to store and distribute information. Paper (not documents) still holds some important advantages compared to the hard disk of a modern computer. You can spill a cup coffee on it and you will most likely still be able to read what it says. You can rip it into pieces but it still possible to put it back together (might require a lot of work though). The best monitor in the world still comes short compared to a sheet of paper when it comes to reading long texts. In other words, we believe that paper still is the best medium if we want to read long texts. On the other hand it is not the best medium if we do not want to access the information sequentially. For quick searches, the hard disk beats the paper by far. A big problem with us humans is that we have a lot of problem with ‘thinking outside the box’. The historical inheritage of how to use documents keep us from taking full advantage of new tools and media. We tend to work with new tools in old ways.

We would like to propose a new way of using documents. Get rid of the old concept of a document as an information holder. Instead use it for what it is best at, to read long texts. It is okay to print some information just to make it easier to read but then look at the printout (document) as a snapshot of the information model, not the actual information holder. The ‘documents’ should never take precedence over the actual information model stored on hard disk. Because of computer networks and ‘browser’ technology everyone can pretty much look at the same information at the same time and also work with it at the same time. Kotonya and Sommerville also discusses the disadvantage of storing information in documents. The context here is requirements management. They mention five distinct drawbacks with storing requirements in documents [10]:

1. Information about requirements dependencies (traceability information) has to be externally maintained.
2. The facilities available for searching the requirements are limited to whatever word processor searching facilities are available. It is not usually easy to find groups of requirements which have common characteristics.
3. It is not possible to electronically link requirements with changes which have been proposed.
4. Any version control of the requirements has to be at the level of the whole requirements document or, at least, individual document chapters. It isn’t usually possible to maintain different versions of the same requirement.
5. It isn’t possible to navigate automatically between related requirements or between different requirements representations (e.g. from a textual representation to a system model).
Although this discussion is about requirements it is our belief and experience that the same goes for most of the information we deal with in software development. Isolation always decreases the value of the information.

The old document concept leads us to ‘flatten’ our information model because that a document is flat in nature. Sure we can use hierarchical numbering etc., but it is hard to use it for anything else than to just divide the information into sequential pieces. The ‘flattening’ might appear harmless enough but we believe it to be an important reasons for losing control over the information model. Flattening the information model makes us loose a lot of the multidimensional relationships between different pieces of information. This in turn undermines the understanding for why the different pieces of information exist in the information model at all. Thus the impact of changes becomes a lot less obvious. These multidimensional connections are the very foundation upon which any work on improving the information flow must rely on.

7.6 Summary
Here we have taken a look at what traceability is and proposed our own definition which is; the degree to which it is possible and easy to get hold of context information of interest, for all information entities (e.g. requirements, design decisions, test cases, reported bugs, progress figures, code etc.) in the given domain. We saw that there existed similar definitions but the one we presented was a bit wider than for instance Lindvall’s [19].

We also looked at how the communication problems presented in the previous chapter relate to poor traceability. We saw that most of them had a clear relation to poor traceability. Many of the problems reported as communication problems might in fact in turn be a result of poor traceability.

We discussed the negative impact that poor models have on traceability. We saw that many of the existing traceability tools only provide possibilities to model limited parts of the total work needed to produce software.

Finally we identified documents, or rather the way we use them, as one of these poor information models that makes us loose traceability.
8 Method

The purpose of this empirical study has been to investigate which problems practitioners find the most troublesome in software development. We would also like to understand how many of these problems that are related to traceability and also how they are related.

There exist a lot of written information about problems in software development but we thought that it was a good idea to take our own ‘real world sample’. A lot of existing literature has been written by authors that can not be considered to represent the average developer or project manager. In order to get this sample of which problems developers find to be the most troublesome we have interviewed six persons working for Ericsson Software Technology, three developers and three project managers. For both groups we have chosen one experienced, one semi-experienced, and one inexperienced person. This was done in order to see if the most troublesome problems vary with experience.

8.1 Qualitative interviews vs. quantitative

There exist two main possibilities when using interviews in an empirical study, qualitative interviews or quantitative interviews. Both qualitative interviews and quantitative interviews have their advantages and drawbacks. Quantitative interview’s strong side is of course the greater amount of underlying data but suffers from a huge risk of just being able to find what you expect [1]. In this thesis we wanted to make an investigation of the most troublesome problems in software development. Using a qualitative methodology allowed us to use very open questions that hopefully could reveal even those problems that we were not aware of ourselves.

8.2 Selection

The selection of the persons that we have interviewed has been done in a quite straightforward way. All of the selected persons either work as Project Managers or as Developers. One of our basic ideas for the selection of people for both these groups was to get views from people with varying experience. We personally knew all the interviewed persons except one Project Manager and one developer. We will later discuss the possible problems of knowing the persons you interview.

8.3 Pitfalls

When performing formal interviews there exist a lot of pitfalls. We will here present some of them and also describe how we have tried to avoid them.

8.3.1 The interviewed persons tries to embellish their company

It might be tempting for the person being interviewed to make things look better than they are. Here we believe that it is actually an advantage to be in the same company as the person we interview. The person knows that we will not buy embellishment of the situation. We also think that it is more likely that the person being interviewed will share also negative things because he or she knows that we will not try to make the company look worse than it is (for obvious reasons).
8.3.2 Lack of objectivity

Especially with us working at the same company as the persons being interviewed, there exist an increased risk that we will focus and take notice only of problems that we have discovered ourselves. We have tried to avoid this pitfall by recording all interviews and then write them down before making any analysis of the data. We have also tried to speak as little as possible ourself during the interviews to avoid steering the answers.

8.3.3 We get what the person being interviewed thinks we want

There is always a risk that the person being interviewed gives us the answers he or she thinks that we want. We have tried to minimize this risk by giving the interviewed person some advantage [1]. All but one of the interviews were performed in the interviewed persons own office. The remaining one was performed on at least neutral ground in a small conference room. We also opened up the interviews with three comfortable questions [1] so that the person being interviewed could feel as much at ease as possible.

8.4 The company

All of the persons that we interviewed work at the same company as we do, namely Ericsson Software Technology (EPK). EPK is one of many Ericsson companies and its main business is value adding products for the mobile networks, e.g. pre-paid products, products for positioning of mobile stations, products for help with charging, just to mention a few. Since EPK is a telecommunication company almost all produced products have tough requirements when it comes to performance, availability, scaling, etc. The company uses the term ‘designer’ for people involved in writing code. Almost all work is project based. The projects usually contain somewhere between 5 and 35 persons although projects sometimes are combined together to form much bigger constellations each one with its own project manager though. The project manager is responsible for the project and typically works with tasks like planning, progress reporting towards superior managers, and coordinating the project both internally and towards outside interests. Even though the project manager works with internal coordination there usually exist a technical coordinator as well. The technical coordinator typically works with tasks like the overall architecture, resolving technical problems, supporting the project manager in the task of coordinating the internal work.

8.5 About the interviews

We have interviewed three Project Managers and three Developers with varying experience, both men and women. We will here give a short summary of each of the six interviews. In this summary we will concentrate on the two most important questions we used, “Which problems do you find to be the most troublesome in software development?”, and “What is the hardest problem in your specific role?”. From each category (Project Manager, and Developer) we have chosen one person with relatively long experience (might be considered short in a different engineering discipline), one person with some experience, and finally one with little experience. Each of the interviewed persons has been denoted ‘p’ for Project Manager or ‘d’ for Developer. The number attached to the ‘p’ or ‘d’ represents the experience level where 1 is the most experienced person and 3 is the least experienced one.

During the interviews we have discovered that the length of the interview increases with the experience. Since we have tried to speak as little as possible during the interviews we suspect that the difference has to do with the fact that more experienced people has identified more problems. It could also be due to that new people do not feel as much at ease as
the more experienced ones. The latter would if it were true have a negative impact on the results. This might lead to that problems that inexperienced people encountered would not seem as common as they are. In order to minimize this effect we tried to give all the persons that we interviewed an advantage by carrying out the interview in their own office.

8.6 Problem areas

From the interviews we have received 12 (2 * 6) statements about the most troublesome problems in software development. When trying to bring order to these statements we identified eight unique problem areas. This gives that in order for something to be classified as a problem area at least one of the interviewed persons has to mention it as one of the most troublesome problems in software development in general or as one of the hardest problems in his/her specific role. This might seem like low criterias, but we believe anything that is perceived by anyone as the hardest problem deserves to be included. For these eight areas we have looked for related information from every interview regardless if it was mentioned as one of the hardest problems or not. We will here give a short presentation of the identified problem areas.

8.6.1 The propagation problem

This problem has to do with propagation of information. Which information should go where? Can we identify which persons that should have a specific piece of information, and if we can, how can we make sure they get it?

8.6.2 The overview problem

The overview problem has to do with problems related to how to get a good overview of existing information as well as problems encountered due to lack of such overview.

8.6.3 The problem of incompleteness

This has to do with the problem that the information flow sometimes is slowed down due to that information is not/can not be propagated until it is in some sort of finished condition.

8.6.4 The coverage problem

This problem is closely related to the overview problem. Does our implementation really cover all requirements? Do our test cases cover everything that should be tested?

8.6.5 The responsibility problem

This problem has to do with unsatisfactory answers to questions like; who knows what, and who is responsible for this piece of information?

8.6.6 The planning/estimation problem

This is the problem that when estimating different work tasks, problems are not included although there always seem to be a lot of those.

8.6.7 The lack of project focus problem

This is as the name tells us the problem of lack of focus on a project. What should we do? Are we heading in the right direction?
8.6.8 The progress reporting problem

This is the problem of how to report and propagate information about progress in different work tasks.

8.7 Summary

We have used an empirical approach to understand which problems that are perceived as the most troublesome ones by the software community. Within this approach we have interviewed three Project Managers and three Developers with varying experience. All of the interviewed persons work at an Ericsson company in the south of Sweden, called Ericsson Software Technology (EPK). We divided their remarks about which problems that were the most troublesome ones into eight different categories.
9 Interviews

9.1 Project Manager (P1)

P1 is a relatively experienced Project Manager who was about to finish P1’s second project as a Project Manager. This might seem as a rather short experience but P1 had worked for six years in the software business as a total, which we consider to be rather long. Another reason for our classification is that both the projects P1 has been involved with and the project P1 is involved with are rather big ones. The project that P1 is currently involved with includes about 30 persons (within the own organization).

Hardest problem:

- “Estimations: They are always too optimistic because they only consider best case scenarios and do not include any problems although there always seems to be a lot of those.”

In your role:

- “The problem of which information to distribute to which people. It is not possible to distribute all information.”


<table>
<thead>
<tr>
<th>Problem</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The propagation problem</td>
<td>X</td>
</tr>
<tr>
<td>The overview problem</td>
<td>X</td>
</tr>
<tr>
<td>The problem of incompleteness</td>
<td>X</td>
</tr>
<tr>
<td>The coverage problem</td>
<td>-</td>
</tr>
<tr>
<td>The responsibility problem</td>
<td>-</td>
</tr>
<tr>
<td>The planning/estimation problem</td>
<td>X</td>
</tr>
<tr>
<td>The lack of project focus problem</td>
<td>-</td>
</tr>
<tr>
<td>The progress reporting problem</td>
<td>X</td>
</tr>
</tbody>
</table>

9.2 Project Manager (P2)

P2 has worked for four years in the software industry and for one and a half year as Project Manager. The project P2 currently is working with involves twelve persons.

Hardest problem:

- “To include all activities that must be done and to assign a responsible person for the, e.g. small portions of functionality. Progress reporting: where are we?”

In your role:

- “To include all activities that must be done and to assign a responsible person for the, e.g. small portions of functionality. Hard to know who is interested in what.”
9.3 Project Manager (P3)

P3 has worked for two and a half year in the software business and as Project Manager for about a year. The project P3 currently is working with involves eight persons.

Hardest problem:

- “The outside organization. Lack of focus on the project from the different stake holders.”

In your role:

- “Hard to hold project meetings. It is hard to give the right information.”

<table>
<thead>
<tr>
<th>Problem</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>The propagation problem</td>
<td>X</td>
</tr>
<tr>
<td>The overview problem</td>
<td>X</td>
</tr>
<tr>
<td>The problem of incompleteness</td>
<td>-</td>
</tr>
<tr>
<td>The coverage problem</td>
<td>X</td>
</tr>
<tr>
<td>The responsibility problem</td>
<td>X</td>
</tr>
<tr>
<td>The planning/estimation problem</td>
<td>-</td>
</tr>
<tr>
<td>The lack of project focus problem</td>
<td>-</td>
</tr>
<tr>
<td>The progress reporting problem</td>
<td>X</td>
</tr>
</tbody>
</table>

9.4 Developer (D1)

D1 has worked for three and a half year within the software business and equally long in D1’s current role as Developer. The project D1 is currently working with involves about twelve persons.

Hardest problem:

- “To know if the things one do is the things that must be done. Testing. Do we know that what we test really covers what is needed.”

In your role:

<table>
<thead>
<tr>
<th>Problem</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>The propagation problem</td>
<td>X</td>
</tr>
<tr>
<td>The overview problem</td>
<td>-</td>
</tr>
<tr>
<td>The problem of incompleteness</td>
<td>-</td>
</tr>
<tr>
<td>The coverage problem</td>
<td>-</td>
</tr>
<tr>
<td>The responsibility problem</td>
<td>-</td>
</tr>
<tr>
<td>The planning/estimation problem</td>
<td>-</td>
</tr>
<tr>
<td>The lack of project focus problem</td>
<td>X</td>
</tr>
<tr>
<td>The progress reporting problem</td>
<td>-</td>
</tr>
</tbody>
</table>
- “To synchronize with others. People who’s work interface with my own work.”

<table>
<thead>
<tr>
<th>Problem</th>
<th>D1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The propagation problem</td>
<td>X</td>
</tr>
<tr>
<td>The overview problem</td>
<td>-</td>
</tr>
<tr>
<td>The problem of incompleteness</td>
<td>-</td>
</tr>
<tr>
<td>The coverage problem</td>
<td>X</td>
</tr>
<tr>
<td>The responsibility problem</td>
<td>-</td>
</tr>
<tr>
<td>The planning/estimation problem</td>
<td>-</td>
</tr>
<tr>
<td>The lack of project focus problem</td>
<td>-</td>
</tr>
<tr>
<td>The progress reporting problem</td>
<td>-</td>
</tr>
</tbody>
</table>

9.5 Developer (D2)
D2 has worked for one and a half year within the software business and equally long in the current role as Developer. The project D2 is currently working with involves about 30 persons.

Hardest problem:
- “To get hold of the persons you need outside the project. People involved with the previous system etc. Who knows what and who can be of help.”

In your role:
- “Hard to get hold of people for discussions about different design matters.”

<table>
<thead>
<tr>
<th>Problem</th>
<th>D2</th>
</tr>
</thead>
<tbody>
<tr>
<td>The propagation problem</td>
<td>X</td>
</tr>
<tr>
<td>The overview problem</td>
<td>-</td>
</tr>
<tr>
<td>The problem of incompleteness</td>
<td>X</td>
</tr>
<tr>
<td>The coverage problem</td>
<td>-</td>
</tr>
<tr>
<td>The responsibility problem</td>
<td>X</td>
</tr>
<tr>
<td>The planning/estimation problem</td>
<td>-</td>
</tr>
<tr>
<td>The lack of project focus problem</td>
<td>-</td>
</tr>
<tr>
<td>The progress reporting problem</td>
<td>-</td>
</tr>
</tbody>
</table>

9.6 Developer (D3)
D3 has worked for half a year within the software business and equally long in the current role as Developer. The project D3 is currently working with involves eight persons.

Hardest problem:
- “The lack of a person with a good overview.”
In your role:

- “I did not know who to ask about specific parts of the system. Who knows what? I was not aware of all documents that existed.”

<table>
<thead>
<tr>
<th>Problem</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>The propagation problem</td>
<td>-</td>
</tr>
<tr>
<td>The overview problem</td>
<td>X</td>
</tr>
<tr>
<td>The problem of incompleteness</td>
<td>-</td>
</tr>
<tr>
<td>The coverage problem</td>
<td>-</td>
</tr>
<tr>
<td>The responsibility problem</td>
<td>X</td>
</tr>
<tr>
<td>The planning/estimation problem</td>
<td>-</td>
</tr>
<tr>
<td>The lack of project focus problem</td>
<td>-</td>
</tr>
<tr>
<td>The progress reporting problem</td>
<td>X</td>
</tr>
</tbody>
</table>

9.7 Summary

For a summary of this chapter we refer to the next chapter that contains both a summary as well as an analysis of this chapter.
10 **Analysis conclusion**

In this chapter we make an analysis based on the interviews to further see whether the first part of our hypothesis is valid or not. Our immediate reaction was that the interviewees seemed to agree on most things. There are only two areas which only one person mentioned. Below you can see a table containing a summary of all the interviews and for each problem area there is a number (0-6) telling us how many of the interviewees that mentioned that particular problem area during the interview.

<table>
<thead>
<tr>
<th>Problem</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The propagation problem</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>The overview problem</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>3</td>
</tr>
<tr>
<td>The problem of incompleteness</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>The coverage problem</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>The responsibility problem</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>3</td>
</tr>
<tr>
<td>The planning/estimation problem</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>The lack of project focus problem</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>The progress reporting problem</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 10-1:

It is interesting to notice that Project Managers and Developers seem to agree pretty much with each other. If two Project Managers mentioned a problem then at least one developer mentioned it too. Neither is there a problem that is only mentioned by Developers and not by Project Managers. The problem area that scored the highest was the ‘propagation problem’; which information should go where and how could we get it there?

10.1 **Project Managers**

When looking at Project Managers specifically we can see that they on average have mentioned more problem areas than the developers. We think that this is probably due to that it is part of their job to identify problem areas and try to minimize the problems there. The problem areas that scored the highest amongst Project Managers were the propagation problem, the overview problem and the progress-reporting problem. All of them are important parts of the work of a Project Manager but were also mentioned by at least one developer so there seems to be some consensus about them.

10.2 **Developers**

The problem areas that scored the highest among developers were the propagation problem and the responsibility problem. The propagation problem was also mentioned by all the Project Managers but the responsibility problem was only brought up by one Project Manager. The difference here could be the affect of the relatively small numbers of interviews. It was two Developers and one Project Manager that mentioned it but on the other hand the Project Managers mentioned a lot more problem areas so it could very well be that unclear responsibilities is something that Developers see as a bigger problem than what Project Managers do.
10.3 The problem areas vs. traceability

We will here go through the problem areas one by one to see whether they have to do with poor traceability or not, and if so, in what way?

10.3.1 The propagation problem

Here are some quotes from the interviews:

Comments from project managers...

Project Manager #1:

- "In my role as project manager I find the problem of which information that should be distributed to which people to be the most profound."

- “Then there is the problem of propagating the decisions ‘back’ to the project, to make all persons aware of what has changed and if this change affects them, and if their changes in turn affect others.”

Project Manager #2:

- “Sometimes one realizes that there exist things that should be propagated down in the project. It is hard to know who is interested in what.”

Project Manager #3:

- “It is hard to hold project meetings. It is difficult to give the right information.”

Comments from developers...

Developer #1:

- “Requirement changes are usually propagated by someone sending a mail saying that there exist a new version of the requirement specification. Often one forgets to take a look at the new specification to see if it applies to oneself.”

Developer #2:

- "Some times it has happened that I have had a certain version of the requirement specification and suddenly discovered that there existed at least one later version."

10.3.1.1 Traceability relation

We believe this is a traceability problem even though it might not be obvious at first glance. Who needs what information? The person/persons responsible for certain a piece of information is a part of the context information for that particular piece of information. In accordance with our traceability definition the easier it is to get to this context information the more traceability we got. Which other information pieces that depend upon it is context information as well. This gives that the more traceability we have got, the easier it is to find those affected by a changing information piece, e.g. a requirement.
When we are not dealing with changes to existing information then the problem is a bit different. If we for example say that someone identifies a completely new requirement for a project. This too is covered by our traceability definition. Who is affected by this new requirement is part of its context information even if it just emerged. As we have said before, our traceability definition does not only include the information pieces that we already have modelled, it also incorporates information pieces outside all of our models as well. This is true regardless of the reasons for why the information pieces are not represented in any model.

10.3.2 The overview problem

Here are some quotes from the interviews:

Comments from a project manager...

Project Manager #1:

- “There was not enough time to look at what other people did.”

Project Manager #2:

- “The second thing is the overview of the project or the lack of overview. Are all things included? Is everything working okay? Is it going alright for that particular person?”

Comments from a developer...

Developer #3:

- “Our project lacked a person with a good overview, someone that understood the bigger picture.”

- “I feel that there has not been so much information about the progress of other components or about the problems encountered. This has led to that one has not been aware of the work of others. If it would have been more visible one could have made more use of each other and get tips.”

10.3.2.1 Traceability relation

If we start with the comment from Project Manager #1; - “There was not enough time to look at what other people did.”. Here the project manager claims that it is lack of time that is the problem. The amount of time necessary to ‘look at what other people did’ has everything to do with traceability as we define it. Our definition of traceability includes the phrase “the degree to which it is possible and easy to get hold of context information...”, where time plays an important role in the easy part. If we somehow could make this type of information more accessible (i.e. increase traceability) then this would directly address this problem.

The comment from the second project manager includes questions like: “Are all things included?”, “Is everything working alright?”. These questions are discussed in the section about ‘the coverage problem’ but were quoted here to present the statement in the context in which it was given.
10.3.3 The problem of incompleteness

Here are some quotes from the interviews:

Comments from a project manager...

Project Manager #1

- “I do not want to receive the information when everything is decided and settled.”

Comments from a developer...

Developer #2

- “I had already implemented most things when I finally got the requirement specification.”

10.3.3.1 Traceability relation

This problem is absolutely a traceability problem. The problem is due to that there often is no traceability what so ever for information pieces under construction. For instance, a document usually must get to an okay state (which is decided by the creator) before it is ‘published’. If instead the work begun with attaching the empty document to the traceability web and then start working on it then this types of problems would be kept to a minimum. Documents in themselves present a traceability problem that we will discuss later.

10.3.4 The coverage problem

Here are some quotes from the interviews:

Comments from a project manager...

Project Manager #2:

- “One of the two biggest problems in the project work is to include all activities that must be done and to assign a responsible person for them, e.g. small portions of functionality. Often these things are left out of the planning until too late in the project.”

Comments from a developer...

Developer #1:

- “The hard problem is usually testing, to test the ‘right’ things and to know that the things one tests really covers what is needed.”

10.3.4.1 Traceability relation

The first remark has everything to do with traceability. This statement captures a lot of the essence of what traceability is (or in this case, is not), pieces of information not connected to anything and left out from all models. Having these types of information islands is always a huge risk. We lose control over projects and tasks and no one can say for sure if we are on track or not.
The second remark is also closely related to traceability. The ‘right’ things should of course be the requirements, directly or indirectly. If there exist a clear relation between test cases and requirements the risk of missing to test important features is greatly decreased. Of course it is still possible to write poor requirements and poor test cases but it is easier to get an idea about the average quality of the tests. If there exist requirements without any test cases attached to them it does not sound like good quality. In or experience just bringing these shortcomings up to the surface usually lead to that they are removed without having to assign each and every task to a specific person.

10.3.5 **The responsibility problem**

- Who knows what and who is responsible?

Here are some quotes from the interviews:

Comments from a project manager...

Project Manager #2:

- “An actual thing that occurred the other day was that someone had a look at a design specification and wanted to sort out some things in it. The person then looked at the name at the top of the document but that person had left the organization so he did not know who was responsible now, or who knew it now.”

Comments from developers...

Developer #2:

- “I feel that it has been hard to get hold of the persons you need outside the project. People involved with the previous system etc. Who knows what and who can be of help?”

Developer #3:

- ”I did not know who to ask for information about framework code used in the project. There existed information, I just did not know it, and I definitely did not know where to find it.”

10.3.5.1 **Traceability relation**

The first comment has to do with traceability in the way that it is about context information that does not keep up with the changes in the real world. When this situation occurs it constitutes a lack of traceability. There is no longer a clear and easy accessible path between the information itself (in this case the requirement specification) and the person responsible for it or the person in the organization that knows the most about it after that the author has left. This information is as we pointed out earlier part of the context information thus implying that we have poor traceability.

The second and the third remark also constitute lack of traceability because there do not exist an easily accessible path between what have been produced and the ones who have produced it.
10.3.6 The planning/estimation problem

Here is a quote from the interviews:

Comment from a project manager...

Project Manager #1:

- "The biggest problem when it comes to software projects in general is time estimations, always too little. Only the effective work is estimated not the problems. This leads to time pressure, which in turn leads to bad quality etc."

10.3.6.1 Traceability relation

This problem might not have an obvious relation to traceability but this problem is in our experience almost always due to that the estimations are not based on historical data. The reason for this in turn we believe is traceability in the way that this type of data is not easily accessible from old projects. You can maybe find old time estimations, but when you do it is not always so easy to see how much time that was actually spent on each work task. This too constitutes a lack of traceability.

10.3.7 The lack of project focus problem

Here is a quote from one of the interviews:

Comment from a project manager...

Project Manager #3:

- “I find the outside organization to be a big and difficult problem. There is a lack of focus on the project both from customers as well as from other stake holders.”

10.3.7.1 Traceability relation

This problem has no clear traceability relation. What can be said is that lack of outside interest might have to do with that it is hard to obtain information about the project. If it easier to access information about the project (i.e. good traceability) then more people might take the time. Still, there is no clear evidence that this problem is related to traceability.

10.3.8 The progress reporting problem

Here are some quotes from the interviews:

Comments from project managers...

Project Manager #1:

- “I feel that I have to request information (nag) in order to receive what I need. This can partly depend on the lack of routines to make it easy for people to report.”

Project Manager #2:
- “The other of the two biggest problems is progress reporting, especially progress reporting up in the organization. It is hard to get cold hard fact. Usually the progress reporting includes quite a lot of gut feeling.”

Comments from a developer...

Developer #3

- “On the other hand I feel that there have not been so much information about the progress and problems for other components.”

10.3.8.1 Traceability relation

All of these comments and especially the part in the second comment about ‘gut feeling’ have a relation to traceability. In our experience this ‘gut feeling’-problem has to do with lack of traceability between reported progress figure and the work tasks they address. The person reporting might very well have a different idea of what is included in the work package he or she is giving report about. - 60%, but of what? The person receiving the progress figures might not be aware of that e.g. three new work packages have been identified within this part of the system. Since the person reporting the progress figures is not sure that he is the one who are supposed to carry them out he just report the progress of the part of the work that he last reported progress for.

10.4 Summary

Many of the discussed problems have obvious connections to traceability, which strengthens our hypothesis. As we have shown in fact all problem areas except for maybe the three with the lowest scores have a clear connection to lack of traceability. Since 16 of totally 20 points are clearly related to traceability this is a strong indication of that our hypothesis is valid.
11 A traceability framework

11.1 Introduction

Communication in the organization is begging for a system approach - at the very least, a team approach. Yet organizationally, communications activities have been scattered over many divisions within the organization (in public relations, training, personnel, marketing, and advertising, to name but a few). Thus, organizations are unable to take full advantage of ‘coordinated communications’ under one roof. [8]

We have earlier established that traceability is a problem in today’s software industry. We have also seen that communication problems are closely related to traceability. Then what can we do to improve the situation?

A number of good ideas have come up during the last decades but a major problem in our opinion is that there is little knowledge and understanding about a lot of these findings in the software industry. I does not matter how good ideas people come up with if they are not communicate to a major part of the people involved in the software industry. Papers and seminars might be a good way to communicate these findings to an interested elite but we believe that other methods must be used in order to reach the broad mass. We believe that tools and applications, if used right, can be of big help here. If ideas and findings are presented in the form of tools and applications it is much easier to comprehend and apply them. Of course bad applications that only visualize the work that should be done and not the underlying concepts are bound to instead decrease the productivity rate.

Then how should such a tool be constructed? Marlow’s [8] suggests that a system approach is needed if we would like to improve the communication flow within our organizations. Since we would like to use a system approach, the tool would probably be more of a framework than an application. We have brought up a number of problem areas in this thesis that should be considered when constructing such a framework. Factors that have been discussed so far in this thesis and that should be considered during the framework construction are:

- Communication problems
- Need for greater flexibility
- Need for more decentralization
- Modelling problems
- Document fixation

As we have seen, improved traceability is the key ingredient for most of these factors so it is really a traceability framework that we should construct.

In accordance with Gilb’s ‘multidimensional tools principle’ - “If your tools can’t operate in all critical dimensions, then your problems will.” [10], we need to design the framework so that it allows the users to model and structure the information in a free and multidimensional fashion.
11.2 Availability

A guiding star for the design of the framework should be Mattias Jarke’s claim that traceability should come as a side effect of the daily work rather than imposing additional bureaucracy [14]. Allen Thomas J.[1] further strengthens this claim in a discussion about why people prefer one information service before another. He claims that availability is by far the most important factor. The quality of the service only influences the decision to a very minor extent he says.

To achieve a high availability we chose web browser technology to navigate through the information model. In order to make the transition as smooth as possible to begin work with the framework it is necessary to make the browser work as expected when navigating through the model. You should for instance be able to bookmark the different parts in the model so that you do not have to start at the top every time. The forward button and the backwards button should work as expected so that it becomes easy and intuitive to navigate through the model.
Using the web technique has several advantages in addition to increased availability:

- You do not have to install a client on each workstation in your organization.
- It has built in multi-site support. Anyone located within the intranet can access and work with the model if desirable.
- Most people are familiar with browsers so it is easy to get people started.
11.3 The building blocks

“Software development can be regarded as model building.”

(Jacobson [15])

If we in accordance with Jacobson’s idea [15] would like to regard software development as model building, then which building blocks do we need? During software development we hear terms like requirements, goals, test cases, bugs, components, estimations, etc. Then, do these things have anything in common? Well, it is all pieces of information and they all have dependencies to other pieces of information. For example, test cases depend (hopefully) on requirements.

To begin our model building we would like to be able to enter any piece of information into our model. For this purpose we introduce a building block, which we call “node”. A node is in its essence a placeholder for information that makes it possible to connect different pieces of information together. A node can hold a description of an arbitrary piece of information. The simplest thing would be to enter the information just as a plain text. In order to give a little bit more power though and also because we already chosen to use browser technology, we decided to use HTML as the descriptive language for the information entities. This makes it possible to include pictures etc. as part of the information, but it does not force it. It is still possible to only use a plain text if desirable.

There exist almost an endless number of information types so it would be a utopia to think that we can have specialized building blocks for them all. It would be even more of a utopia to think that we could identify and construct them already before we start developing. This does not mean that we should not have specialized building blocks. It just means that we will run into situations where we would like to model pieces of information for which we do not already have specialized building blocks. In order to tackle this problem we introduce a non-specialized node. All specialized node types are in fact just extended versions of this one. This non-specialized node allows you to model an information piece using HTML. This node type also provides functionality to connect to other nodes (information items), which we will discuss further in the next section.
11.4 The connections

We said before that all information items relate to other information items. These relations are really what traceability is all about. We need some mechanism to model these relations so we add functionality to the node that also keep a list of which other information items it depends on and also a list of information items that is dependent of itself. The reason for having lists in both directions is so that we later will be able to traverse the model in any direction.

In order not to give users of the framework the possibility to go and lose these important dependencies we introduce a rule that makes it impossible to create an information item in the framework without connecting it to at least one other information item. When you are about to enter an information item to the model you must begin by telling the framework which other information item (node) you want your new node to be dependent on. This rule of course makes it necessary to have a start node provided by the framework itself. This top node is given the name “Common Ground”. This node is the only one that does not depend on another node. Now we have the possibility to structure information as a giant web, which is how it should be structured [5] p. 79.
11.5 Folders

In order to increase the structure in our information model we introduce the concept of folders. A “folder” is a special kind of node that does not need to contain any information in itself. An example could be a requirements folder. It does not contain any information in itself. Its sole purpose is just to be a point where requirements can be attached to achieve a good structure. In this way folders can be used to categorize information items even within information items of the same type. Since an information item can be connected to an arbitrary amount of other nodes it is possible that one information item can be categorized into two different categories. As you might have heard in a famous commercial spot on TV; a person can be both a dentist and a mother.

11.6 Operations

Each type of node defines not only a template for the information but also a set of methods that specifies what you can do with it. This is true for the folder nodes as well. For example does the project node have a method called ‘generateProgressReport’ that recursively checks all sub nodes for progress figures and puts together a report. Another example could be the ‘requirement folder’ node, which has a method for printing a paper version of the requirement specification. All recursive methods make use of the direction of the relations. They flow from the superior node towards the dependent one. Because it is possible to model circular dependencies these methods have a built in protection. When they reach a node for the second time during the same operation they just return.
11.7 Modifications

All connections can be changed at any time, including the original one. There is however one rule. A node must have at least one parent. This does of course not apply to the top node that is created automatically. If you want to remove a node completely you will have to delete it. Doing so removes not only the node itself but also all its sub nodes, which do not have parents anywhere else in the structure. Deleted nodes however are not completely erased but becomes invisible until specifically called for. The reason for this has to do with version management and undo operations.

11.8 Responsibility and authority

Each node has at least one responsible person. When the node is created the person creating it will automatically become responsible. This can be changed later if needed. CG does also contain a list of users including id, name, role, e-mail address etc. The responsible person can set ‘read’ and ‘write’ permission for other users or groups of users, just like it can be done for files and directories in a file system. We suggest a moderate use of this functionality. CG is after all intended to improve the visibility and thereby the information flow. To get back to the previous discussion, it is only the person responsible for a node that can add dependencies. This is to help create a commitment culture rather than a dictatorial culture. The mechanism with a responsible person is also used for propagation issues. If for example a requirement changes every person who has connected his or her node (e.g. a component node) to that requirement as a child will receive a mail describing the change: what, who, when, and why.

11.9 Object oriented approach on information

CG is intended to be used in an object-oriented fashion. The different types of nodes are structured in an inheritance hierarchy. This makes it possible to define methods on a super node that will be inherited by sub nodes. The most important ingredient in object oriented thinking is in our opinion information hiding. This has over the years proven to be an outstanding mechanism for structuring information intuitively. CG does only provide possibilities for an OO structure. It does not force you. It is totally possible to create all nodes on the second level with the top node as the only parent. Such an arrangement would of course be devastating for information sharing but it is a possibility. It might seem dangerous to give this power to the user (developer), but we believe that no one is better suited than the user to make these kinds of decisions. On the other hand we realize that it can be difficult to create a good structure especially the first time so the CG comes with a number of project templates. A project template contains nodes that might be good to have in a project model. This given structure is of course possible to change as seem fit.
11.10 How the work could progress

Someone gets the assignment to start investigating the possibility for a new product. The responsible person starts by creating a stakeholder folder and a requirement folder. The requirement folder is labelled ‘mobile war game’. The game is location based and the idea with the game is to have two teams with at least three persons in each where each person has his or her own mobile station. If some people in one of the teams are able to ‘catch’ a person from the other team within a formation where you can draw a line from one team member to another and no person in the attacking team is further away than 20 meters from the person they are trying to contain then this person is out of the game. Of course such a formation requires at least three persons. The game takes place within a game area. Anyone who steps outside that area is automatically out of the game. The game goes on until one of the teams have less than three persons left in the game. Each player can at any time look at a map in their mobile station display. The location of all game participants and the borders for the game area is drawn with the map as background. The idea is that the members on a team come up with some sort of a tactic and communication with each other, either directly or via the mobile station.

The responsible person begins to create a node called ‘requirements folder’, and below that, a lot of requirement nodes. Some of them based on information he has come up with himself, others from an investigation with teenagers, and still others from the own organization. Three stakeholder nodes are created and connected to the stakeholder folder, one called ‘own thoughts’, another called ‘teenage investigation’, and the last one ‘internal or-
ganization’. Each created requirement node is then connected to one of the stakeholder nodes depending on where it sprung. Two requirements were mentioned in an internal paper but also in the teenage investigation so those are connected to both ‘teenage investigation’ and ‘internal organization’.

Later he realizes that he needs to structure the requirement even further and creates two new nodes, one called ‘functional requirements’ and one called ‘non-functional requirements’. Both of these are connected to the ‘requirements folder’-node. Each requirement is then connected to one of these nodes and the previous connection directly to the node ‘requirements folder’ is removed. After this he clicks on the ‘requirements folder’ node and chooses to print out a snapshot of what he has got so far. The document is structured according to the sub categories given by the model, functional and non-functional requirements.

11.10.1 Project launch meeting

![Diagram of project launch meeting]

Final after that some more requirements have been inserted into the model and some more requirements sub-categories have been created a meeting is arranged where each attendant gets his or her own copy of the printouts. If the attendants are not too many and the company have a good computer projector the work could be done directly on-line while the meeting progresses. Then no printouts are needed except for preparations maybe. During the meeting it is decided to launch the project and a subset of the elicited requirements are chosen for the product. This leads to that a product node is created and below that one a project node called ‘1.0’. Below the ‘1.0’ node a new requirements folder is created and the requirements decided to be included in the project are connected to this folder. The rest of the requirements remain in the model both for historical reasons and because that new products or new versions of the product might implement them.

11.10.2 Project launch

Now a project manager is assigned the project and starts working with the project node. He also starts to register the people who have been assigned to the project. A group of technically responsible people start designing the system. They break it down into smaller
components with minimal interfaces between them. A component node is then created for
every component that the group comes up with and is finally assigned to a particular de-
veloper. That developer continues to divide ‘his’ component into classes and creates a
class node for each one he identifies. Since the work is being done incrementally the tech-
nical managers together with the project manager decide on some requirements that must
be included in the first increment. This is done in the way that an ‘increment one’-node is
created and the proper requirements are connected to it.

Each developer is then given the task of estimating the required time he will need to finish
his component. The estimates can either be done on component level or preferably on
class level. In either case it is possible to directly see the estimated total time for each
component. The project manager can also directly see the entire total for the project (as
can for that matter anyone else). Each developer, in discussion with the technical managers
connects his component to the proper requirements, not only for iteration one, but for
all of the iterations.

Each developer reports worked time and progress on the preferred level of abstraction and
the progress for the project becomes very visible. At any point in time it is possible to re-
estimate any part of the system. Each node keeps a log for the changes that has been made
to it, so it is possible to see how the estimates have changed over time and who did the
modifications.

Every person responsible for a component also starts creating ‘component test case’ nodes
that are placed under a node called ‘basic test’ located in turn under the component itself.
In this way you can also keep track of the test progress in an intuitive way.
Below the component the developer adds diagrams, texts, html-links and everything needed to understand what the component is intended to do and why it is designed the way it is. The component responsible for the authority checks for example has html links to sources about SSL (secure socket layer).

The system description is located below the project node and contains hyperlinks to the component descriptions so no information exists in two places at the same time, thus avoiding inconsistency in the information model. The component descriptions could even in turn have hyperlinks to the actual code.

The project also has a test manager. The test manager himself or together with his team creates a lot of ‘system test cases’ which are placed below a node called ‘system test’ in turn located below the project node. Each ‘system test case’ node contains a system test case and it is possible to report which ones that have been carried out successfully and which have not. For each test case that fails a bug node is created and this node is connected to the component responsible for the failure. Some of the bugs might be considered serious enough to perform root case analysis on. Each root case analysis is then connected to the proper bug.

It is only the imagination that sets the limits to what can be done. It is perfectly possible that the model might include functionality for critical path analysis etc. When the project manager shall report to higher-level management he only has to add a prose text about the status in the project as a model-node and chose the ‘generateReportMethod’-method on the project node. All numbers are taken care of automatically.

The CG also makes it quite easy to share components between projects or even products. You just have to create some additional connections.

11.11 The Common Ground approach

In the following chapter we will go through the problems brought up earlier, originating from existing literature and also from the empirical study. Can a traceability framework help with these problems, and if so, in what way? We will not only keep the discussion on an abstract level. For each problem we will also give an example for how it could be implemented into a concrete traceability framework (e.g. the Common Ground) to make it more understandable. We begin with the problems from the empirical study.

11.11.1 Problems from empirical study

11.11.1.1 The propagation problem

The propagation problem is as we have seen before the problem of who should get what. How can a traceability framework help here? Well, we earlier claimed that the propagation problem in fact is a traceability problem.

If we have an information model that shows how different pieces of information depend upon each other and also who is responsible for each piece of information then we have everything needed to propagate information about changes to existing pieces of information (e.g. modified requirements etc.) to the people whom it concern. When it comes to completely new pieces of information (e.g. a new requirement) then the problem is a bit harder solve. If for example a new requirement enters the picture then it is a person, probably a system architect, who must see to it that existing implementation nodes and if nec-
ecessary new ones hook up with the requirement so it is completely covered. For this work a traceability framework cannot provide more information than which developers that exist and probably how loaded they are. The rest is a creative process and not a simple traceability task. On the other hand a traceability framework could provide the possibility of subscribing on different information nodes e.g. ‘requirements’ and thereby receive information every time when e.g. a new requirement arrives.

In the ‘Common Ground’ each node contains information about the responsible persons. The default is that it is the person who created the node. The information includes an email address. It is possible to configure the framework so that all persons responsible for a sub node of the node in question gets an email informing them that the node has changed and possibly also how it has changed. Each person will get at the most one email regardless of how many sub nodes he or she is responsible for. The person making the modification will get no email at all. Turner says - “The project manager must define those who need the information, so that when people receive something they know they ought to read it” [22].

The Common Ground also provides a possibility to subscribe to different nodes. If you for e.g. subscribe on the folder node ‘requirements’ then you will be informed each time a new requirement is attached to that folder.

11.11.1.2 The overview problem

“...it is not enough to separate the relevant items from the nonrelevant ones; users are drowning in wheat. Instead, our task is to figure out which items to present to each user at a particular time, given his or her specific resource and task constraint.” [7]

According to Bornstein [7] we should structure our information so that each user can approach it from their specific viewpoint. Since it is impossible to figure out in advance who might be interested in what we must address the problem in a different manner. The user must be able to choose for himself or herself how to approach the information. In order to achieve this we must allow the user to navigate from an arbitrary piece of context information towards the information. This discussion has to do with the need for flexibility and adaptability towards the users. There exist of course another dimension of the problem. It is completely possible to structure information so that no one can get a good overview. Here we believe Parnas idea of encapsulation (data hiding) is the key to the solution. If we structure our information as small pieces built up from each other then we are using Parnas idea of encapsulation with its advantages. If we instead create large chunks of information, like for instance in documents, then we are not.

Using the CG it is possible to structure information in an object-oriented way like Parnas suggests. CG also allows you to model all traceability connections between your information pieces. You can navigate towards a specific piece of information from any piece of related context information, thus giving you the means to reach information in a way that make sense to you.

11.11.1.3 The problem of incompleteness

As we have discussed before, a lot of information is not propagated until it is finished. ‘Finished’ tends to be very late. A comment from the interviews; “I had already implemented most things when I finally got the requirement specification.”
The underlying problem here is that traceability is attended last and not first. The way it goes is usually that documents are written and approved or at least in ‘decent’ states before information about their whereabouts is published. If instead the work begun by attaching the information piece to the traceability web and then start working on it then this type of problems would be kept to a minimum. Waiting on this decent states makes the information flow lose speed.

In the ‘Common Ground’ it is not only easy to attach pieces of information into the existing information web - it is an absolute demand. A requirement or a test case becomes visible as soon as it is created. You do not have to wait for the entire specification to be written and approved before you can see ‘your’ requirements, test cases, etc. It is of course a good thing to know if a requirement or test case is approved. The ‘Common ground’ has a feature that supports this. It is possible to approve each requirement, test case, etc. by itself. The responsible person will probably still approve all of them at the same time but this finer grain still has a big advantage. If for example one requirement is changed after that all requirements has been approved, only that requirement will show up as not approved. The rest will still look valid (which they are).

11.11.1.4The coverage problem

A comment from the interviews: - “One of the two biggest problems in the project work is to include all activities that must be done and to assign a responsible person for them, e.g. small portions of functionality. Often these things are left out of the planning until too late in the project.”

In order to address this problem the traceability framework must make it easier to include “all activities that must be done and to assign a responsible person for them” or maybe even make it hard not to. This problem is of course also related to the overview problem discussed before.

As we said before, in ‘Common Ground’ it is not only easy to attach pieces of information into the existing information web, it is an absolute demand. The ‘Common Ground’ does not allow for creation of any item without attaching it to at least one other part in the information web. The CG does also provide very easy mechanisms for connecting e.g. test cases and requirements together and if no test case is attached to a requirement then we can safely assume that we have not covered all requirements in terms of testing.

11.11.1.5The responsibility problem

Finding the person who created, or who is responsible for a specific piece of information, is a traceability problem. A traceability framework must make it easy to model this type of information and also to modify it.

In the ‘Common Ground’ each node has a responsible. This will automatically be the person entering the information piece. This is possible because all persons working with the information have to log in. The responsible person can be changed later if desirable if not the information comes as a side affect of the daily work, which is the way to go according to Jarke [14].
Some of the comments from the interviews had to do with the problem that the person who was responsible did not work for the company any longer. In the CG it is possible to tell the system that someone is leaving. The CG will then produce a list of all information items in the web that are attached to this person and tell you to re-attach them to someone else.

11.11.1.6 The planning/estimation problem

When coming to the problem of planning and estimating we earlier said that this is traceability related in the way that the problem is often lack of historical data, or more to the point, lack of traceability so that it is impossible to find the data. A traceability framework could provide some help with the ‘the planning/estimation problem’, at least long term. Used correctly, future projects will be able to make use of data produced within the framework. Reusable data can of course be produced without a specialized traceability framework but we believe that it is harder. What a traceability framework cannot do however is fix old mistakes (i.e. mistakes made before it was put into action).

The CG cannot provide much help with your old data in accordance with what we said before, but it does provide powerful mechanisms so that at least future projects will be able to easily get this type of information from your project. On the other hand, the problem of planning and estimating is closely related to the ‘overview problem’ and the ‘coverage problem’ where the framework does provide a lot of support.

11.11.1.7 The lack of project focus problem

We said earlier that the ‘lack of project focus problem’ could be related to traceability in the way that it might be due to that it is hard to find and understand information from the project. If this is not the case, then a traceability framework will probably not be able to solve this problem even to a small extent. On the other hand, if it in fact is the underlying problem, then a traceability framework could have a significant impact on the problem.

The CG offers possibilities to structure information so that it is easy to navigate through it. It can also offer several entry points for the same information. This means that different categories of people or even single individuals can navigate to the information in a way that makes sense to them. The CG does not solve this by itself but it provides the tools for it.

11.11.1.8 The progress reporting problem

A comment from the interviews: - “The other of the two biggest problems is progress reporting, especially progress reporting up in the organization. It is hard to get cold hard fact. Usually the progress reporting includes quite a lot of gut feeling.”

With a traceability framework it is possible to replace a lot of the gut feeling. Instead the progress figures could be the product of mathematical calculations based on small work packages that are either finished or nor finished. In order to do this the framework must provide mechanisms to report progress in a very easy and straightforward manner. In addition to this the framework should provide automatic mechanisms for processing all progress figures reported from within the project.
In the CG all work packages can be represented. It is also possible to connect sub-work packages to them and doing so actually decreases the progress figures for the top work package. The work can be broken down to individual classes or even further. If the work packages are small enough it is possible to completely avoid reporting in % finished and thereby remove much of the ‘gut feeling’ that usually appears in this context. On a higher abstraction level there might be progress figures in % finished but these are just calculations of underlying binary (finished or not finished) progress reporting.

CG implements more features to help with the overview problem. For instance, the ‘project’ node comes with a method that recursively checks sub nodes for progress figures and calculates a sum. All that is needed is to click on the progress button for the ‘project’ node. The developers can report progress on an abstraction level that they seem fit e.g. class, component, or work package. This gives that the progress figures for a project can be up to date at all times, not just ones a week. Since one single piece of information only exist in one place in the CG, the developer just has to report the same figure one time. If the figure will be used for two different things the figure is available for both purposes but still only reported once. It is perfectly possible to structure the information in the CG so that one can see the progress for a single requirement, design part, test case, etc.

This is one problem area where a traceability framework really shows its value. It is easy to construct very powerful mechanisms for progress reporting within such a framework.

11.12 Basic communication problems

Many of the basic communication problems in the list we presented earlier and that we will go through here are strongly related to each other. We do not consider this to be a problem. In fact, we believe that this is an indication of that our hypothesis is correct. In the cases where a solution to the problem already have been discussed we will refer back to the section where it first was discussed.

11.12.1 The message end up in the wrong place

This problem can have several reasons. We mentioned before that the point-to-point addressing scheme puts the burden on the sender to be aware of both who should receive a message as well as the e-mail addresses for those people. If this type of information instead could be modelled and kept in a traceability framework, then this would most likely have a positive impact on this problem.

Common Ground implements a number of features working together to address this problem. Each node will have a responsible. Default is that the responsible is the same person who created it. This information is automatically connected to each node. This is possible because that the person use a unique identity when working with the Common Ground. The responsible could of course be changed to someone else than the creator, but this will require a manual action.

If for instance, when a requirement node is changed the CG automatically sends an e-mail saying what has changed and how, to all persons responsible for that particular requirement-node or one of its sub-nodes. Every person will of course get at the most one e-mail and the persons making the change is excluded. This gives that e.g. all persons responsible for an implementation part or a test case, connected directly or indirectly to this requirement, will be notified about the change.
11.12.2 The message is not noticed

Two reasons for this problem brought up earlier was firstly that we get a lot of non-relevant information (see previous section), secondly that even the amount of relevant information is to big to be dealt with all at ones. One reason for why we receive non-relevant information was discussed in the previous section. The problem with too much relevant information implies that we must find a way to present the ‘right’ information to each user at any particular time, given the specific task he or she is working with.

In the CG all information is modelled in a giant web. It is possible to close in on information from virtually any logical association. This allows the user to have information presented in different ways at different times. For example, a requirement can be reached starting from a bug via a test case via an implementation-node, or from corporate goals down to the specific project thus reaching the requirement.

11.12.3 The message is delayed on its way

The CG approach to this problem is the same as for the problems of messages ending up in the wrong places so we just refer to that section.

11.12.4 You get to many messages

This problem as well is addressed by the CG with an automated information service just as it was for the previous problem.

11.12.5 The message is not understood

The reason for this problem that we brought up earlier was the ‘basic’ traceability problem, lack of context information. In order to solve this we must be able to model and keep the association paths(connections) between our information pieces. Of course the context information in itself must also exist in our models.

The problem of lack of context information, or means to find it, is really the problem around which a traceability framework is constructed and also hopes to address most other problems. The CG provides functionality to enter any piece of information in an easy and intuitive fashion. It also drives us to model and preserve the association paths between the entities. In fact, it does not allow for an information entity to be entered without connecting it to at least one other entity in the information web.

11.12.6 The message does not appear in the proper context

The CG approach to this problem is the same as for the previous problem, ‘the problem of messages not understood’.

11.12.7 The message is misunderstood

The CG approach to this problem is the same as for the two previous ones.

11.13 Summary

We have here presented a conceptual traceability framework and also tried to show that it can be implemented in a concrete form. We have also looked into how the framework could address and help with the problems brought up in the empirical study as well as the communication problems brought up in existing literature. We saw that the framework in fact could provide help with most of these problems in an easy and natural way.
12 Summary

This thesis has been constructed around the approval or disproval of the following hypothesis:

*The majority of the really troublesome problems within software development has to do with poor traceability, either directly, or via communication, so a framework that drives us to define and preserve the connections between the different information entities such as: requirements, design decisions, test cases, estimation figures, etc. should be able to provide us with help on the majority of problems reported by the software community.*

During this thesis we have established that traceability is one key, if not the key, for improving the success rates in the software industry. This has been done following two paths. The first one through existing literature and the second one through our own empirical investigation.

As an underlying foundation we have established that there exist a problem to solve at all. The software industry is still struggling with a lot of problems. Several sources report (to say the least) discouraging success rates, or rather unsuccess rates. Success is still the exception when it comes to software development.

With the help of existing literature we have established a path from an increasing change rate to a greater need for good traceability.

This path starts with evidence in existing literature that point to that change, not stability, is the natural order in today’s global environment. In order to sustain the increasing change rate, organizations must become more flexible. Many suggest decentralization as a way to accomplish this.

We have explained how decentralization in fact increases the need for good communication. More than one individual has identified communication in itself as key factor for success in the software business. This is not a contradiction to our hypothesis since it suggests that communication problems are caused by poor traceability. We have reproduced a list of basic communication problems from existing literature that attempts to capture the underlying aspects of ‘the communication problem’.

We have discussed traceability as a concept and proposed our own traceability-definition, namely: *the degree to which it is possible and easy to get hold of context information of interest, for all information entities (e.g. requirements, design decisions, test cases, reported bugs, progress figures, code etc.) in the given domain.*

We have gone through the list of basic communication problems mentioned before and established that almost all of them have a relation to poor traceability. In other words; poor traceability leads to poor communication, which completes the path from an increasing change rate to a greater need for good traceability.
We have further established that traceability is a key for increased success rates in software industry by performing our own empirical study. We began with a number of interviews in order to find out which problems that were perceived as the most troublesome ones within software development. The interviews were performed with project managers and developers at an Ericsson company called Ericsson Software Technology, located in the south of Sweden. These interviews showed that there for the most parts existed a consensus about which problems that were the hardest ones. We organized the problem remarks received during the interviews into eight categories.

For each problem category we have investigated how it is related to traceability. The conclusion is that 16 of the 20 remarks given within these problem categories had a clear traceability relation. This gives that both existing literature as well as our own empirical study points to that the underlying assumption in our hypothesis in fact is true. However, the part about the traceability framework was still to be investigated.

Having established that traceability really is a key factor in software development we looked at some of things that might make us loose the valuable connections (relations) between our information pieces. All the reasons we discussed had to do with poor modelling and over-simplification. Documents as a concept and the way we use them was brought up as an important reasons for poor traceability.

We have constructed and presented a conceptual traceability framework and tried to show that it is possible to implement this framework in a concrete world. An attempt to do so ourselves was carried out but had to be omitted due to the time budget for this thesis. You can read more about in the ‘Further research’-chapter. The traceability framework was constructed so that it would preserve and even urge us to model the vital connections between our information pieces. The framework was constructed on top of existing browser-technology, which makes it easy to get started. Further more we have given a description of how this traceability framework could be used in the everyday work.

We have also taken a closer look, into if and how the presented traceability framework could provide some help for the problem areas brought up during the interviews as well as the basic communication problems from literature. The conclusion was that in fact such a framework has the ability address most of these problems in an intuitive way, not suggesting that it would solve them all completely. This was the last piece necessary to claim that there exists some evidence for all parts of our initial hypothesis. We claim that this in fact shows that ‘the chain of success’ really does exist.
The contribution from this thesis we hope is a new, or at least broadened understanding of traceability and how it affects the everyday work performed within the software industry. We also hope that we to some extent may have broadened the perception of traceability towards something more than just requirements traceability. In best case, into something that should be considered for all tasks performed within our industry.

The chain of success

increased traceability

leads to

more effective communication

increased flexibility

higher success rates
13 Further research

In this thesis we have presented a hypothetical traceability framework. An obvious continuation would be to implement this framework and evaluate it from use in an actual software project. We have partly implemented such a framework with most of the features described in this thesis. This framework implementation was however omitted from this thesis due to the fact that we discovered that a good aesthetically appealing implementation was impossible to complete within the time-budget for this thesis. A just-enough implementation we come to believe would not be fair to the ideas presented here. However the implementation have had an impact on this thesis in the way that it brought up some problems that probably otherwise would have been left out e.g. how is it possible to navigate in an information web without getting lost, and can this be done in an way that feels intuitive. Our subjective opinion after the implementation is that it can be done, and it can feel intuitive. We did not run in to any conceptual blockaders but it is of course possible that a dry-run do not reveal all intricate problems that would have been encountered in a live-environment with a lot more users.

We do not imply that a framework implementation suitable for a ‘real’ project is gigantic but it was nevertheless too much work to be included in this thesis. We believe that a basic implementation of good quality could be done in less than 1000 man-hours if performed by a single individual. Our advice is to put a great effort into the browsing mechanisms, not because they are hard, but because they are important for the perception of the framework. When it comes to the implementation of the different nodes, start simple and easy and extend on demand by the users.

Another thing to investigate further might be to decide about whether the findings in this thesis apply to a wider range of projects and organizations than the ones found within the telecommunications domain. It has never been our intention to limit this thesis to that domain, but we ourselves are working within this domain, and so are the people that we have interviewed.

Relatively little work has been done around traceability in a more universal sense than just requirements traceability, so the field is wide open.
14 Appendix

14.1 Empirical study

14.1.1 Questions

- For how long have you been working in the software engineering business?
- For how long have you worked in your current position?
- How big is the project, which you currently work with?
- What do you feel is the biggest problem when it comes to working in projects?
- What is your biggest problem in your specific role?
- How do you feel about the information flow within projects?
- Do you have to request the information you need or is it being ‘pushed’ to you or maybe both?
- How do you feel about the requirements handling in relation to projects?
- How do you feel about the design work within projects?
- How do you feel about the test work within projects?

14.1.2 Interview 990701 - Project Manager #1

X has worked six years as a total and six years in the software business. X has worked 3 years in X’s current position which is Project Manager. The project, which X is involved with right now has been at the most 30 persons.

The biggest problem when it comes to software projects in general is according to X time estimations, always too little. Only the effective work is estimated not the problems. This leads to time pressure, which in turn leads to bad quality etc.

X finds the information flow in X’s first project rather good. Everyone was located at the same place. In the current project X has been involved with X thinks there has been a lot of problems with the information flow. X feels that there is a lot of discussion going on at high management level without letting people in further down the hierarchy (designers). When the decisions are taken they are often poorly under built and based on information received by scratching on the surface. When the people further down shall implement it, it often turns out to be a lot more complicated than expected.

In X’s current role X finds the problem of which information to distribute to which people to be the most profound. It is not possible to distribute ‘all’ information. If one tries to ‘collect’ a picture of what is happening this might appear as withholding information. On the other hand one must try to ‘protect’ the designers from all rumors. All this is an act of balance because it is not possible or wise to sit on all information and not let anyone know about it. This can be a problem. What is useful information for whom? The same thing in the other direction. All comments from all persons cannot be propagated to top management. It is hard to know who wants what.

X feels that X has to request or pull (nag) to receive the information X needs. X says that this can partly depend on the lack of routines to make it easy for people to report. This should probably be decided and implemented in the beginning and in some way force it, then it will run more smoothly later on. X is very interested to here also rumors etc.
cause often there is something to it. For instance if two different solutions are discussed this can be an important reasons for choosing one of them before the other instead of having to change everything after that a high level management decision has been official. X does not want to receive the information when everything is decided and settled.

X has experienced that the requirements usually are settled very late. Some of the requirements are very specific and one understands immediately what to do. Other requirements are very fussy and possible to interpret in anyway you like. The difference in requirements usually comes from varying knowledge about different issues by the person/persons writing it. The fussy requirements are usually due to that the person/persons writing the specification has not investigated the issue closely enough. The person writing these specifications could either be a system specification group or it could be persons involved with a prestudy for the project.

When a requirement is changed a Change Request is written and decided upon in a special meeting. The meeting where the decision should be taken often tends to be a discussion about alterative solutions by people with second hand information. X claims that these discussions should be held before these meetings and by the right persons. Then there is a problem to propagate the decisions ‘back’ to the project, to make all persons aware of what has changed and if this change affects them, and if their changes in turn affect others. The person directly affected by the change usually does not present a problem. Now this type of information is propagated from the Project Manager directly to the designers.

X says that the design work tends to slow down with an increased number of people because a lot of the time is spent straightening out misperceptions and misinterpretations and other problems caused by lack of communication. It is easier to make a smaller and closer team more effective. X suggests dividing the project into smaller teams working closely together. Each of these teams should have an assigned contact person probably also responsible for the team. X has used this approach in a project and it worked well for a while until time pressure broke it down. There was not time enough to look at what other people did.

X suggests that more time should be spent on testing. X experience is that it often works quite poorly. The that has not been involved in the design does not have the same knowledge of the weaknesses of the system and which parts that have caused the most trouble. The simple do not have enough knowledge. They test the ‘wrong’ stuff. They do not find the bugs. Maybe the testers should be involved in an earlier stage. X experience is that a better result has been achieved when designers have been involved with the testing. A high turn over amongst the testers has had a negative impact. An iterative work process has lead to that the testers have got a more immature product at an earlier stage of the development cycle. This has been hard to handle.

Geographical spread projects at least to beginning with and with inexperienced people is a quite limiting factor. Regarding the ‘Common ground’ numbers are not everything (a free text, a comment to each number). It must be simple to report otherwise you will get no information.
14.1.3 Interview 990705 - Designer #2

X has worked 1.5 year as a total within the software business and equally long as designer. X has also worked outside this field but only for shorter periods. The project X currently is working has involved at the most 35 persons.

X feels that it has been hard to get hold of the persons you need outside the project. People involved with the previous system etc. Who knows what and who can be of help? Might be due to our location.

X feels that the information flow as a whole within the project has been ok even though X have missed some information from higher up in the hierarchy, which are the potential customers, who has been promised what etc. Maybe more information about stuff like this should be distributed in a more formal way.

X has experienced a need for support from others just to discuss different matters. This need was not met at least not in the beginning. Design teams is a very good idea but one must think carefully before putting them together. It is probably a good idea to let the team members make the design together. Time pressure is probably the reason why it failed. Sometimes one just has a need to share stuff.

X both receives and requests information that X needs. The requirement specification has not covered all stuff. Most things were already implemented when X got the requirement specification. A real OPM has been missing. X says that requirement changes are usually not propagated at all. Some times X has had a certain version of the requirement specification and discovered that there exists at least one later version. X claims that X has no real knowledge of the company process regarding requirements handling. Who decides upon changes etc.? Maybe this is due to too many new persons in the project.

X says that design meetings are extremely important.

When it comes to testing X says that X does not know how to test X stuff. X has performed basic test but is not sure about what it is. Maybe a checklist for basic test could be a good idea. Of course memory leaks is a good thing to check but it is not possible to do this every time. It takes too much time. Maybe some sort of regression tests could be used. There have been a lot of things that X wants to test but lacks the equipment. A lot of the reported bugs are not possible to simulate so one has to guess. Some times X has looked for errors when they in fact the errors came from another subsystem next to the one X is working with. Maybe there should be a course in writing error reports.

X suggests a phone list of resources outside the project, a description of what shall be included in an error report.

14.1.4 Interview 990707 - Designer #3

X has worked for a half year in X’s current role and equally long in the software business as a total. X has previously worked for one year outside the software business. The project X currently is working with involves eight persons.

Few experienced people involved in coding. X says that the project has lacked someone with an overview (bigger picture). Too few design meetings, which have led to different structure in some components. Regarding the information flow X feels that X continuous-
ly has been updated of what is going on. On the other hand X feels that there has not been so much information about the progress and problems for other components. **This has led to that one has not been aware of the work of others. If it would have been more visible one could have more use of each other and get tips.**

X feels that the information flow from project manager has worked well but how the different components look now has not been propagated. X did not know who to ask for information about framework code used in the project. There existed information, I just did not know it, and I definitely did not know where to find it. **X did not know which documents that existed to be able to look for them.** Lack of information about information. X mostly has requested needed information.

X has not experienced any major problems with the requirements handling.

Design issues were discussed informally and orally.

X wrote a test specification for the graphical user interface. When the system test started the system was not ready for that. On the other hand the initial (basic) test worked quite well thanks to a good test tool.

### 14.1.5 Interview 990707 - Project Manager #3

X has worked for two and a half year in the software business and equally long as a total. X has worked as a project manager for almost a year. The project X currently is working with involves eight persons.

**X finds the outside organization to be a big and difficult problem.** Lack of focus on the project. Both from customers and other stakeholders.

X feels that the internal information flow within the project has worked well. X thinks that this is probably due to that all persons have been located to the same place.

**It is hard to have project meetings. It is difficult to give the right information.** X thinks this is probably due to lack of experience.

Much information has been received from the technical coordinator. Information about status etc. has been collect orally (in some kind of percent finished).

When it comes to requirements X thinks that they have not been detailed enough. No one really knows. A lot of the requirements have been quite ‘stable’ but there are a few, which have changed rather drastically.

Lack of experienced people has been a big problem. The test itself has gone quite okay. On the other hand X thinks that there has been a lot of shortcomings in the tests process.

### 14.1.6 Interview 990813 - Designer #1

X has worked 3.5 year as a total and equally long in the software business and in X:s current role. The project X currently is involved in has 10-12 members.
According to X one of the hardest things is to know if things one do is the things that should bee done. This is usually due to poor requirements. Some of the requirements are completely unnecessary.

The big problem is usually testing. To test the right things. To know that the things one tests really covers what is needed. Already during the design work consideration should be taken to testing.

X thinks that in X:s role the hardest thing is to synchronize with others but it works quite okay.

Requirement changes are usually propagated by someone sending a mail saying that there exist a new version of the requirement specification. Often one forgets to take a peak at the new specification to see if it applies to oneself.

Fifty percent of software development is to find out what to do and how to do it. X usually uses the web as the most important tool to look for information.

14.1.7 Interview 990819 - Project Manager #2

X has worked 4 years in the software industry and 1.5 years as a Project Manager, which is X current role. The project currently is working with involves 12-13 persons.

One of the two biggest problems in the project work is to include all activities that must be done and to assign a responsible person for them, e.g. small portions of functionality. Often these things are left out of the planning until to late in the project.

The other of the two biggest problems is progress reporting, especially progress reporting up in the organization. It is hard to get cold hard fact. Usually the progress reporting includes quite a lot of gut feeling. Recently X discovered that the used time in the project was 2000 hours below planned.

Small tasks are handled beside the larger project plan, and not within it. If they would have been included one could realize at an earlier stage if it is realistic to think that one will meet at certain time constraint. Some times one realizes that it might be things that should be propagated down in the project. It is hard to know who is interested in what. X thinks that most people have the information they need to do their job but maybe they could have done it a lot better and felt more like a group if they would have known more.

An actual thing that occurred the other day was that someone had a look at a design specification and wanted to sort out some things in it. The person then looked at the name at the top of the document but that person had left the organization so he did not know who was responsible now, or who knew it now.

The first thing X thinks is hard X’s role is the relation towards people above X in the organization. I do not know what they want says X.

The second thing is the overview of the project or the lack of overview. Are all things included? Is everything working okay? Is it working okay for that particular person?

X says that 95% percent of the information X has to request or collect.
A lot of time in each project is spent on writing test specification. New specifications for each project and the old specifications are not taken advantage of.
15 References


thesis, Department of Computer and Information Science, Linköping University, 1994


[27] Standish Group, Chaos, 1995

[28] Stefaine N. Lindstaedt, Kurt Sneider, Bridging the Gap between Face-to-Face Communication and Long-Term Collaboration, ACM

