Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology

Observations on Software Project Deviations

Examiners: Professor Kari Smolander
Researcher Uolevi Nikula

Instructor: Professor Kari Smolander

Lappeenranta, 10.12.2007

Minna Perttula
Korpisuonkatu 14 B 14
53850 Lappeenranta
050-3138108

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Department of Information Technology

Minna Perttula

Observations on Software Project Deviations

Master's Thesis

2007

77 pages, 18 figures, 5 tables and 1 appendix

Examiners: Professor Kari Smolander
Researcher Uolevi Nikula

Keywords: Deviation, software project, measuring, qualitative study

Software projects have proved to be troublesome to be implemented and as the size of software keeps increasing it is more and more important to follow-up the projects. The proportion of succeeded software projects is still quite low in spite of the research and the development of the project control methodologies. The success and failure factors of projects are known, as well as the project risks but nevertheless the projects still have problems with keeping the schedule and the budget and achieving the defined functionality and adequate quality.

The purpose of this thesis was to find out what deviations are there in projects at the moment, what causes them, and what is measured in projects. Also project deviation was defined in the viewpoint of literature and field experts. The analysis was made using a qualitative research approach.

It was found out that there are still deviations in software projects with schedule, budget, quality, requirements, documenting, effort, and resources. In addition also changes in requirements were identified. It was also found out that for example schedule deviations can be affected by reducing the size of a task and adding measurements.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan osasto

Minna Perttula

Havaintoja ohjelmistoprojektien poikkeamista

Diplomityö

2007

77 sivua, 18 kuvaa, 5 taulukkoa ja 1 liite

Tarkastajat: Professori Kari Smolander
Tutkija Uolevi Nikula

Hakusanat: Poikkeama, ohjelmistoprojekti, mittaaminen, laadullinen tutkimus
Keywords: Deviation, software project, measuring, qualitative study

Ohjelmistoprojektit ovat vuosien mittaan osoittautuneet ongelmallisiksi toteuttaa ja ohjelmistojen koon kasvaessa projektien järjestelmällinen seuranta on entistä tärkeämpää. Projektien onnistumisprosentti on edelleen jokseenkin pieni huolimatta vuosien tutkimuksesta ja projektinhallintamenetelmien kehittymisestä. Ohjelmistoprojektien onnistumistekijät ovat tunnettuja, samoin kuin projektien riskit, mutta edelleen projekteissa on ongelmia muun muassa aikataulun ja budjetin pitämisen suhteen sekä määritellyn toiminnallisuuden ja riittävän laadun saavuttamisessa.

Tässä työssä selvitettiin laadullisen tutkimuksen menetelmällä, mitä poikkeamia ohjelmistoprojekteissa esiintyy tällä hetkellä, mikä niitä aiheuttaa ja mitä projekteissa mitataan. Lisäksi työssä määriteltiin projektipoikkeama sekä kirjallisuuden että työelämän asiantuntijoiden näkökulmasta.

Työssä selvisi, että ohjelmistoprojekteissa esiintyy edelleen poikkeamia liittyen aikatauluun, budjettiin, laatuun, määrittelyyn, dokumentointiin, työmääriin ja resursseihin. Lisäksi projekteissa tapahtui muutoksia vaatimuksissa. Työssä selvisi myös, että poikkeamien esiintymiseen voidaan vaikuttaa etenkin aikataulun suhteen esimerkiksi pienentämällä tehtävien kokoa ja lisäämällä mittaamista.

TABLE OF CONTENTS

ABBREVIATIONS

ACWP	Actual cost for work performed.
BAC	Budget at completion.
BCWP	Budgeted cost for work performed.
BCWS	Budgeted cost for work scheduled
CPI	Cost performance index.
CV	Cost variance.
FP	Function point. A method used to measure the size of software.
SLOC	Source lines of code. The number of lines of code.
SPI	Schedule performance index.
SV	Schedule variance.
UML	Unified Modelling Language. A software modelling language.

FOREWORD

This thesis was made as a part of RIGHT project in Lappeenranta University of Technology.

I would like to thank the companies that participated in the project for cooperation. I would also like to thank the instructor of my thesis Kari Smolander and the examiner Uolevi Nikula for the advice I received during the research process.

I give my special thank-you to my husband Seppo for supporting me during the research process. Erityiskiitokset myös vanhemmilleni kannustamisesta opintoihin.

In Lappeenranta 10.12.2007

Minna Perttula

1 INTRODUCTION

Software project failure and success has been studied for years but still the rate of challenged or failed projects is quite high. Projects still exceed their schedules and budgets and don't achieve the quality requirements. Only 29 % of software projects succeed and 18 % of them fail according to Standish Group's study (Masticola 2007). The remaining 53 % of projects are completed over schedule and budget (Masticola 2007).

Projects face always uncertainty. Uncertainty can be categorized as a *risk*, a *change*, or a *deviation* depending on the impact level and the way of managing it. *Risks* have extensive impact on the project and they can be prepared for. The impact of a *change* can also be quite big but changes can't always be prepared for. Some events such as absence of employees due sickness are too small to plan for and to monitor one by one. Their impact may not be significant but when there are a lot of them, they start to have bigger effects on the project. There can also be events that weren't anticipated and that have a big impact on the project. The latter two cases are called *deviations*.

Project failure has been studied mainly in the viewpoint of project management (Ho Leung 1999; Haggerty 2000). Success factors of software projects have been studied (Reel 1999) as well as project risks (Keil et al. 1998; Sumner 2000; Wallace and Keil 2004; Wallace et al. 2004), and changes in projects (Ho Leung 1999). There are also several studies about measuring the software itself (Gustafson et al. 1993) and predicting the failure of software components (Nagappan et al. 2006; Schröter et al. 2006). Software project deviations are a less studied field.

The purpose of this study is to reveal the black spots of software projects today and find out what causes deviations. The value of finding the problematic areas in projects is that those areas can be better observed in the future and if the deviations are repetitive they can be prepared for. Also when the deviations are

detected as soon as they emerge it is easier to take corrective actions and put the project back on track.

This thesis was conducted as a qualitative study and the method that was implemented was grounded theory. The research process consisted of two phases in which over 30 expert interviews from four companies were analyzed. The purpose of the first phase was to produce the research questions concerning project deviations. In the second phase of the study two companies were chosen and more detailed interviews were conducted and analyzed. The purpose of the second phase was to form a general view of what deviations are there in software projects today and what causes them.

This thesis has been carried out as a part of RIGHT (Finding the Right Context in Globalizing Software Development) project which is a research project of Lappeenranta University of Technology (LUT) and South Carelia University of Applied Sciences. The objectives of RIGHT project are the following (RIGHT 2007):

1. To provide more understanding of how development processes and methods are dependent on the business context and its changes.
2. Development of a theoretical model or a framework of contexts, which will characterize different situations and contingencies the software development organizations face in their daily work.
3. To help and support software organizations in developing or selecting right SD methods and processes according to the changing business environment.

First in this thesis in Chapter 2 software project failure and the reasons for it, the differences between risk, change, and deviation are introduced. Chapter 3 focuses on project management and especially measuring software projects is discussed. In Chapter 4 qualitative research and its data collection methods are introduced. Chapter 5 describes the research process including data collection methods and data analysis. Results of the research are described in Chapter 6. Results of the first phase of the study are reported first in Chapter 6 and after

that the results of the second phase which includes two separate cases. Discussion and summary of the thesis are in Chapters 7 and 8.

2 SOFTWARE PROJECTS

This chapter concentrates generally on software projects and the failure of software projects based on the literature. Project risk, change, deviation, and their differences are introduced as well as actions on failing projects.

A project is defined as a task that is performed once and that has certain resources and an organization. A project is implemented systematically according to a timetable prepared beforehand. (Haikala and Märijärvi 2004) Merriam-Websters Thesaurus defines a project as a method worked out in advance for achieving some objective.

Software projects can not directly be compared to projects in other industries. There is no historical data of software projects because the discipline is quite new. Software projects also have bigger failure rate (see Figure 1) than traditional projects.

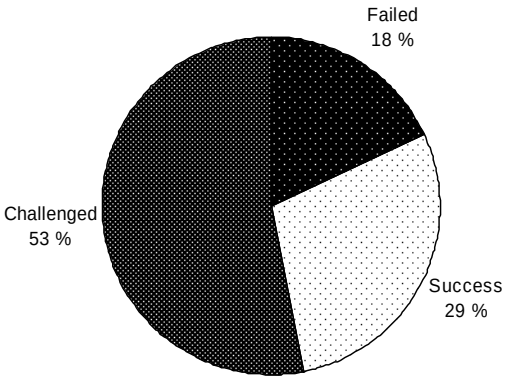

Figure 1. The success of software projects in 2004 (Masticola 2007).

Standish Group has examined the success of projects and the situation in 2004 can be seen in Figure 1 (Masticola 2007). In Standish Group's study a project is

considered *succeeded* if it is completed on time and on budget, with all features and functions originally specified. A *failed* project is never implemented or is cancelled before completion. A *challenged* project is completed over budget and over schedule.

The failure rate of projects varies according to how a *failed* project is defined. Some studies consider a project to be a *runaway project* if it has failed significantly to achieve its objectives and/or has exceeded its budget over 30% (KPMG according to Glass 1998c, 4). Standish Group's definition of *unsuccessful projects* – failed and challenged projects – contains bigger number of projects than KPMG's definition.

Several software projects still exceed their budget and timetable although the reasons for this are known. The same reasons in unsuccessful projects recur over and over again. One reason seems to be the lack of history knowledge, there is no measured information from previous projects to be used when planning a new project. Another reason for software runaways could also be that usually the project goes to the one that has underestimated the budget and the timetable most wrong. (Haikala and Märijärvi 2004, 54) Also Haapio (2007, 9) emphasizes that it is in the both parties' interest that the effort is estimated correctly in the bidding phase to ensure adherence to the project's budget and schedule. Liu (2006) emphasizes the importance of early detecting of problems in projects.

Some common reasons for software project failure are listed in Table 1. Numerous reasons for software project failure have been found and reported and only a few of them are listed in Table 1.

Table 1. Common reasons for software project failure.

Inadequate project planning	(May 1998; Jones 2004)
Lack of quality control	(Jones 1996)
Vague or unstable requirements	(Jones 1996; May 1998; Gaitros 2004)
Poor estimating	(Jones 1996; May 1998)
Inadequate measurements	(Jones 2004)
Inexperienced clients	(Jones 1996)
Inexperienced management	(Jones 1996; Gaitros 2004)
Poor user input	(May 1998; Gaitros 2004)
Unskilful staff	(May 1998; Gaitros 2004)

Project success has also been studied in the perspective of the size of the software (see Table 2 where the size of software is measured by function points (FP).).

Table 2. Software project outcome by size of project (Jones 1996, 4)

	Early	**On Time**	**Delayed**	**Cancelled**	**Sum**
1 FP	14.68%	83.16%	1.92%	0.25%	100.00%
10 FP	11.08%	81.25%	5.67%	2.00%	100.00%
100 FP	6.06%	74.77%	11.83%	7.33%	100.00%
1,000 FP	1.24%	60.76%	17.67%	20.33%	100.00%
10,000 FP	0.14%	28.03%	23.83%	48.00%	100.00%
100,000 FP	0.00%	13.67%	21.33%	65.00%	100.00%
Average	5.53%	56.94%	13.71%	23.82%	100.00%

As it can be seen in Table 2, when the project size grows also the risk of cancellation and delay grows. Small projects are much more likely to be on time and even early.

When a software project is defined failed, it usually fails with schedule, cost, or quality, or possibly all three (Jones 1996, xxiv; Liu, Kane and Bambroo 2006). Projects don't become runaways suddenly, there are always early warning signs that should not be ignored. Liu et al. (2006) highlights the lack of important project data and analyzing tools as the reason for the inability for early warning in a software project.

2.1 Project Deviations

The relation between risks, changes, and deviations is presented in Figure 2.

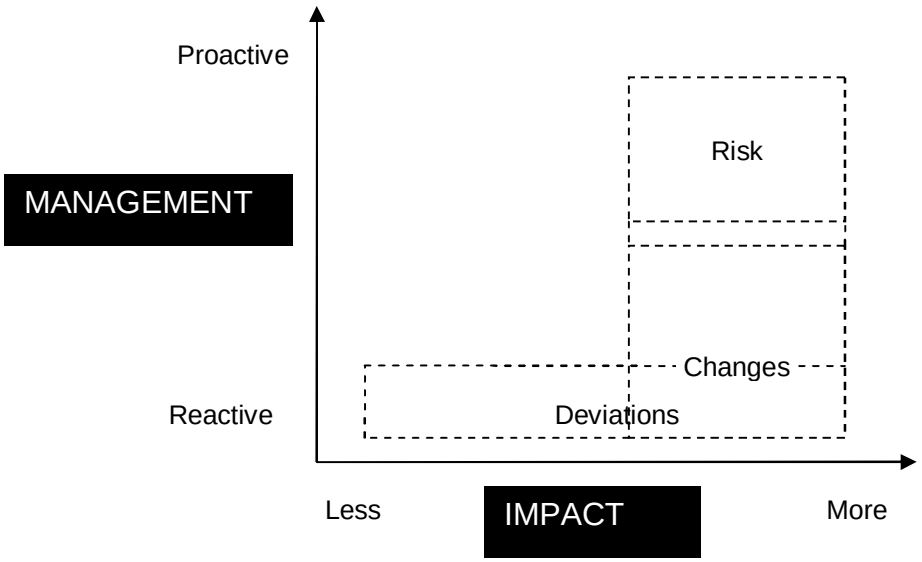

Figure 2. The relation between risk, change and deviations. (Hällgren and Maaninen-Olsson 2005)

The framework in Figure 2 is from a study (Hällgren and Maaninen-Olsson 2005) that analyzes deviations in an automation project but the definitions of risk, change, and deviation can be assumed to be similar also in a software project because the study is focused on project-intensive organization. A risk is according to Hällgren and Maaninen-Olsson (2005) known but not yet realized situation. Changes are handled reactively which means that they are not anticipated but managed when the situation materializes. Risks and changes recognize only the large impact situations and ignore those with less consequence on the cost, time and scope triangle. Deviations contain also the less consequence situations and thus a bigger number of situations is recognized than in a risk or change analysis. Changes and deviations are both identified in relation to a plan, changes deal mostly with the major project plan but deviation takes into account all plans in the project. (Hällgren and Maaninen-Olsson 2005) In this thesis a deviation is considered “a situation, regardless of consequence – positive or negative, large or small – that deviates

from any plan in the project" according to Hällgren and Maaninen-Olsson (2005).

2.2 Project Risks

A software project *risk* is a problem that is anticipated before the beginning of a project (Glass 1998b). One can be prepared to a *risk* by minimizing the probability of its occurrence and, on the other hand, by making a 'plan B' (Haikala and Märijärvi 2004, 238). Even if it were possible to predict all the possible disturbances that might occur in a project it wouldn't be wise because it would require a great deal of history knowledge and very few conditions (Krasna et al. 1998).

A framework for categorizing risks is presented in Figure 3. Risks can be categorized according to their range of action and their relative importance as well as the level of the control as can be seen in Figure 3. (Keil, Cule, Lyytinen and Schmidt 1998)

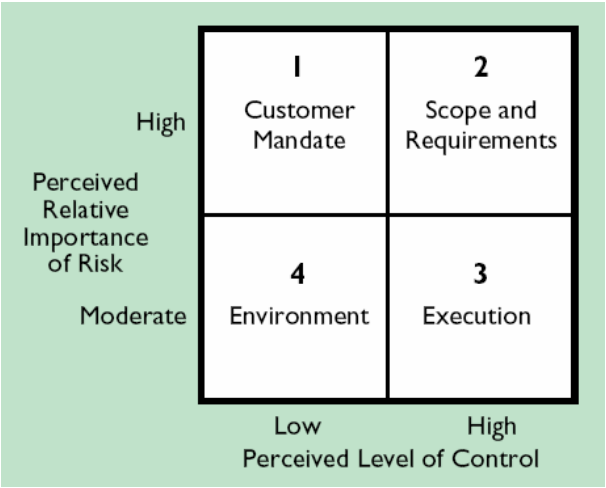

Figure 3. A risk categorization framework. (Keil, Cule, Lyytinen and Schmidt 1998)

Customer mandate category is a risk that can not be controlled very well and it can be an important risk. Customer mandate can not be controlled but it can be

influenced. Therefore good relationships with the customers and customer commitment are essential. Another important risk category is scope and requirements. It can be controlled by managing ambiguity and change. Execution category is classified as not so important risk because the risks in that category are usually quite well controlled by following an established development methodology and anticipating and responding to events that can threaten the development process. Environment category includes things that can happen inside the organization that would threaten the project and also external risks, such as changes in the competitive environment. (Keil, Cule, Lyytinen and Schmidt 1998)

2.3 Project Changes

Hällgren and Maaninen-Olsson (2005) classify *changes* as realized situations with a significant divergence to the project plan. They are handled when they occur, so they are reactive in their nature. (Hällgren and Maaninen-Olsson 2005) In this study, *a change* is considered *a deviation with a large impact on the project.*

Requirement changes are possibly the most common changes in a project and at least they have been studied a lot. According to Jalote (2002, 42) requirement changes can cost even 40% of the total cost and unhandled changes can have an adverse effect on the schedule and the quality. Another possible change in a project can be a change in the schedule, for example the customer wants the product sooner than originally planned.

Changes cause the project to be planned again and the possible outcomes of this planning are the following (Wysocki et al. 1995) :

- The change can be accommodated within the project resources and timelines.

- The change can be accommodated but will require an extension of the deliverable schedule.
- The change can be accommodated within the current deliverable schedule but additional resources will be needed.
- The change can be accommodated but additional resources and an extension of the deliverable schedule will be required.
- The change can be accommodated with a multiple release strategy and prioritizing of the deliverables across the release dates.
- The change cannot be accommodated without a significant change to the project.

2.4 Actions on Unsuccessful Projects

Glass (1998c, 238) introduces a view on problematic software projects. In a survey, companies were asked about their recovering attempts during a runaway project. Here are their answers in order of decreasing commonality:

1. Extending the schedule
   This is an easy and relatively painless method if the schedule was in the first place arbitrary.

2. Better project management procedures
   This method suggests that the management procedures that were used earlier were somehow inadequate.

3. More people
   Adding people to a project that is late has been said (Fred Brooks: The Mythical Man Month) to make the project even more late. This method is useable if the people that are added are already knowledgeable and experienced.

4. More funds

   This method is basically a result of whatever other method in this list except for numbers 6 and 11.

5. Pressure on suppliers by withholding payment

   This method can be used if the project is dependent on outside suppliers.

6. Reduction in scope of project

   It is not always possible to reduce the scope of a project but usually it is possible to defer some features.

7. New outside help

   This means outsourcing which is very popular in the computing business.

8. Better development methodologies

   The project was using bad methodologies or technologies. Changing a methodology or technology in the middle of a project may work but it may also cause the project to fail. This is not an easy method, adoption of a new methodology or technology may take some time.

9. Pressure on suppliers by threat of litigation

   See number 5.

10. Change of technology used on the project

    See number 8.

11. Abandoning the project

    This method will only stop the project consuming money and resources.

12. Other (Glass 1998c, 238-243).

It is very important to think about the lessons learned after the project and to collect history knowledge. Unfortunately, this is something that is not usually

done and the same mistakes recur over and over again (Glass 1998c; Haikala and Märijärvi 2004, 54).

Liu et al. (2006) introduces another view to prevent unsuccessful projects. According to Liu et al. (2006) there are three major problems with software projects: budget overrun, schedule overrun, and poor quality. Liu et al. emphasize the fact that usually the problems are detected too late to correct them. They introduce an early warning system for software quality improvement and project management as their solution for failing projects. The early warning system uses fuzzy logic based on an integrated set of software metrics. (Liu, Kane and Bambroo 2006)

Preventive actions can be taken to assure that the project does not fail. For example some people believe that maximizing the quality is economical. This is the “quality is free” thought (Crosby 1986). It means that as the costs of defect prevention are increased, the costs of rework decrease by larger amount than the prevention cost was. Then again, there are those who believe that it is possible to spend too much on quality (Slaughter et al. 1998).

3 PROJECT MANAGEMENT

This chapter concerns project management in general, phases of project management, project control methodologies, and measuring projects.

Project management is essential in today's software development. As the systems grow in size and complexity, more efficient project management methodologies are needed to keep the project under control.

The progress of a project can be normally described with an S-curve (see Figure 4). The S-curve describes the speed of project's progress.

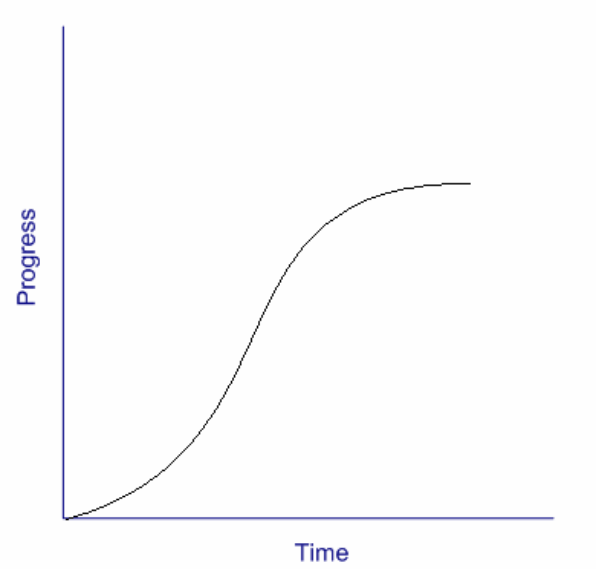

Figure 4. S curve of project progress.

As shown in Figure 4 the project starts slowly with a learning time when the team is forming and learning to work together. After that the working pace accelerates as the team stabilizes and it is able to work more effectively. In the end of a successful project the pace of the work slows down again as the final touches are put on the project. (Wysocki, Beck and Crane 1995, 49)

3.1 Phases of Project Management

Software development life cycle provides a clearly defined and repeatable process for developing software (Murch 2001).

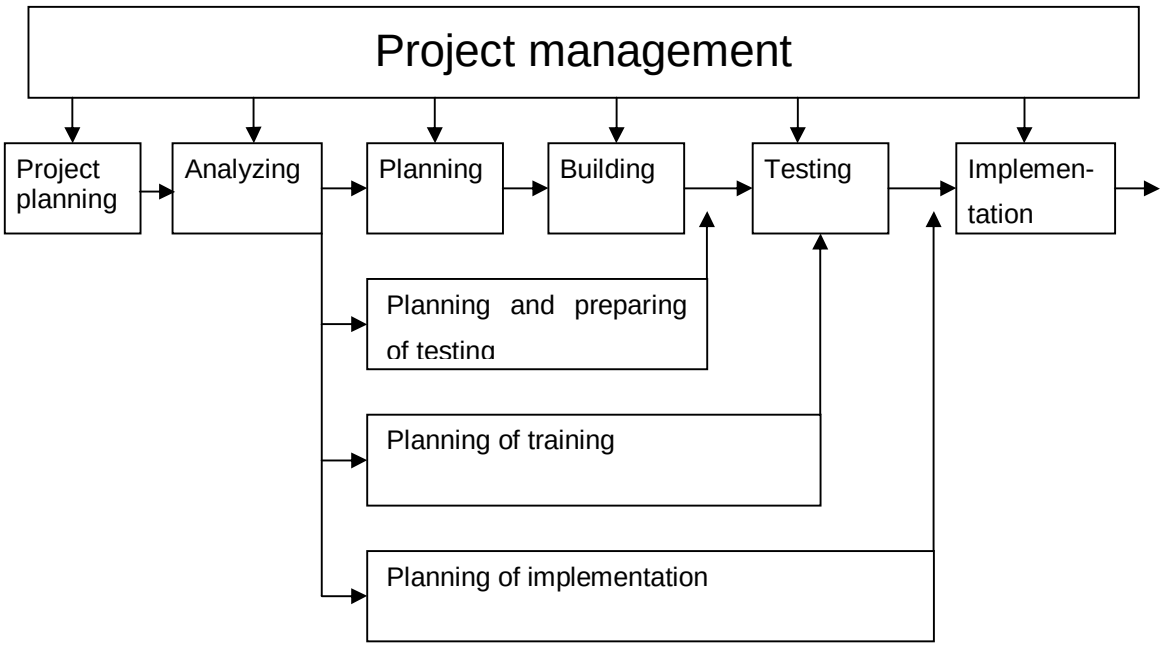

Figure 5. Software development life cycle.

Project management is present in every phase of software development which can be seen in Figure 5.

The phases of project management can be summarized from Haikala and Märijärvi (2004, 226-228) as follows. The first phase in project management is designing the project. This includes the goals of the project and the frame conditions. The goals should be of that kind that the evaluation is possible when the project ends. Frame conditions can for instance be the available resources and the timetable. Tasks that are related in the designing phase are organizing, focusing the goals, analyzing risks, choosing the technologies and the methods of working, designing the supporting actions, and partitioning of the project and

scheduling the parts. Partitioning the project and scheduling the parts are often the most problematic phases in designing.

The second phase is starting the project. It is good to have a start-up meeting so that the project is clearly started and all the participants know which project they work for and which is everyone's part in implementing the project. This is not always possible, for example when all the participants are not decided yet.

The next phase in a project is the follow-up and the instruction of implementation. Tasks in this phase are, for example, project meetings, project inspections, registering realizations and focusing the project plan when the project moves on. The goal of the follow-up is to detect deviations as early as possible so that they can be corrected.

The project can be ended with a termination meeting where a summary and the results of the project are presented. All necessary information for maintenance should be filed and a final report of proceedings, successes and failures should be written. Key figures should be collected for the quality system.

3.2 Project Control Methodologies

A methodology is defined as a group of repeatable processes that include project specific procedures, rules, and guidelines for building manageable systems that are value-bearing for the organization and of good quality (Murch 2001, 139). Project management benefits from the use of methodologies the following ways (Murch 2001, 140-142):

- Management can protect the investment by ensuring that the factors supporting the project are clearly defined and functional.
- Management and users know in advance what to expect from each project.
- The results are of high quality.

- Surprises such as budget overruns, changes in the extent, delay of implementation and other risks can be minimized.
- The progress information of the project is easily available.
- Profitability improves.
- Communication improves.

A good methodology integrates also other processes into the project management methodology as can be seen in Figure 6 (Kerzner 2001, 84). As it is shown in Figure 6 also change management and risk management should be linked to project management methodology. According to Kerzner (2001, 84) in the future companies can be expected to integrate even more of their business processes in their project management methodology.

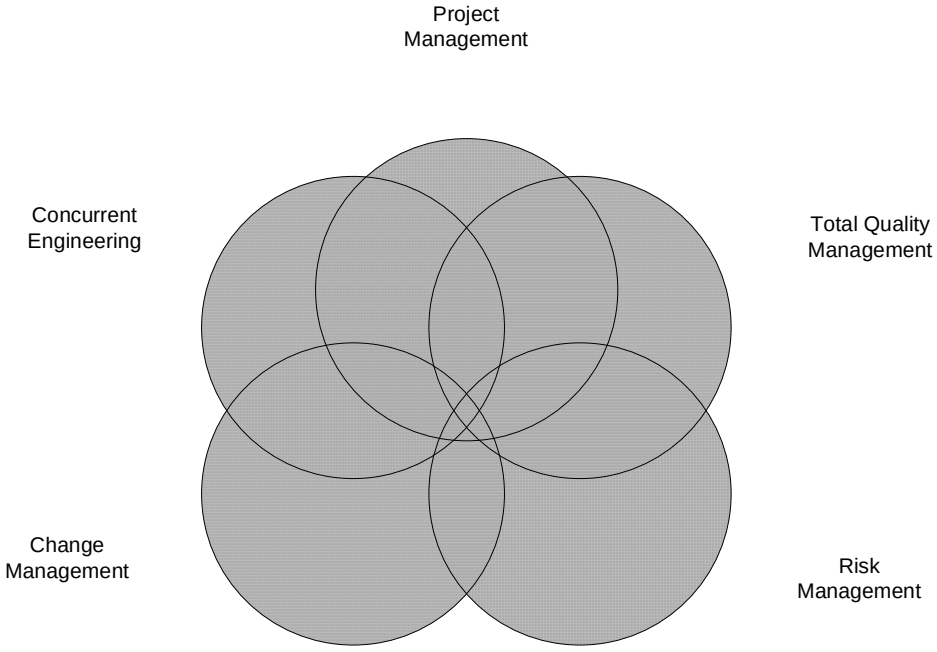

Figure 6. Integrated Processes. (Kerzner 2001, 84)

A methodology usually consists of guidelines, methods, tools, and models. Guidelines are the certain procedures that are necessary for successful software development. Methods are the detailed descriptions of processes that support the operation throughout the software development lifecycle. Tools mean especially such project control tools that are combined with a

methodology that is using experiences from previous projects. Models are the reusable documents and checklists. (Murch 2001, 139)

3.3 Project Follow-up

The actualization of the schedule and the cost can be followed along the project by calculating deviations between actual and estimated values (see Figure 7). To be able to calculate the deviations some measured data is needed.

- Budgeted cost for work scheduled (BCWS) is the budgeted cost that is defined for every task in the schedule. BCWS represents the budget of the tasks that were planned to be completed.
- Budgeted cost for work performed (BCWP) is the budgeted amount of cost for completed work within a given time period. BCWP represents the budget of the tasks that were actually completed.
- Actual cost for work performed (ACWP) is the amount that is measured as actually spent in completing the work within a given time period.
- Budget at completion (BAC) is the budgeted sum of all project costs.

The deviation between the actual and the planned cost at a milestone can be calculated with the cost variance (CV) equation (1).

$$CV = BCWP - ACWP \quad (1)$$

The deviation between the actual and the planned schedule at a milestone can be calculated with the schedule variance (SV) equation (2).

$$SV = BCWP - BCWS \quad (2)$$

From the former equations the cost performance index (CPI) can be calculated according to equation (3).

$$CPI = BCWP / ACWP \quad (3)$$

CPI tells about keeping the budget; if CPI = 1, the performance is as planned, if CPI < 1, the performance is poor and if CPI > 1 the performance is exceptional. Also schedule performance index (SPI) can be calculated as described in equation (4).

$$SPI = BCWP / BCWS \quad (4)$$

SPI tells about keeping the schedule the same way as CPI about budget. In Figure 7 the behaviour of BCWS, BCWP, and ACWP at a milestone and estimates at completion are shown.

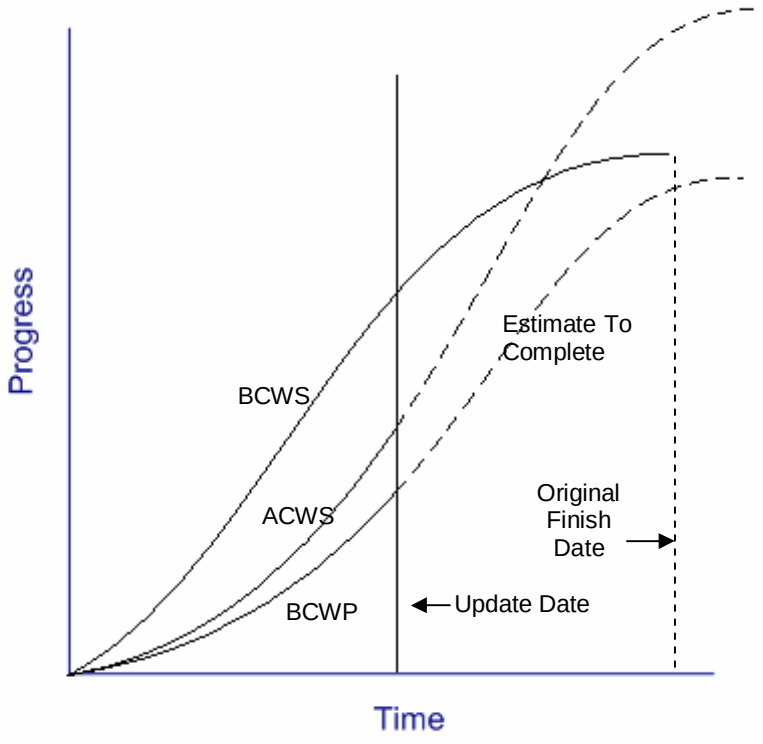

Figure 7. BCWS, BCWP, and ACWP curves (Wysocki, Beck and Crane 1995, 225)

If no corrective actions are taken when a deviation in cost or schedule is detected, the overrun will keep on growing as can be seen in Figure 7.

The cost and the schedule indexes are needed to estimate the completion schedule or cost. Estimate at completion (EAC) can be calculated for cost as in equation (5).

$$EAC = BAC / CPI \quad (5)$$

EAC can be calculated also for the schedule (see equation (6)) to see what the schedule will be at end of the project if corrective actions are not taken.

$$EAC = ACWP + (BAC – BCWP) / (CPI * SPI) \quad (6)$$

These estimates help to follow the project and to initiate corrective actions when a deviation is detected. Thus the project can be put back on track as soon as possible.

3.4 Measuring Software Projects

Jalote (2002) mentions controlling the project effectively as the basic purpose of measurements in a project. According to Jones (2006) most of the problems in software projects that are delayed or disastrous result from inaccurate estimation, inaccurate status reporting, lack of historical data from similar projects, and suboptimal quality control. These causes can be minimized or even eliminated by the adoption of formal estimation methods and tools, formal monthly status reports of both quantitative and qualitative data, collecting historical data, and improving quality control methods (Jones 2006).

Measuring projects is important also because people tend to work for goals which they know are being monitored. ”Rational, competent men and women can work effectively to maximize any single observed indication of success”.

(DeMarco 1982 pp. 58) Most surveys of software practice show that metrics are not much used in projects (Glass 1998a). By measuring projects it is possible to detect deviations. If a project is not measured at all, there is no data to which to compare the plans. DeMarco (1982) mentions that people tend to set delivery in the shortest possible time as the goal of the project if they are not given any other quantifiable goal - which is very common in software projects. Thus other goals that are probably very important for the project are forgotten (DeMarco 1982).

Commonly measured quantities in software projects are size, schedule, effort, and defects (Carleton et al. 1992; Grable et al. 1999; Jalote 2002 pp. 110). The project success can be measured by observing time, budget, and customer satisfaction (Mahaney and Lederer 1999).

A metric is a measurable indication of some quantitative aspect of a system and scope, size, cost, risk, and elapsed time are the quantitative aspects of a typical software project for which metrics are required. A metric, however, is not "…an observation expressed in numeric terms: 'Ninety-five percent of the programmers feel that we are more than fifty percent done'" (DeMarco 1982). A useful metric should be measurable, independent, accountable, and precise. A metric can be a "result" or a "predictor". A result is a metric of an observed value of a completed system and a predictor is a metric that is noted in an early phase of the project and that has a strong correlation to some later result. (DeMarco 1982)

3.4.1 Effort

According to Haapio (2007) estimating the effort in an early phase as accurately as possible is very critical on the account of keeping to the project's budget and schedule, and the success of resource allocation. If effort is underestimated, the costs and schedule overruns which weakens the customer satisfaction and causes financial problems to both the customer and the supplier. Also

estimating the amount of work for known tasks can be hard (Haikala and Märijärvi 2004). One possible way of estimating the effort is to use three estimates: pessimistic estimation (p), presumable estimation (a), and optimistic estimation (o). If the effort is considered to be beta distributed random variable then the maximum likelihood estimate can be calculated with equation (7). (Haikala and Märijärvi 2004, 244)

$$(p + 4a + o) / 6 \qquad (7)$$

There are some methods for estimating work loads and some of them are presented in Chapter 3.4.4.

3.4.2 Schedule

Setting milestones for a project is an easy way to follow the actual realization of the project schedule. Software Engineering Institute (Carleton, Park et al. 1992) mention that it is also important to define the exit or the completion criteria associated with each date because there are variations in the beginning and the ending activities so that, for example, one project may require that requirements analysis should be done throughout the project and another one may require it to be done at some certain point of the project.

Estimating the timetable for a project is problematic because estimating all tasks is difficult or even impossible. Estimating the schedule for a project is usually based on effort estimation that equals size estimation which is described in Chapter 3.4.4.

3.4.3 Budget

The biggest cost factor in software development is the cost of labour that can compose 50-70% of all costs (Haikala and Märijärvi 2004). In order to keep the budget it is thus essential to estimate the work load realistically and correctly. The calculation of the budget is based on historical knowledge, best estimates, or industrial engineering standards (Kerzner 2001, 828). Budget estimation is usually linked closely to size estimation and estimating techniques on size are described in Chapter 3.4.4.

Tightening the schedule of a project can increase the costs significantly as can be seen in Figure 8.

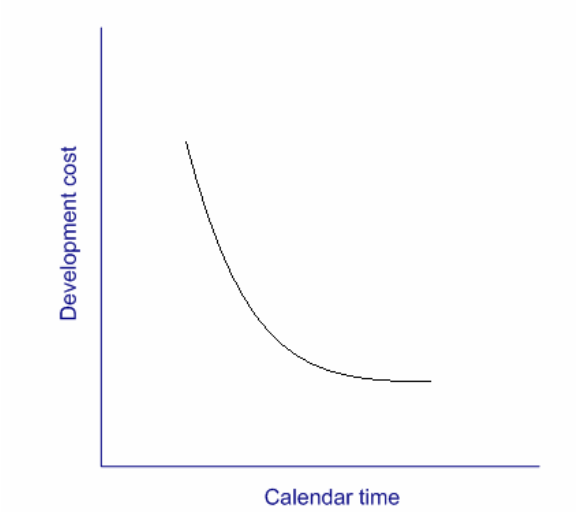

Figure 8. Effect of time on the development cost. (Haikala and Märijärvi 2004, 243)

Software projects don't divide into pieces too easily which causes that if the project's schedule is tightened the cost of development increases significantly as can be seen in Figure 8 (Haikala and Märijärvi 2004, 243).

When it comes to lowering the costs in a project it is easier to achieve cost reduction in an early phase of the project as shown in Figure 9.

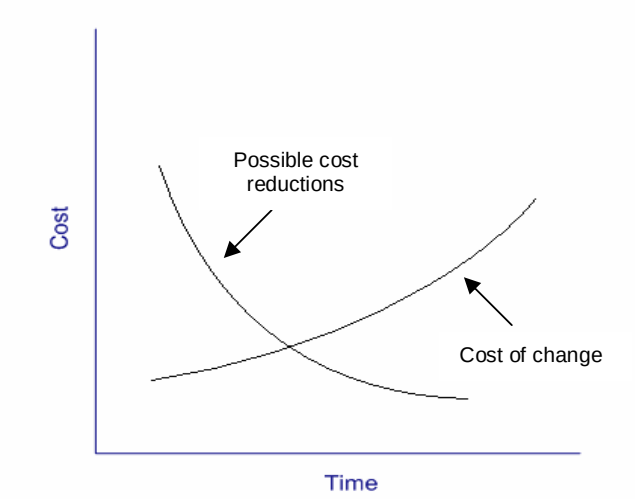

Figure 9. Cost reduction analysis. (Kerzner 2001, 819)

If the costs have to be reduced in a project, it is important to do it as early as possible because big cost reductions are possible only in the early phases of a project. It is essential to detect the deviations as early as possible when they can be fixed quite easily and the initial budget is still reachable. Changes should also be focused on the early phases of the project. They are much easier and cheaper to implement in the planning phase than in the implementation phase.

3.4.4 Size

Size of software is generally reliable predictor of project effort, duration, and cost (Chen et al. 2004). The most popular methods for software sizing have been for years the Function Point Analysis (FPA) and counting the number of Source Lines of Code (SLOC) (Chen, Boehm, Madachy and Valerdi 2004). Ferens (1998) states that “Software size is a key input to all software cost models”. For example Barry Boehm’s COCOMO-model (constructive cost model) takes the size of the software in SLOC and the cost drivers (the degree of difficulty of the project, the experience of participants etc.) and produces the amount of work in man-months and the ideal calendar time (Haikala and Märijärvi 2004). The COCOMO-model was introduced in the 80’s and has been developed during the years until the new version COCOMO 2000.II was

presented in 2000 (Haikala and Märijärvi 2004). There are also other software cost models such as SLIM, SEER, and Price/S (Chen, Boehm, Madachy and Valerdi 2004).

The function point analysis examines the functionality (number of inputs, files, outputs etc.) of the software. As a result the number of function points is achieved and the whole amount of work can be calculated when the amount of work needed to implement one function point is known (based on history knowledge). (Haikala and Märijärvi 2004)

The downsides of SLOC and FP sizing are that with SLOC it is not possible to accurately count the size of software until the software construction is complete, while the most critical software estimations need to be performed before the construction. Also the FP analysis has to be done manually and the estimator has to have experience. (Chen, Boehm, Madachy and Valerdi 2004)

Faster, cheaper, and more effective solutions for estimating the software size are being searched. One of the newer methods is Fischmans “UML-Based Software Sizing” presented in 1999. It estimates the software size from its Unified Modeling Language (UML) design. (Chen, Boehm, Madachy and Valerdi 2004)

The size metric can also include the number of document pages generated (Grable, Jernigan, Pogue and Divis 1999).

3.4.5 Quality

According to Kerzner (2001) customers demand higher performance, faster product development, higher technology levels, materials and processes pushed to the limit, lower contractor profit margins, and fewer defects/rejects. Quality is difficult to accurately define because it is defined by the customer.

Quality can be measured only when the quality requirements are defined. The common quality requirements are performance, reliability, usability, and flexibility. A quantifiable requirement is set to these requirements. (Murch 2001, 81) Quality can also be measured by defects or by customer satisfaction.

3.4.6 Defects

Defects in the software can be measured as indications of quality. Most project managers track defects at least during the testing and integration phases (Grady and Caswell 1987).

There are techniques for measuring how much effort defect repairs will require, how many faults remain in a system, and which modules will have defects (Mockus et al. 2005).

3.4.7 Customer Satisfaction

Krishnan (1993) emphasizes customer satisfaction as an important factor in retaining the current customers and gaining market share. Also Denning and Dunham (2003) are of the same opinion saying that if customers see little value in a product, they will not buy – and if they see value but are dissatisfied with their interactions with the company for sales, warranty service, or technical support, they will not return.

According to Grady (1992, 24) customer satisfaction can be measured by surveys and interviews. Krishnan (1993) uses feedback to measure customer satisfaction.

The attributes of customer satisfaction are functionality, usability, reliability, performance, and supportability (Grady 1992). These are almost the same as

the common quality requirements which is no surprise because quality and customer satisfaction are linked to each other quite closely.

3.4.8 Conclusion

This chapter concerned the phases of project management, project control methodologies and integrating other processes into the methodology. The ways of following the project's schedule and budget were introduced. The metrics of a software project were reported as well as software project measuring techniques.

4 QUALITATIVE RESEARCH

The baseline in qualitative research is describing real life. The research subject is endeavoured to be studied as comprehensively as possible. It is not possible to achieve objectivity in qualitative study in the traditional way. The objective of qualitative study is rather to find or reveal facts than to prove existing theorems. (Hirsjärvi et al. 1997, 152) Strauss and Corbin define qualitative research as any kind of research that produces findings not arrived at by means of statistical procedures or other means of quantification (Strauss and Corbin 1990, 17).

Qualitative study is not ready as the results are analyzed. They should also be interpreted. Interpretation means that the researcher discusses about the results of the analysis and makes conclusions of them. Finally syntheses of the results should be made. The syntheses collect the main issues together and provide answers to the research questions. (Hirsjärvi et al. 2002)

4.1 Data Collection Methods

The most common ways of data gathering are interview, observation, questionnaire, and written material. Questionnaires are usually used in surveys which are quantitative studies and the other three are usually used in qualitative studies.

Interviews can be divided in three categories; structured, semi-structured or theme interview, and open interview. Structured interview is made using a form which has the questions in a definite order. Theme based or semi-structured interview is a mix of structured and open interviews. The themes of the interview are known but there is no definite order or form. Open interview is the closest to normal conversation of the interview types. Open interview does not

have a solid framework; the interviewer has to guide the situation to the right direction. (Hirsjärvi, Remes and Sajavaara 2002)

A great advantage of an interview as a data collection method compared to the other methods is that the data collection can be adjusted flexibly according to the situation. There is also better possibility to interpret the answer more than for example in a posted survey. (Hirsjärvi, Remes and Sajavaara 2002, 192)

The reasons for choosing the interview as the data collection method in a research are that the area of research is little researched or unknown, it is known already at the starting point that the area of research will produce many-sided answers, the meaning is to achieve clarifying answers, or the area of research is timid or awkward. (Hirsjärvi, Remes and Sajavaara 2002, 192-193)

4.2 Grounded Theory

The grounded theory method was developed by Glaser and Strauss (Glaser and Strauss 1967) in the 1960's and later extended by Strauss and Corbin (Strauss and Corbin 1990). "The grounded theory approach is a qualitative research method that uses a systematic set of procedures to develop an inductively derived grounded theory about a phenomenon" (Strauss and Corbin 1990, 24). Grounded theory is suitable for studies that aim in uncovering the issues from the practice that may not have been identified earlier (Glaser and Strauss 1967).

Tesch (1990, 59) divides the qualitative research methods into four main groups concerning the research interest:

- the characteristics of language
- the discovery of regularities
- the comprehension of the meaning of text/action
- reflection

Tesch categorizes grounded theory in the field where the research interest is in the discovery of regularities (Tesch 1990). The purpose of grounded theory is to find regularities, identify elements and explore their connections as can be seen in Figure 10.

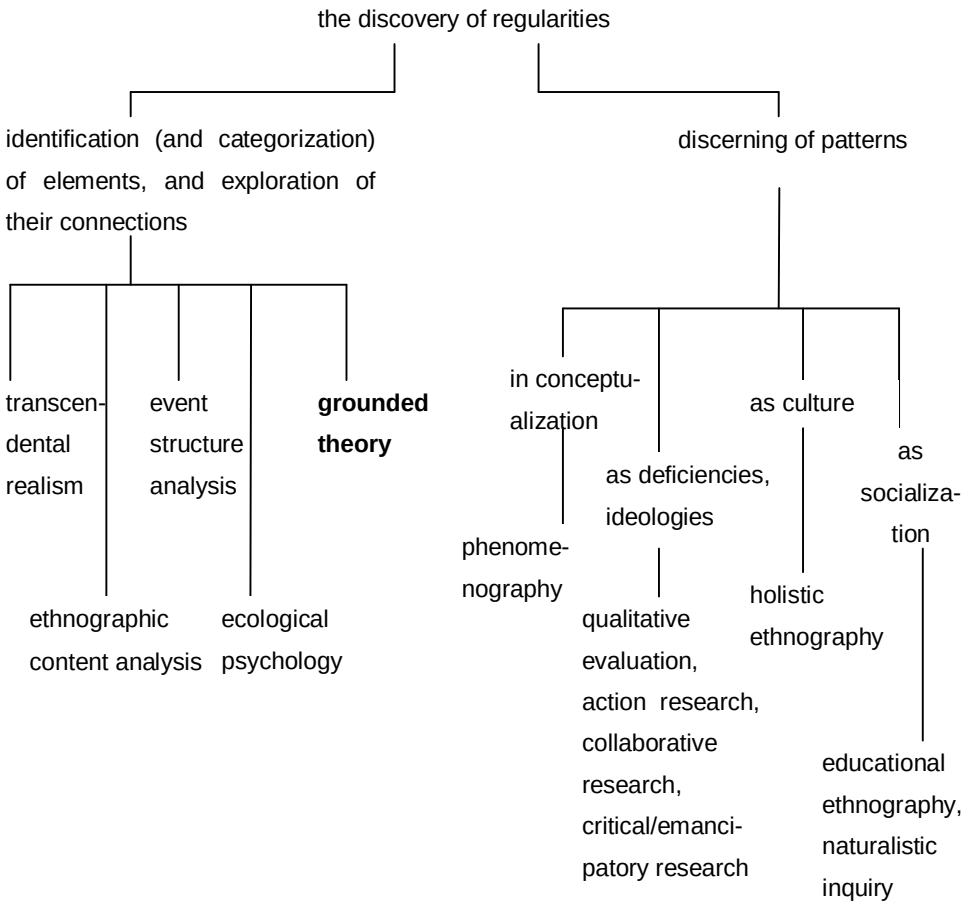

Figure 10. Placing of grounded theory among the types of qualitative research. (Tesch 1990)

Analysis in grounded theory consists of three main phases: *open coding, axial coding,* and *selective coding*. *Open coding* is the process of breaking down, examining, comparing, conceptualizing, and categorizing data*. Axial coding* is a set of procedures whereby data is put back together in new ways after open coding, by making connections between the categories. *Selective coding* is the process of selecting the core category, systematically relating it to the other

categories, validating those relationships and filling in the categories that need further refinement and development. (Järvinen 1999, 42)

4.3 Reliability and Validity

Validity and reliability are the factors that affect on the trustworthiness of the study. They are not self-evident in a qualitative study.

Reliability means the repeatability of the results of the measuring which means that it measures the ability of the study to give non-random results. Reliability can be found by several means; if two researchers find the same result, it can be considered reliable or if the same person is studied in different times and the same results are achieved, again the result is considered reliable. (Hirsjärvi, Remes and Sajavaara 2002)

Validity means the study's ability to measure exactly the thing that was meant to be measured. For example, in surveys it can happen that the respondents have misunderstood the questions and if the researcher analyses the results according to his/her original reasoning, the results can not be considered valid. (Hirsjärvi, Remes and Sajavaara 2002)

Reliability is often difficult to achieve in empirical studies. However, describing the implementation of the research improves the reliability. In interview studies, the conditions and the places where the interviews were made should be described. Also the time spent on the interviews and possible misreading should be reported. Classification is essential in qualitative studies, thus it is important to also report the root causes of classification. Analyzing the results requires the researcher the ability to weight the answers and to put them in the level of theoretical inspection. The analysis phase should also be justifiable, direct bits of interviews are helpful in that. (Hirsjärvi, Remes and Sajavaara 2002)

The validity of the research can be improved by triangulation. Triangulation means using several methods in a research or that several researchers participate in the study. (Hirsjärvi, Remes and Sajavaara 2002)

5 RESEARCH PROCESS

In this chapter the process of the research is described in detail. The research process is divided in two consecutive phases which are shown in Figure 11.

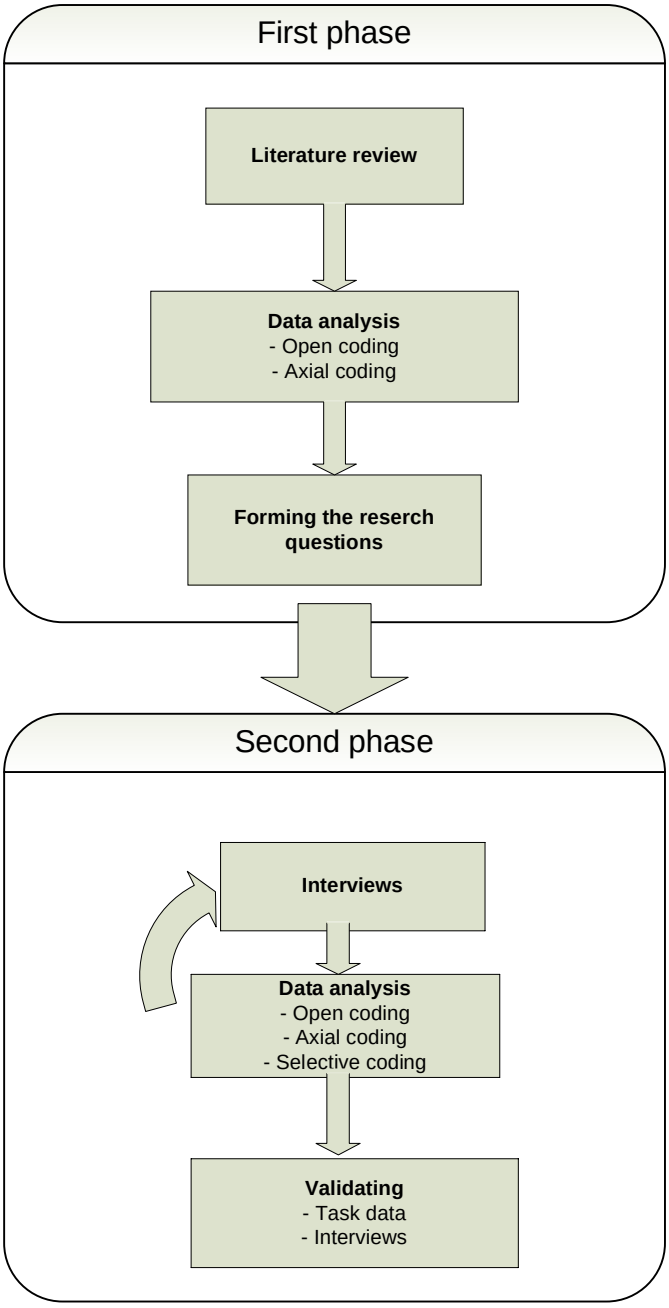

Figure 11. The research process.

The research began with a literature review where an overall picture was achieved. The next step was to analyze the first phase interviews with open and axial coding. Based on the two first steps, the research questions were formed for the second phase. The second phase started with doing more interviews in chosen companies and analysing them with open, axial, and selective coding. Interviewing and the data analysis took turns and proceeded in parallel. The validating phase consisted of task data and interviews based on the task data. The phases of the research according to Figure 11 are described in more detail in Chapters 5.1 and 5.2.

5.1 First Phase of the Research

The research began with a literature review that provided an overview of the subject. The next task in the process was to analyze the first phase qualitative data. These two stages guided the final formation of research questions.

The organizations from which the data for this research was collected were all software developing organizations (Table 3). The classification of size of the company is based on the recommendation of European Community for defining small and medium sized companies (EY 2003).

Table 3. Analyzed organizations.

Organization	Character of business	Size
A	Operative systems for industry	Big
B	Software development in central database systems with high availability requirements	Medium size
C	Automation systems for industry	Medium size
D	Internal systems development	Medium size

5.1.1 Data Collection

The data that was used in the first phase was collected by other researchers in the RIGHT-project and consisted of 32 semi-structured interviews. The interviews collected in the project were conducted in 4 companies. The interviewed persons had different work titles, among which several project managers, upper managers, and developers. In total the material consisted of nearly 1000 pages of text.

5.1.2 Data Analysis

The data analysis was carried out with Atlas.ti which is a program that is used in assisting on the coding and analyzing phases of grounded theory. The first phase of analysis was open coding. An example of open coding in Atlas.ti can be seen in Figure 12.

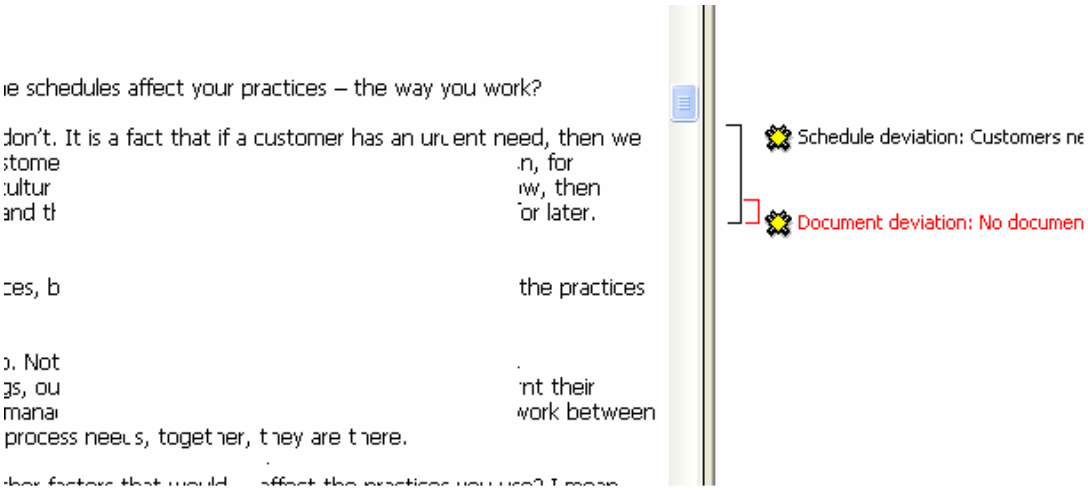

Figure 12. Example of open coding in Atlas.ti.

Along with open coding also axial coding was used in the first phase.

The categories did not saturate in the first phase, therefore more data was needed. The saturation of categories means that when no new codes appear to

a category from the data, the category is saturated. However the core category started to emerge already. Also the types of deviations started to form quite clearly.

5.2 Second Phase of the Research

Two companies were chosen for the second phase of research, companies A and B in Table 3 on page 37. The purpose of the second phase was to focus the concept of deviation and to reveal more precisely what kind of deviations occur in the projects and what causes them.

5.2.1 Data Collection

The data in the second phase consisted of two kinds of data. First interviews were conducted in the companies. The interviews were semi-structured and they were focused on deviations in software projects. The themes of the interviews can be seen in Appendix 1. The purpose of the interviews was to give more detailed information about deviations than gained in the first phase. The questions got more specified as the interviews went along and new knowledge was derived.

The interviewed people were project managers and other managers. In company A three interviews were conducted. The average duration of the interviews was 30 minutes. Company B provided also their task database from the time period of April 2004 and September 2007. This data was used as a reference material for the interviews in company B. Based on the task data, 3 semi-structured interviews were conducted. The average duration of the interviews was 50 minutes.

The chosen cases are not comparable to each other because their software development models are totally different. In company A the interviews covered projects that can last several years while in company B the study focused only on the small tasks.

5.2.2 Data Analysis

The collected data was analyzed as the interviews went along. The first task of the data analysis was to transcribe the audio recorded interviews. After that the transcribed interviews were transferred to Atlas.ti program. The next task of the analysis was to go through the data with open and axial coding. This time these coding techniques were implemented at the same time. The same categories as in the first phase of the research were still used but some new codes emerged. During the data analysis of company B, new codes were no longer detected which meant that the categories had saturated.

In the second phase the analysis consisted open, axial, and selective coding. The second phase data was first coded with open and axial coding and then compared to the first phase data with selective coding. Causal connections of categories and codes were formed according to companies (Figure 13). Causes of deviations cleared up with different network views in Atlas.ti as shown in Figure 13.

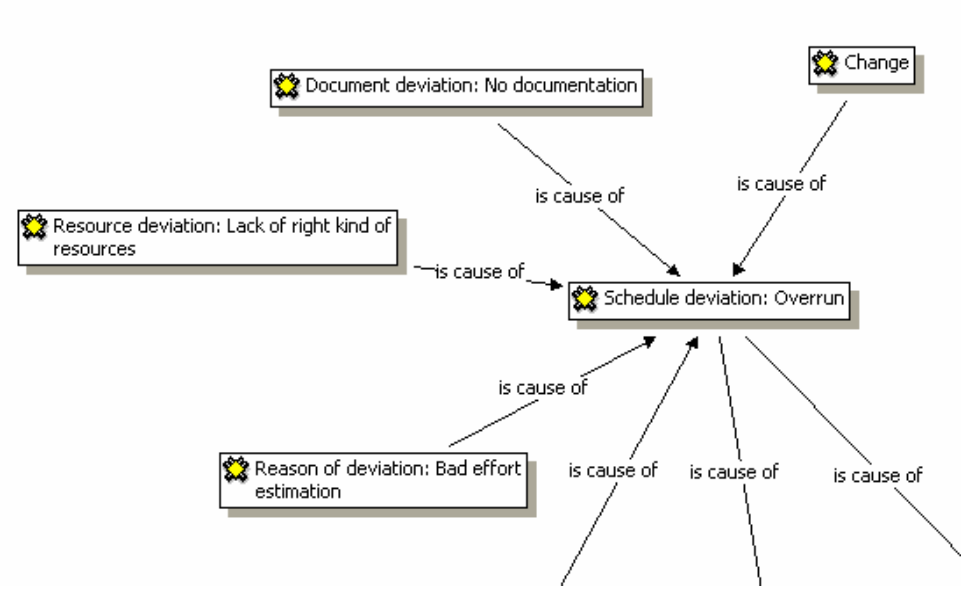

Figure 13. Outlining the causal connections with Atlas.ti.

.

Validating of the results was done with company B's task data and the interviews based on them. The interviews were analyzed the same way in both companies, except that company B's interviews were also examined based on the task data.

6 RESULTS OF THE RESEARCH

The results of the research are described in this chapter according to the two phases. The first phase results are described as a whole in Chapter 6.1 and the second phase results in two parts, both companies separately in Chapters 6.2 and 6.3. The research questions focused during the first phase analysis, so they are presented in Chapter 6.1.

6.1 Results of the First Phase

The goal of the thesis was to examine project deviations from two angles, first according to literature and secondly according to practitioners. The research questions arose along the literature review and the data analysis of the first phase data.

During open and axial coding ten categories were found and the codes were included in them. The identified categories were:

- Schedule deviation
- Budget deviation
- Document deviation
- Quality deviation
- Requirement deviation
- Effort deviation
- Resource deviation
- Reason of deviation
- Measuring
- Change

In quite an early phase of the data analysis it became evident that most of the deviations in projects concentrated on the schedules. There were also deviations with quality, requirements, documents, resources, effort and budget. In addition there seemed to be quite many changes that were defined as deviations with a large impact on the project.

The definition of deviation varied somewhat among the practitioners, so it was decided to specify. There seemed to be quite lot deviations with the schedule and little less the others. This guided to ask what kind of deviations are there. Also a lot of explanatory factors emerged from the data.

Considering these matters, the research questions for the thesis were formulated as follows:

1. What is the definition of project deviation in a software project?
2. Are there deviations in software projects?
3. What kind of deviations are there?
4. What causes deviations?

Based on the findings, guidelines about how these deviations can be prevented will be written. The mentioned questions will be answered in Chapter 6.2 and 6.3.

6.2 Results of the Second Phase – Company A

The categories saturated during the second phase of the analysis. No new codes emerged from the data. The categories are explained in Table 4.

Table 4. Explanation of categories.

Schedule deviation	Schedule overrun, too tight schedule etc.
Budget deviation	Budget overrun
Document deviation	Lack of documentation, insufficient documentation or insufficiently updated documentation
Quality deviation	Too much errors
Measuring	What is measured in projects and how
Change	Changes in definitions during project
Requirement deviation	Inaccurate requirements, deviations in functionality
Effort deviation	Misestimated effort
Resource deviation	Lack or change in staff or other resources
Reason for deviation	Different causes for deviations

Schedule deviation was chosen for the core category because all the other categories seemed to have connections to it and most deviations seemed to be related to the schedule.

The first thing that was examined from the data was how the projects are measured, what is actually measured and are the results of measurements analyzed. This is described in Chapter 6.2.1.

6.2.1 What Is Measured in Projects

Project metrics were divided in two categories in company A, the standard metrics and the project specific metrics. Standard metrics included schedule, effort, profit, number of errors, and customer satisfaction.

> "We have obligatory metrics that have to be (collected) in every project and well, if I recall correctly they are schedule, effort, customer satisfaction. They are obligatory, in addition of that it is possible to put other metrics that are project specific. "(Project manager)

The list is completed by another expert in company A.

> "Well, we have of course these, schedule, effort, profit, customer satisfaction and then… then of course the number of errors is being tracked. "(Quality manager)

An additional metric could be, for example, the effort caused by change management.

> "… sum of work from change management process. And that is a good metric because it's ratio towards actual work load shows how well the things have been planned. "(Project manager)

Customer satisfaction was measured by a questionnaire usually in the end of a project.

> "There is a questionnaire for customer satisfaction that is done in the end of a project. And if necessary they can be done before the end of project so that we know how the customer is feeling. "(Quality manager)

Company A analyzed the measured quantities mostly in the end of a project except for the schedule and the effort which were monitored along the project.

> "In principle yes in the termination inspection. Let's put it this way… let's say that during the project we track schedule and effort very specifically and in the end of a project also customer satisfaction. "(Project manager)

Another view of follow-up was from a quality manager who saw that the projects were followed monthly and the schedule was updated according to the progress so far.

> "We follow them in monthly level, how the job has been handled, how is it going. And then we have from the beginning of the project, we have the actual amounts and then in the end of the month we do a prediction for the rest of the project, how long will it take. That's how we follow them. It depends on the case of course,

> we can track some things more precisely, for example if we do some extra work we can follow the amount of the extra work, has it kept its budget or what…"(Quality manager)

Deviations that were identified were the same as in the first phase analysis: schedule, budget, document, quality, requirement, effort, resource, and change. The next thing that was analyzed was the factors that cause these deviations.

6.2.2 Effort Deviations

Lack of experience in effort estimation caused deviations between the estimated and the actual work loads. If the effort was estimated incorrectly, deviations could not be avoided.

> "Our effort estimations, they weren't correct. We made the FPA analysis but now we have found out that the analyses don't correlate at all with what happened in this project. "(Project leader)

Unknown or new technologies could also cause false estimations in effort.

> "… If w ego to a bit more unknown field of technology and implement with tools of a new technology, it may cause deviations regarding for example effort estimation. " (Project manager)

Deviations in quality could cause deviations in effort.

> "… the large number of errors and they were fixed. It becomes expensive of course and then the project overruns with effort and schedule. "(Quality manager)

6.2.3 Schedule Deviations

Schedule deviation was the core category of the research. Most deviations were related with schedule overrun.

Effort estimation was one major factor that could cause schedule overruns if not done properly or the estimation was incorrect.

> "We haven't been able to estimate the implementation demanding of something, or what effort it takes and what kind of know-how perhaps and in a way it is misestimating of perspective. We have assumed that something is straightforward and then we discover that it wasn't so simple after all. "(Project manager)

Another important factor causing schedule overruns was *change*. Change had two scopes affecting schedules: The customer wanted to have a change in the requirements or then something was not noticed in the specification phase and it had to be fixed.

> "not very easily because if we have already gone very deep in the time plan and are implementing programming something then the customer says 'oh I am sorry I have this and that 'and it is not easy in that sense any more ok if they understand to give the information before hand then it is of course possible to change it but if we announce that it takes that much of time and that is 2 months from now for example so then of course it ruins the time table" (Project manager)

> "One clear feature that always has to be taken into account during implementation and it is focusing the requirements. It seems to be a feature. No matter how hard you try to nail down and ensure on the writing desk, when the implementation starts, there are always this kind of things. "(Project manager)

Third major factor in schedule overruns was *requirement deviation*. When requirements were not specific enough, the implementation was difficult and the time scheduled for implementation was spent on defining the requirements again.

"When the requirement is bad, inaccurate and there are a lot of things to be clarified, even though we have the best coders put into this and good subcontractors, they can't do anything if they don't know exactly and if it hasn't been told clearly. They have to ask all the time what this means, what is behind this. The efficient coding power goes to clarifying things and the guys get frustrated quickly when they can't do what they do best. And the subproject managers get stressed too, they don't have time to help everyone, everyone is asking more information and clarification, more documents should be done." (Project leader)

There also seemed to be problems with *documenting* and *updating the documents.* All requirements or changes were not documented at all and in some cases there were several versions of documents.

"It's probably one of our weakest areas of all. In documenting… Actually because of its weakness in some projects we get into serious situations every now and then. In principle we handle documentation: we know that in this phase we get this kind of documents but what should be in those documents and how accurately, it is a little difficult area. There is no precise guidance for, no example templates. To know how to do these things is based pretty much on the professional skills and experience of the people." (Middle manager)

"… it had bad specification. First we started implementing based on that and then we made the technical plans against the database version and then the database changed and the specification wasn't fixed. So we did… We didn't have specifications according to which we should have done properly. And it caused errors." (Quality manager)

It could also be caused by the customer's actions, the lack of right type resources, or some external reasons such as licences or the schedules of the supplier.

"Usually it is, you could say that schedule deviations there are inner and outer factors. Inner factors are that effort or personnel that is, some expert is not available as much as we would have wanted, so the effort changes. Another one is

> outer factors, for example dependency of some licence, interface, or some other. Those are the ones that can affect. Well the third is of course that we don't get the client's decisions on time "(Project manager)

The demands of the customers could affect on schedule and budget so that they were tightened to the edge.

> "Also the customer can dictate the terms, this will be ready then and this is how much it can cost. If you don't agree, there are other suppliers too. … And it affects on the schedules. They are made tight and there is no room for surprices. "(Project manager)

Too tight schedule could cause rushing in some critical phases in the development process and not reacting to noticed deviations caused more serious trouble later.

> "Well, we did such a basic error there that we didn't admit on time, we should have put the cat on the table and admit that the specification is not ready, it will take 2-4 months more time to get it ready. "(Project leader)

6.2.4 Budget Deviations

Budget deviation was clearly related to the *effort estimation* in company A. If the effort was underestimated, the budget was likely to overrun. Also if the *quality* of the software was not very good and the product had a lot of errors, it caused more work and the budget overran.

> "Probably that causes it to reflect to the budget. Things are more laborious and that way the effort of the project increases and that reflects to the budget. " (Project manager)

> "That resulted into large number of errors and they were fixed. It becomes expensive of course. "(Project manager)

Another reason of budget overrun was the *technical changes* in the software implementation.

> "Of course a change of a technical solution is more serious, if we have to do some licence purchasing or something that we haven't agreed beforehand, then the budget changes can be more or less serious. "(Project manager)

6.2.5 Quality Deviations

Tight schedule was a major cause of also quality deviations. *Lack of testing* was another quite obvious reason for quality deviations.

> "Critical timetable, if there just isn't enough time, it shows in the quality right away. And then testing is another. The better you test, the better you catch the deviations. "(Project manager)

Quality deviation had effect on *customer satisfaction*.

> "Well of course if the work load is exceeded it causes pressure on the schedule and also customer satisfaction is put on the line. "(Quality manager)

Lack of proper requirement caused errors.

> "We didn't have a requirement according to which we could have done properly. And it caused errors. "(Quality manager)

6.2.6 Document Deviations

Too tight schedule seemed to be the major cause for document deviation. The schedule had two kinds of effects on documenting; the documents were done in a hurry and they were not specific enough or there was no time to update the

documents. Lack of resources could also cause documents to be incomplete or left without updating.

> "The requirements were also incomplete and a lot of information stayed just in the head, it wasn't documented. Subproject managers had only images about how things were agreed at some point in the conversations." (Project leader)

> "There is a rush; we don't have time to update the documents properly, how they have changed. … So we have three different things that we should synchronize and because we don't have time, resources for them, we have three different documentations of the software." (Project leader)

6.2.7 Changes

Changes in company A's projects were often caused by the customer. A customer could want to change something in the implementation phase or then all the vital information about customer's wishes was not achieved.

> Interviewee: "Yes and that is a challenge for us, it is very challenging for us to get the customer give us all the details in the beginning so that is very challenging"
> Interviewer: "So you have lot of changes because of incomplete information from the customer"
> Interviewee: "Exactly it is that is the reason for changes it is like a main issue." (Project manager)

Technical limitations could also cause deviations as well as forgetting something or not taking something into account in the requirement phase.

> "Yea I think so for example if you think that there is a major change in technology for example, and then you have already set with the customer that this is what you can do, and then there is a big change from outside to the project, that might be kind of thing that you have to do whatever you have to, although you know that you will get late in the project you just have to do it, for example, lets say there is

> something from law and I just forgot it and now it has to be done. It definitely has affect on the practice. "(Developer)

> "Technical limitations become deviations which are taken care of via project management or change management. "(Project manager)

If the requirement were not accurate enough for some reason, *changes* were to happen.

> "If we notice in some implementation phase that we would like to have something more that is the biggest problem. In some design phase we can always go back to beginning. But if we are already doing what we have agreed earlier, that is perhaps the biggest problem that we slip there. Suddenly we notice that we didn't take something into account and now we should have there. And when these wishes appear, it is an indication of requirements not being accurate enough. This is a common problem in these projects that we don't take all things into account. " (Project manager)

6.2.8 Requirement Deviations

Lack of experience caused deviations in the requirements. The people in the project didn't have experience about methods that were used, making requirements and documenting them. Also schedule affected in requirements.

> "A 'offers methods but we didn't have experience about those methods, how the specification is done, what kind of results it would make, and what should have been the level of documentation. Because of that even though we defined it [a long period of time] and specified those things, we got in a hurry and then it didn't… Now we can say that the requirements weren't specified enough. "(Project leader)

New or unknown technology could cause deviations in requirements.

> "Or some technical solution didn't after all work the way we had originally planned. " (Project manager)

6.2.9 Resource Deviations

Resource deviation in company A meant deviations with people and as is said in the following comment, people are the main resource in software business. Resource deviations happened when people working in the project left or got sick.

> "Resources like people those are the prime resources in this market. Computers stay, all other stuff stay, but resources won't. For example if you have a project in which you have 5 or 6 people, and 3 of them changes then you know that you will not be able to complete the project in given time or even in given budget because each project is unique and they usually handle specific information. So, for example, the project I did before, to set up the environment took 3 days so you can take it that for a new comer to put up the environment and to run the program it may take up to 2 weeks. "(Developer)

> "It is very common that these absences, for example people get sick or some other absences appear, so the project are usually quite tight with their schedule, so this kind of things have always effect. "(Project manager)

6.2.10 Conclusion of Company A

The factors that caused deviations and the results of deviations in company A are gathered in Table 5.

Table 5. Causes and results of the deviations.

Deviation	Cause	Result
Schedule	Effort estimation, change, requirement deviation, document deviation, customer's actions, lack of right type resources, external reasons, demands of a customer, not reacting to noticed deviation	Requirement deviation, more resources, priorization, effect on customer satisfaction
Budget	Effort estimation, quality deviation (too much defects), change	Outsourcing, more effective working methods
Document	Too tight schedule, resource deviation (not enough people)	Confusion, implementation is difficult or impossible
Quality	Too tight schedule, too little testing, the customer has unclear objectives, the requirement is not accurate enough	Effort overrun, budget overrun
Change	Customer wants a change, requirements are not accurate enough, forgetting something in the specification phase	Schedule overrun, effort deviation
Requirement	Too tight schedule, effort deviation, unknown technology, inexperience of making requirements	Change, implementation is difficult or impossible, confusion
Effort	Inexperience of estimators, unknown technology, quality deviation (too much defects)	Schedule overrun, budget overrun, the requirements are not accurate enough
Resource	People change work, people get sick	Schedule overrun, document deviation

It seems that schedule was the most dominant factor in company A's projects. All other deviations seemed to have effect on the schedule or then schedule related reasons caused them as can be seen in Table 5.

The customer had quite much power and was able to somewhat dictate the schedule of a project. This usually caused the schedule to become too tight.

Effort estimation was an important factor that caused problems later in the project. Bad effort estimation could be caused by inexperience. Incorrect estimation of effort caused too tight schedules and often when the schedule was tight, the requirements were specified in a hurry and remained incomplete. When the software was implemented according to defective requirements, the quality of the software suffered. It had a lot of errors or then the functionality was not what it was supposed to be. Fixing these and possibly specifying the requirements again during the implementation takes time and is expensive.

Tight schedule also caused that the documents of the project were left without updating. Unclear documents caused confuses in the project which was likely to cause changes in the schedule.

It is apparent that in company A a lot of different things are measured in projects. Mostly the measuring seemed to be collecting of history knowledge after the project had finished. The effort and the schedule were followed also during the project and based on the findings, the effort and the schedule were corrected for the rest of the project.

It seems that identifying deviations in projects could be improved in company A. It is important to identify deviations as early in the project lifecycle as possible and take corrective actions. To be able to identify deviations, effective measurement during the project and analysis of the results is necessary.

6.3 Results of the Second Phase – Company B

Company B uses a task model in a majority of their software development. Only a small portion of work is done as projects. The task model is a model for small-scale development where the tasks last from few days to two months.

> 'But after that (the transition to new task model) the major part of software development goes to the category 'small-scale development'. So it is from two

> weeks to two months: a new software, a new character, changes." (Upper manager)

> "So mainly, lets put it this way, the relation could be lets say 9/10 changes to the production is done with the task model." (Manager)

The difference between tasks and projects when it comes to deviations is that tasks are easier to handle.

> "I'd say that in the task model they are handled better on average. They are smaller pieces; I think that effects the most on it, they are easier to manage. But of course when it comes to projects, the entire wholeness will be late, so it shows a lot more. If you have a few projects and one of them is well late, the proportion is bigger. In that sense the task model is handled better and the task can be scheduled better than projects." (Manager)

6.3.1 What Is Measured in Projects

It seemed to be more difficult to create the correct metrics for the task model than for a project.

> "And then what comes from the production, we don't have a good meter, I have been trying to think about how we could catch the real, for example defects half a year after launching or anything. But they are probably easier to build in a project. It is a bit difficult to see what actually caused it because we divide the work in small pieces." (Manager)

The defects in task model were followed based on the whole system, not to the individual tasks.

> "So we look at them as a whole, if there appears a lot of reports from the field and a lot of corrections have to be made, then we look at what caused it." (Manager)

The finishing times were followed specifically in company B.

"Most of measuring is based on it (schedule) in one way or the other, so mainly finishing times, it tells about how effectively and well it goes through the pipe. And the number of tasks has to be adjusted to it in some level because they have clearly shrunk on their size but then again the efficiency has gotten better during the years. "(Manager)

The returning of tasks to the previous phase, for example from testing to programming was followed because there is a connection to the completion times.

"Well, how much reopen round there comes during the journey. Along the process, the ones that are returned backwards whether they go back to planning or programming, well in principle they are equal. … We follow them because they have a clear correlation to finishing times or how they are completed in the planned schedule. "(Manager)

Customer satisfaction was measured with a questionnaire once a year.

"So we have customer satisfaction questionnaires once a year. We follow it actively. "(Manager)

Measuring tasks was thought to be good although toilsome. It was thought to be beneficial and affecting for example on customer satisfaction. The task model insists on measuring because there is no project manager to look after the work.

"Toilsome. But necessary and useful. "(Manager)

"I think it's a good thing. I see that with the level of measuring that we have been doing for a few years we achieve good results. It shows in customer satisfaction among others. We achieve quite well the schedules, those that are a lot late, there are only few of them. And the measuring, to make this kind of a model work, it actually requires following, there is no project manager so the metrics has to built some other way. "(Manager)

Company B's task model does not take budget into account; it is separate from the task model. Hence there were no observable budget deviations.

> "Deviations in budget, we don't even try to manage them with this model. It's very task centered which means that there is no project schedule separately. And those things are usually discussed in another table and the work hours are written to another tool. They don't have anything to do with running the process. "(Manager)

Only the schedule and the quality are measured and followed in the task model and thus only schedule and quality deviations can be detected.

> "I'd say that the deviations in this model, one could imagine that quality deviations and schedule, completion and such deviations. "(Manager)

6.3.2 Schedule Deviation

Lack of resources seemed to be a major cause of schedule deviations in company B. The tasks may start to gather in a queue and that causes more schedule deviations.

> "It actually has something to do with queues and the general work load. There are some deviations typically related to certain times. If one starts to delay it probably causes some others to delay too. It could be something sudden, the resources are torn somewhere else or something like that. "(Manager)

Schedule deviations could also be caused by *other duties of employees*.

> "... suprisingly there are not much of them, let's say external issues from task model that would mess the model. There are some, for example the duties of main programmer or software administrators duties more likely. "(Manager)

Inexperience of the developer could also cause the schedule to extend.

"It varies actually, now the latest have been two, three weeks even. It is a long time. The estimated programming time was 10 work days but there were some issues caused by my own inexperience of the area, so it may have overrun. " (Developer)

Schedule deviations were handled in company B with overtime or adding resources, or then the situation was negotiated with the customer.

"The same pattern – can we work overtime, can we add resources, if not, we go back to the customer and discuss about is the extended schedule accepted, can we make prioritising among the customers tasks so that we would get some more resources. "(Upper manager)

The schedules were not tampered with after they were estimated. Original estimations were regarded as learning material in company B.

"And then because the principle in our task model is that we don't reschedule them during the journey, so we try to keep the estimations. Just to learn something. It would be easy to reschedule them and make the accuracy better by it but we haven't gone there. We keep what we have estimated in the beginning. "(Manager)

6.3.3 Quality Deviation

New techniques could affect the quality because there was not yet knowledge about programming and testing with the new technique.

"If new techniques come or we introduce them, there are naturally more of them (defects) than usually. For example introducing technique X was something that caused more quality deviations because the process didn't work yet. We didn't know how to program them and how to test them, planning didn't show as much though. "(Manager)

Testing was not always very well planned but the model has developed and also testing was included in it. Testing is now planned before implementation.

"In today's model testing is along the project from the beginning. Now that we get the planning process integrated, the model will have planning of testing before programming. That's the goal. Until now the weight has been in the end, more like finding bugs in the end." (Manager)

6.3.4 Number of Tasks

The number of tasks has increased significantly as shown in Figure 14 and the trend seems to be almost linear.

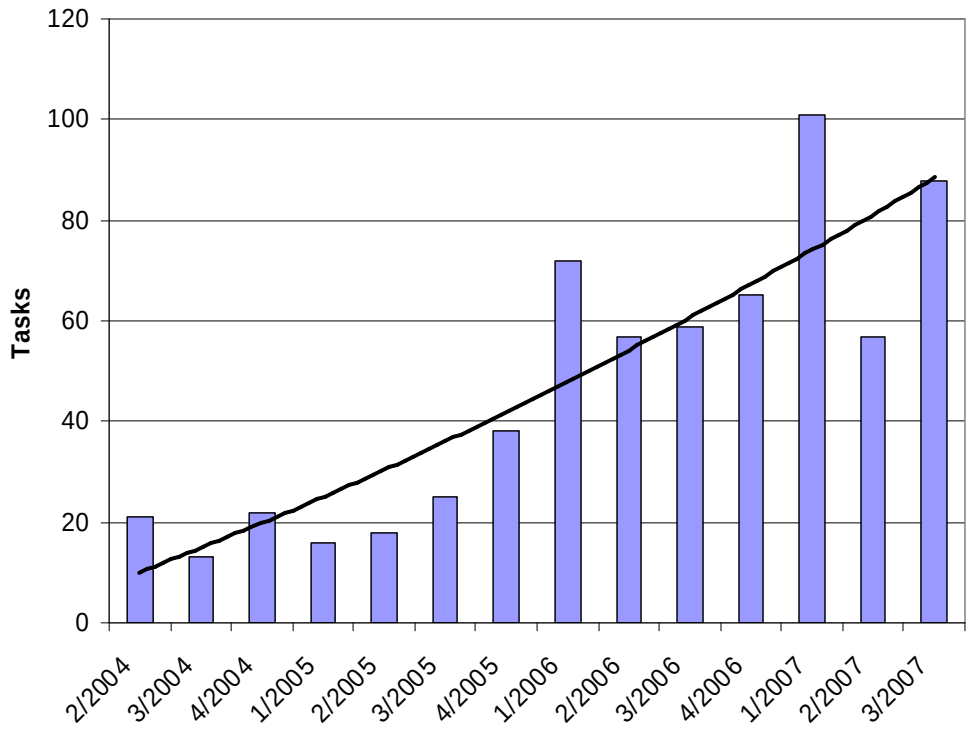

Figure 14. Number of tasks in company B quarterly.

The deviations from the trend in Figure 14 in the first quarters of 2006 and 2007 are caused by the fact that company B has the busiest time of the year in the turn of the year.

"The turn of the year is on average very busy. There are a lot of tasks and their completion is approximately after the turn of the year. A couple of days after the turn of the year, it has an effect. They are actually tasks that are done in the last quarter in the previous year and the are just completed after the turn of the year. " (Manager)

"We try to make things ready to the turn of the year and right after it. And also for practical reasons, if we come to an empty table after the summer then we have defining, planning in the fall, after which they are realized into tasks in the end of the year. After that they usually concentrate on the beginning of the winter that is from January to April / May is the busiest time. Then again towards the summer things start to slow down and they are consciously moved to the fall. " (Upper manager)

The second quarter of 2007 has been a little quieter than should have according to the trend. This is partially caused by introduction of a new planning process.

"We have had actually a little quieter before the summer this year. There was, we took a planning process into use, which is joined to this task model also and it has buffered the tasks a bit, so that less tasks have come through. We seldom have quite time. "(Manager)

6.3.5 Duration of Tasks

In Figure 15 it can be seen that the average duration of the tasks has decreased.

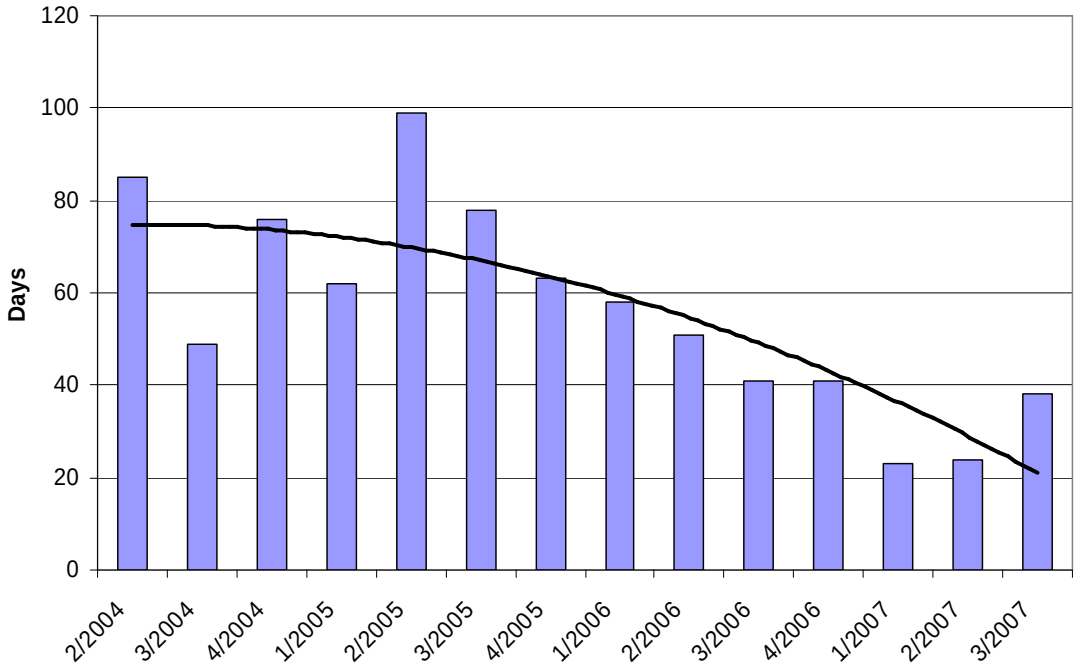

Figure 15. Average duration of tasks in company B quarterly.

An upper manager explains the decreasing trend of the average duration of tasks so that they have increased the speed of software developing.

> "There is a correlation (If the lead-time shortens, more tasks will go through the pipe) – we haven't added resources remarkably so it would be surprising if it were somehow else. Because the only rational explanation is that we have been able to do faster, it also means that we have done more but we haven't added resources, so the lead-time has to have dropped. "(Upper manager)

He sees that the reasons for this have been better planning, better testing, and new, faster techniques.

> "No, there is partially also – a lot of reasons affect this: better planning, better testing, and also technique X shows here. "(Upper manager)

Also a middle manager emphasizes the importance of efficiency in decreasing the lead-time but he mentions also learning to specify the tasks better.

"Well there is, the trend is generally speaking decreasing, so I would suspect that it is because of two reasons; we have learned to specify the tasks better and they have possibly decreased a little on their size, but we are able to do them faster. There is probably bigger effect on the increased efficiency in long run. "(Manager)

The increase of duration in the last quarter is due to the lack of resources.

"The reality has affected the extension of the duration of the task because clearly there are a lot of tasks and too little people. So one small project ties resources plus there is a lack of resources "(Manager)

6.3.6 Size of Schedule Deviations

Also the mean value of deviations has decreased as shown in Figure 16.

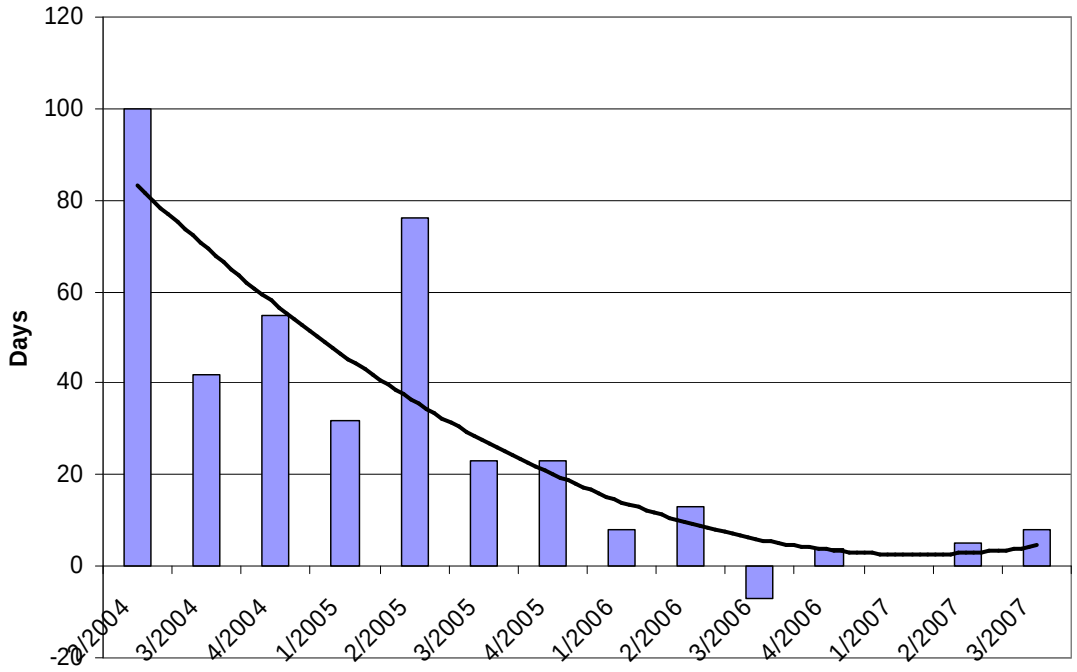

Figure 16. Mean value of deviations in company B quarterly.

In Figure 18 it can be seen that the deviations were remarkable right after the implementation of the task model but since then they have dropped.

"When we changed from the project organization to the task model … we knew the project work in a way, but we didn't know how to estimate and think about how the maintenance and small-scale developing was done. So approximately only 20 % of our schedule promises were correct. Only about 25 % were completed within a week from what we had promised and a half overran by a month –not good. The customers weren't happy. "(Upper manager)

The extensive deviation from the trend in Figure 16 on the second quarter of 2005 is explained on the summer vacations. Due to statistical interpretations the deviation is in the second quarter as it should be in the third quarter according to company B's experts. The data in the graph was collected according to the estimated finishing time and as the experts interpreted the graph they were thinking about the actual finishing time. The tasks were estimated to be finished before the summer vacations in 2005 but they weren't completed by then. That caused the tasks to overrun their schedule and complete after vacations, which makes dozens of days of delay and explains the huge difference.

"I remember that the in 2005 we scheduled the tasks observing the summer, but it shouldn't be seen until… so we didn't understand to observe the summer in 2005 but in 2006 it was fixed, so 2005 it should be seen more in the third quarter that the tasks were running late, but it is seen in the second. "(Manager)

This caused the next year's summer's tasks to be over estimated hence the negative value in the third quarter of 2006.

"So this 3/2006 is because of the fix, so we kind of overreacted it a little. We scheduled there before the summer the tasks to be ending after summer. And in practice we got most of them ready before summer. That caused it, we were calculating extra time just in case keeping the previous summer in mind. " (Manager)

6.3.7 Schedule Overrun

The trend in the percentage value of the overrun tasks is decreasing according to Figure 17. There is quite a big deviation in the third quarter in 2006 and then again in the first quarter of 2007.

Figure 17. The percentage value of tasks with schedule overrun.

The exceptionally good value in the third quarter in 2006 is caused by the pessimistic schedule estimation before the summer.

> "It is explained directly by the summer. It shows the too pessimistic estimation for the ending. It makes the gap. We kind of took into account the summer, although we wouldn't have had to in most cases, we got them ready before summer." (Manager)

The same cause is behind the exceptionally high value in the third quarter in 2007. The schedules were attempted to be made more realistic this time.

"We tried to fix the horribleness from the previous which makes as much to the other direction there in 2/2005 and 3/2005. Then we are on the other side, there is a little fix there. "(Manager)

6.3.8 Relation of Task Size and Deviation

In Figure 18 the average size of a task is shown in the relation of average deviation. As was shown in Figure 16 in the early phases of the task model adoption the schedule deviations were larger and the tasks were bigger than they are now. The big deviations and at the same time the big tasks in Figure 18 occurred in the early phases of the task model adoption.

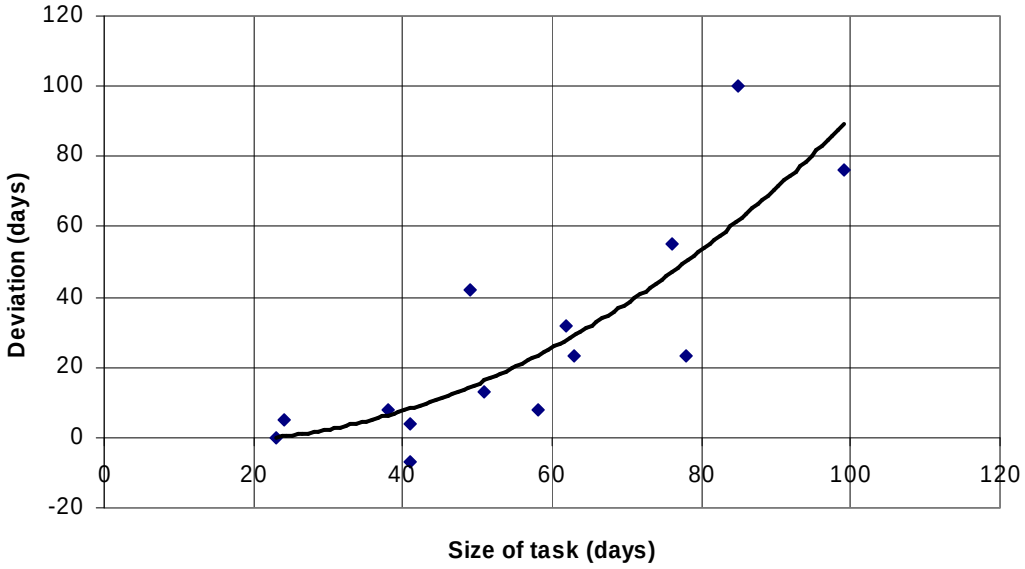

Figure 18. Average size of a task and average deviation according to quarters.

The optimal size of a task seems to be according to Figure 18 from 20 to 40 calendar days. This is also the aim of company B.

"We have thought it ourselves that from one to two weeks of programming would be an optimal task size. So the total effort is around two weeks, programming and

> testing. That means in practice about one calendar month effort, no I mean calendar month duration approximately. Plus / minus the possible delays caused by the lack of resources for example between the phases. You could say from the picture that it is a quite good guess. The bigger the task is, the more it involves uncertainties. ”(Upper manager)

6.3.9 Conclusion of Company B

Company B’s deviations were concentrated on the schedule and the quality. They have had also effort deviations in the beginning of introducing the task model but as the task size has reduced have the deviations with effort reduced also. As the size of the tasks has reduced, it has become easier to estimate the effort and thus the schedule deviations have also reduced.

Schedule deviations in company B were caused by the lack of resources and other duties of employees, inexperience (with some technique) of the developers, and the schedule estimation, especially when it comes to estimating the schedules close to the summer vacations. Quality deviations were caused by new techniques and the low level of planning of testing.

There has been deviations in company B’s tasks but the deviations have been clearly reduced. In 2004 and 2005 more than half of the tasks were exceeding their schedule but the proportion of overran tasks has been decreasing quite steadily since. The trend of schedule overruns is decreasing, although there seems to be some variation as the schedule estimation is still adjusting. Company B has also managed to decrease the size of deviation. First when the task model was introduced, the schedule deviations were huge. At the moment, the deviation varies between 0 and 10 days which is a clear improvement. The exceptions that can be seen in Figures 14-17 are caused by external factors such as the summer vacations or the introduction of a new technique.

It can be seen in company B’s task data that the schedules have been taken under specific examination and it has helped to identify the black-spots of the tasks concerning schedule estimation. Company B has identified the problematic parts and is working continuously to correct them. It seems to be difficult to estimate the schedule of tasks around summer but the estimates are improving. Company B has also found out the optimal size of a task by measuring the task duration and comparing it to the schedule deviations.

7 DISCUSSION

Company A's project metrics were schedule, effort, profit, number of errors, and customer satisfaction. Company B measures schedule, customer satisfaction, and reopen rounds.

Company A has developed tracking systems but analyzing the results is not yet in the same level. Although they measure different things in projects, the results could be compared to the original estimations more and the causes of deviations could be thought more. It seems to be thought that it is not so important to have very detailed plans and specifications and thus the projects are implemented too early. Reacting to noticed deviations could also be improved.

The two companies have different attitude towards schedule estimations and laying out timetables. Company A estimates the project schedule again each month according to the progress so far. Company B does not change the schedule after it has been set. This difference could be because of the very different duration of the development process. However company B's method is able to provide valuable measured data for learning.

The two companies that were chosen to the second phase of the research for a closer examination and their software development are not comparable to each other due to their completely different size of projects / tasks. However it can be seen, that measuring and analyzing some problematic area in a development process enables corrective actions. Company B has managed to reduce their schedule deviation noticeably. Even though company B's task model is not comparable to bigger projects, it can be assumed that measuring and observing a problematic part of a project helps to reduce deviations.

Above mentioned issues awakens a question whether reducing other deviations with the same way is possible. It can be thought that when the problem areas

are known and they can be measured, also the reasons for deviations can better be pondered and thus the root causes can be tackled.

Another interesting thing to be thought of is company B's measured data concerning schedules. More informative results could be achieved if some other measurements were added and compared to the schedule data. For example, measurable things could be defects or defect removal efficiency or then customer satisfaction.

8 CONCLUSIONS

In this thesis software project deviations were studied. The purpose of the study was to find out whether there are deviations in software projects at the moment, what deviations are there, and what causes them.

It was found out from literature and case companies that there are still deviations in the projects and the most deviations are related to schedules. Schedule deviations are caused by effort estimation, requirement deviation, changes during the project, reacting to deviation too late, document deviations, customer's actions, lack of right type resources, demands of a customer, and external reasons. The results of schedule deviations are requirement deviations, adding resources, priorization, and they also have effects on the customer satisfaction.

According to literature bad effort estimation, reacting to deviations too late, requirement deviation or unstable requirements, customer's actions, and inexperienced clients are common reasons for a project failure.

It was also found out that it is possible to reduce schedule deviations by measuring and observing the software development process and taking corrective actions according to the causing factor of the deviation. This is supported also by the literature as inadequate measurements are said to be a common reason for software project failure. Also reducing the size of the task that is being developed reduces deviations with schedule because the smaller tasks are easier to manage. The same trend can be seen quite clearly also in the literature. The smaller the project is, the better it will keep its schedule.

REFERENCES

Carleton, A. D., R. E. Park, W. B. Goethert, W. A. Florac, E. K. Bailey and S. L. Pfleeger (1992). Software measurement for dod systems: Recommendations for initial core measures. Pittsburgh, Pennsylvania, Software Engineering Institute, Carnegie Mellon University.

Chen, Y., B. W. Boehm, R. Madachy and R. Valerdi (2004). An empirical study of eservices product uml sizing metrics. 2004 International Symposium on Empirical Software Engineering. ISESE '04.

Crosby, P. B. (1986). Laatu on ilmaista. Helsinki, Laatuteema Oy.

DeMarco, T. (1982). Controlling software projects, Yourdon Press, Prentice Hall.

Denning, P., J and R. Dunham (2003). "The missing customer." Communications of the ACM **46**(3): 19-23.

EY (2003). 2003/361/ey pienten ja keskisuurten yritysten määritelmä.

Ferens, D. V. (1998). The conundrum of software estimation models. Aerospace and Electronics Conference, 1998. NAECON 1998., Dayton, OH.

Gaitros, D. A. (2004). "Common errors in large software development projects." CrossTalk The Journal of Defense Software Engineering (March).

Glaser, B. and A. L. Strauss (1967). The discovery of grounded theory: Strategies for qualitative research. Chicago, Aldine.

Glass, R., L. (1998a). "Controlling software diseases." ACM SIGMIS Database **29**(3): 13-15.

Glass, R., L. (1998b). "Issue management." ACM SIGMIS Database **29**(4): 16-18.

Glass, R. L. (1998c). Software runaways. Upper Saddle River, Prentice-Hall.

Grable, R., J. Jernigan, C. Pogue and D. Divis (1999). "Metrics for small projects: Experiences at the sed." Software, IEEE **16**(2): 21-29.

Grady, R. B. (1992). Practical software metrics for project management and process improvement. Upper Saddle River, New Jersey, Prentice Hall.

Grady, R. B. and D. L. Caswell (1987). Software metrics: Establishing a company-wide program. Upper Saddle River, Prentice Hall, Inc.

Gustafson, D., A., J. Tan, T. and P. Weaver (1993). Software measure specification. Proceedings of the 1st ACM SIGSOFT symposium on Foundations of software engineering. Los Angeles, California, United States, ACM Press.

Haapio, T. (2007). Improving the effort management of the non-construction activities in custom software development projects. Faculty of Business and Information Technology, Department of Computer Science. Kuopio, University of Kuopio**:** 74.

Haggerty, N. (2000). Understanding the link between it project manager skills and project success research in progress. Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research. Chicago, Illinois, United States, ACM Press.

Haikala, I. and J. Märijärvi (2004). Ohjelmistotuontanto. Helsinki, Talentum.

Hirsjärvi, S., P. Remes and P. Sajavaara (1997). Tutki ja kirjoita. Vantaa, Dark Oy.

Hirsjärvi, S., P. Remes and P. Sajavaara (2002). Tutki ja kirjoita. Vantaa, Tummavuoren kirjapaino Oy.

Ho Leung, T. (1999). A framework for management software project development. Proceedings of the 1999 ACM symposium on Applied computing. San Antonio, Texas, United States, ACM Press.

Hällgren, M. and E. Maaninen-Olsson (2005). "Deviations, ambiguity and uncertainty in a project-intensive organization." Project Management Journal **36**(3): 17-26.

Jalote, P. (2002). Software project management in practice, Addison-Wesley.

Jones, C. (1996). Patterns of software systems failure and success. Boston, International Thomson Computer Press.

Jones, C. (2004). "Software project management practices: Failure versus success." CrossTalk The Journal of Defense Software Engineering (October).

Jones, C. (2006). "Social and technical reasons for software project failures." CrossTalk The Journal of Defense Software Engineering **19**(6): 4-9.

Järvinen, P. (1999). On research methods. Tampere, Tampereen Yliopistopaino Oy.

Keil, M., P. Cule, E. , K. Lyytinen and R. Schmidt, C. (1998). "A framework for identifying software project risks." Communications of the ACM **41**(11): 76-83.

Kerzner, H. (2001). Project management: A systems approach to planning, scheduling, and controlling, John Wiley & Sons, Inc.

Krasna, M., I. Rozman and B. Stiglic (1998). "How to improve the quality of software engineering project management." ACM SIGSOFT Software Engineering Notes **23**(3): 120-125.

Krishnan, M. S. (1993). Cost, quality and user satisfaction of software products: An empirical analysis. Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering - Volume 1. Toronto, Ontario, Canada, IBM Press.

Liu, X. F., G. Kane and M. Bambroo (2006). "An intelligent early warning system for software quality improvement and project management." The Journal of Systems and Software **79**: 1552–1564.

Mahaney, R., C. and A. Lederer, L. (1999). Runaway information systems projects and escalating commitment. Proceedings of the 1999 ACM SIGCPR conference on Computer personnel research. New Orleans, Louisiana, United States, ACM Press.

Masticola, S. P. (2007). A simple estimate of the cost of software project failures and the breakeven effectiveness of project risk management. Economics of Software and Computation, 2007. First International Workshop on the Economics of Software and Computation (ESC'07), Minneapolis, MN, USA

May, L. J. (1998). "Major causes of software project failures." CrossTalk The Journal of Defense Software Engineering (July): 9-12.

Mockus, A., P. Zhang and P. L. Li (2005). Predictors of customer perceived software quality. Proceedings of the 27th international conference on Software engineering. St. Louis, MO, USA, ACM.

Murch, R. (2001). It-projektinhallinta. Helsinki, Edita Publishing Oy.

Nagappan, N., T. Ball and A. Zeller (2006). Mining metrics to predict component failures. Proceeding of the 28th international conference on Software engineering. Shanghai, China, ACM Press.

Reel, J., S (1999). "Critical success factors in software projects." Software, IEEE **16**(3): 6.

RIGHT. (2007). "Right project web site." Retrieved 15.11.2007, 2007, from http://www.it.lut.fi/project/RIGHT/.

Schröter, A., T. Zimmermann and A. Zeller (2006). Predicting component failures at design time. Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering. Rio de Janeiro, Brazil, ACM Press.

Slaughter, S., A. , D. Harter, E. and M. Krishnan, S. (1998). "Evaluating the cost of software quality." Communications of the ACM **41**(8): 67-73.

Strauss, A. and J. Corbin (1990). Basics of qualitative research: Grounded theory procedures and techniques. Newbury Park, CA, SAGE Publications.

Sumner, M. (2000). Risk factors in enterprise wide information management systems projects. Proceedings of the 2000 ACM SIGCPR conference on Computer personnel research. Chicago, Illinois, United States, ACM Press.

Tesch, R. (1990). Qualitative research: Analysis types and software tools, The Falmer Press.

Wallace, L. and M. Keil (2004). "Software project risks and their effect on outcomes." Communications of the ACM **47**(1): 68-73.

Wallace, L., M. Keil and A. Rai (2004). "Understanding software project risk: A cluster analysis." Information & Management **42**(1): 115-125.

Wysocki, R. K., R. J. Beck and D. B. Crane (1995). Effective project management, John Wiley & Sons, Inc.

APPENDIX 1. Themes of the interviews

1 What is a project deviation?

1.1 How a project deviation is defined?

1.2 What kind of deviations are there? (Schedule, budget etc.)

1.3 How a deviation is discovered?

1.4 Can there be positive deviations?

2 Have there been deviations in the projects?

2.1 What kind of projects has had deviations?

2.2 What effects can deviations have?

2.3 Are the consequences small or large?

2.4 Do the same deviations recur in every project?

3 What causes deviations?

3.1 Can one deviation cause more deviations?

3.2 What causes different types of deviations? For example in

- Schedule
- Budget
- Quality
- Documents (For example there are no document, the documents are not updated)
- Requirements

4 Follow-up / Are the projects measured?

4.1 Are the changes documented?

4.2 What is measured?

- Follow-up of working hours
- Schedule (Milestones – Actual and estimated)
- Defects
- Size

4.3 Are the results being analyzed? How?