TDT4735 Software Engineering, Depth Study

Code Reuse in Object Oriented Software Development

Lisa Wold Eriksen

Supervisor: Tor Stålhane
Coordinator: Alf Inge Wang

Norwegian University of Science and Technology, NTNU
Department of Computer and Information Science, IDI
Fall 2004

Abstract

Code reuse in object oriented software development has been common for some time. This report aims to create an understanding of the nature of code reuse in a group of software developing companies in Norway. The research goals were to get an overview of how common code reuse is, to find out what the desired and achieved effects of code reuse are, and to learn about the used tools and procedures which are specifically developed for code reuse. To reach these goals, a series of interviews were conducted. The interviewees were software developers from 24 software developing companies of different sizes, locations, and application domains in Norway. Code was reused by software developers in all the companies, but there were few companies where tools and/or procedures specifically developed for code reuse were employed. The majority of the companies performed reuse in a disorganized manner, without separating reusable code from other code. The interviewees reported that the achieved effects were the same as the desired effects. The effect most commonly mentioned, was improved development efficiency, i.e. saved time and money. The effect which was second most often referred to, was improved quality of software. Other effects named were improved stability of software, simplified testing of software, uniformity of how problems are solved, more accurate time and price estimates, and marketing advantages. It seems that many software developers are aware of the positive effects of code reuse, but lack consciousness as to how these effects best can be achieved.

Preface

This document was written as a part of the graduate level course "TDT4735 Software Engineering, Depth Study" during the fall semester 2004. The author is a fifth year student at the Norwegian University of Science and Technology (NTNU), Faculty of Information Technology, Mathematics and Electrical Engineering (IME), Department of Computer and Information Science (IDI), where this course is taught.

My sincere thanks to Tor Stålhane, for help and guidance throughout the project and for teaching me "Experimental Software Engineering". I would also like to thank coordinator Alf Inge Wang for teaching me "Software Technology Evaluation" and doing a great job at providing information — especially on the web — both as a coordinator and a teacher. Two more persons to make an impact on my work were Peter Rønning (always available) and Mona Elisabeth Østvang (thanks for "making my day").

Last, but not least, I wish to thank all the companies and interviewees who participated in this project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> Proper reuse of code increases the speed of software development projects.
>
> - Rickard Öberg[1]

1.1 Motivation

Since I first started programming four years ago, I have produced a lot of code. Different projects and assignments meant writing different code, but more and more often I found myself thinking "Oh, I've done this before!" This was followed by a intense search through directories and files to find where exactly I had this piece of code. More often than not I came up with nothing and had to program the same code over again. Every time, I thought to myself "If only I had a personal code library". In addition to my own code I have also had access to others' code in group projects and assignments. It would be nice to have this code in a library as well. But the time and efforts needed to create my own library of re-usable code seemed to be greater than I could afford. This seemed to be the case for many of my classmates as well.

While I was in the process of deciding on a subject for my fifth year project, I went to my supervisor to get some advice on what to choose. We discussed several topics, amongst others code reuse. This again started me thinking about my own lack of reusable code and also made me wonder about how common code reuse really is. When I start working as a programmer, one of the first things I will probably do is create a personal system for organizing code and code snippets that I think I will be using often. In relation to

[1] Quoted with permission

this I wondered: How common is code reuse in the software industry today? What kind of code reuse systems exist and are actually in use? What are the expected and experienced effects of reuse? I searched the web for information, and found several statements claiming that code reuse most certainly is the way to go and that reuse saves time and money in software development. For more information on this, see the Prestudy, section 2.1. The subject of code reuse indeed seemed intriguing and I decided that code reuse would be the subject of my project.

1.2 Problem Definition

When I had decided that I wanted to have code reuse as the main subject for my project, I had to concretize what exactly it was I wanted to do. First and foremost I decided that I would only consider object oriented programming languages, as these are the kind of languages I know best. After discussing the topic with my supervisor, searching on the web and coming up with ideas I had some basic notion of what I wanted to do. First I made a thought diagram, displayed in figure 1.1. This diagram contained several topics and sub-topics which I felt would be interesting to work on. I then tried to group some of these topics together to form suitable project subjects. I also considered some possible subjects for my Master's thesis based on the potential subjects for my project. Some examples are included in table 1.1. I decided that I wanted to do some research on what kind of tools and

Project	Gather information about and test tools
Thesis	Develop tool(s)
Thesis	Gather experiences, look at existing procedures or develop new procedures
Project	Look at tools and procedures
Thesis	Develop tool(s)
Project	Learn about tools and procedures
Thesis	Develop procedure(s)

Table 1.1: Project Ideas

possibly procedures specialized for reuse exist, and perhaps test some of them. Depending on the results from the project, I could try to develop a tool myself or do more thorough research on the tools and methods I studied in the project for my Master's thesis. With some help from my supervisor and some more thinking I decided to use the project title "Reuse of code in object oriented software development" and the following project description: *"Proper reuse of code increases the speed of software development projects." (Rickard Öberg) In this respect, tools and procedures for code reuse and*

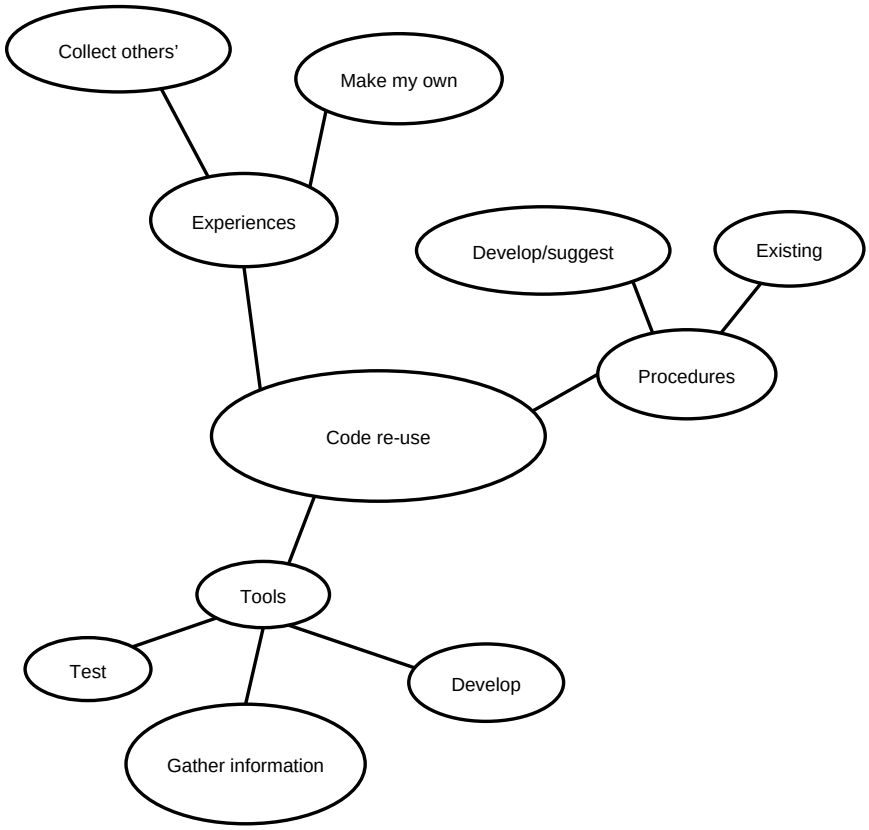

Figure 1.1: Thought Diagram for Reuse Topics

experiences with reuse are important. The project is about examining and assessing the present solutions for reuse of object oriented code.

Reuse is, however, a loose term. I had already decided that I wanted to look at the lower level of reuse; the reuse of source code. This in contrast to higher level reuse such as patterns or process reuse/experience databases. There are still many definitions of and ways to interpret "code reuse". I wanted the overview of how the software industry reuses code to be as general as possible. Thus, I decided not to exclude any interpretations of reuse. A wide definition which describes the essence of code reuse is: Code is reused when it 1) already exists, and 2) is chosen over the possibility to write new code.

1.3 Goals

The project description was pretty general. To be a bit more specific, I made a list of things I wanted to achieve, i.e. goals for my project work. The main goal was to learn about tools and methods for code reuse and to find out how common code reuse is. It was important for me to learn about the tools and procedures which are actually used in the software industry today. I deconstructed this into the following sub-goals:

- Get an overview of how common code reuse actually is
- Get to know why people choose to reuse code and what they get out of it
- Learn about the specialized tools software developers are using for code reuse
  - What kind of tools and how they are used
  - How the developers feel about using the tools
  - How useful the tools are
- Learn about the specialized procedures used in connection with reuse
  - What kind of procedures are used and how they are used
  - How the developers feel about using these procedures
  - How useful the procedures actually are

1.4 Report Outline

The rest of this report is organized as follows:

Prestudy and Approach The prestudy provides information on various aspects, or effects, of code reuse; such as saved time, improving quality, and marketing advantages. A brief explanation of how I wanted to approach the subject of code reuse is also included in this chapter.

Research Method Theory In this chapter, general theory on research processes is presented together with theory on a specific type of research; surveys.

Research Planning This chapter describes the planning of the research process, based on the theory in the preceding chapter. The plans include definition of objective, definition of target population, sample selection, survey methods, interview guide, and plans for data analysis.

Research Implementation This chapter describes the implementation of the plans from the previous chapter.

Results This chapter contains the research results and a discussion of the results.

Evaluation This chapter contains an evaluation of the research process and the process of extracting information from the interview notes.

Conclusion and Further Work This chapter contains a conclusion, and some suggestions for further work.

Chapter 2

Prestudy and Approach

In this chapter I will describe some of the aspects of code reuse commonly brought up in the resources I found on the Web. I will also explain in brief how I wanted to approach the subject of code reuse in order to reach the project work goals described in section 1.3.

2.1 Prestudy

First a distinction: There is a difference between code reuse and *proper* code reuse. Just as proper reuse can have a positive effect of the development, poor reuse can have a negative effect. This is important to have in mind, and it applies to every aspect of code reuse. Through the rest of this chapter when I mention the positive effects of reuse I am referring to proper reuse.

The aspect of code reuse which is generally the most emphasized is time, and thereby costs. Rickard Öberg states that "Proper reuse of code increases the speed of software development projects". The fact that reuse saves development time is mentioned on lots of Web sites, both software developing companies' Web sites and educational Web sites. Also reuse and object oriented programming are linked through statements naming the possibility of reuse as an advantage of using object oriented programming languages. Time is saved by reusing code for several reasons. One of the most obvious ways reuse saves time is by reducing the time spent writing code. Time can also be saved when it comes to working out how to translate an idea into code, as this might already have been done in already produced, reusable code. Time can also be saved on testing if the reusable code has already been tested thoroughly or if a specialized test system (e.g. a class or a module) is provided alongside it. When a piece of reusable code which is specific to one matter (e.g. Web certificates) is produced, it can save time by allowing

developers who don't have deep skills in this matter to save some of the time it takes to learn all there is to know about the matter.

One clear effect of saving time is saving money. Another, perhaps not so obvious, effect is goodwill. When the company is able to deliver the product in a shorter amount of time, the customers will probably be more satisfied. Another effect of reuse, particularly modular reuse, is that as more reusable modules are produced, the developers might not need to do much more than putting modules together to produce a new software product. This also helps provide a relatively accurate price estimate.

Another important aspect is risk and quality. The fact that the program code already has been used for a while can provide valuable information as to the quality of the program code. This can be stability, performance, scalability, and security. Sometimes the code is "tweaked" in time, remaining bugs are found and corrected, possibly producing better code. As previously mentioned, however, one has to take care when reusing code. One example of poor reuse is the catastrophic error which occurred during the launch of the first Ariane 5 launcher in June 1996[1]. In this case, software used in Ariane 4 had been reused for Ariane 5 and it was decided that it would not be wise to make changes in software which worked well on Ariane 4. Differences in technical constraints from Ariane 4 to Ariane 5 caused the software to malfunction during launch, resulting in the crash of Ariane 5. This goes to show that code which works well in one context doesn't necessarily work well in another context.

One way do avoid some of the problems connected to reuse is to produce easily understandable, self-documenting code and generating documentation. Good documentation of for example functionality, context and explanation of programming choices made (e.g. one algorithm over another) will help the developers wishing to reuse the code. This documentation will enable them to assess whether the code is appropriate for their use, and if they wish to change something, the documentation will help them understand how best to do it.

2.2 Approach

As previously stated, the main goal of this project was to discover how common reuse is and learn about tools and methods for code reuse. I wanted to learn about the tools and techniques which were actually in use. Learning about all the relevant aspects of all available tools and procedures would

[1] See http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf for the report made by the inquiry board after the accident.

take too much time if I in addition was going to find out which of them were actually used, by whom, how and what the effects of using them were. I could not find any recent studies about this, so I decided to conduct my own study of the "state of the art". I figured the easiest way to find out about the currently used tools and procedures was to ask the people who might use them. This would also make a good basis for my Master's thesis. The next chapter will contain some theory regarding how such studies should be performed.

Chapter 3

Research Method Theory

In this chapter I will present some general theory on research processes (section 3.1). As I wanted to communicate with software developers in order to reach my project work goals (section 1.3), I have chosen to focus on theory describing a type of research which includes oral communication: Surveys. Included in this chapter is general information on surveys (section 3.2) and the key steps in planning a survey (section 3.3).

In planning and conducting this project I have found several sources of information helpful. Fledsberg ([1]) describes the general theory on research methods, mainly inspired by Ringdal ([2]). Galobardes ([3]), Scheuren ([4]) and Boone ([5]) focus on surveys, which is also described by Wohlin ([6]).

3.1 Research processes

[2] states that the Scientific method is procedures or techniques to answer research questions. This includes, amongst other things, techniques to collect data and analyzing these. The author goes on to remark that without method knowledge, assessments of choice of research arrangements or results based on statistical techniques become shallow. [2] presents the research process as consisting of the steps presented in figure 3.1. This is, however, a simplified picture which hides the fact that more often than not, one has to turn and go back one or more steps.

3.1.1 Rough Problem Definition and Research Questions

The first step, Rough problem definition, consists of an idea which stems from the researcher's own interests or/and from users or customers. This

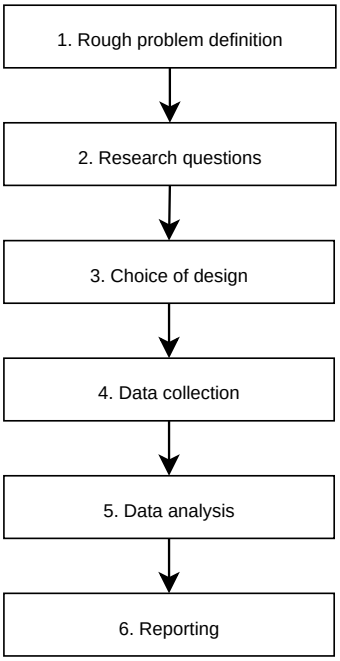

Figure 3.1: Main Steps of the Research Process

idea has to be transformed into research questions (step 2). This transformation might be based on previous research and theory, and the research questions may take the form of questions or hypotheses. Questions are open (e.g. "Is there any relationship between the number of storks and the number of childbirths in the country?") while hypotheses are claims about reality (e.g. "There is a positive linear relationship between the number of storks and the number of childbirths in the country.").

3.1.2 Choice of design

Step number three, Choice of design, constitutes making a rough sketch of how a specific investigation should be formulated. A design is based on several choices, for example whether the investigation should be qualitative or quantitative. Qualitative methods result in text data while quantitative methods result in quantifiable data. It is also possible to construct a design which combines qualitative and quantitative methods. The two most common qualitative designs are field observations and conversational interviews. Surveys and experiments are the most common quantitative research designs. General information on surveys is included in section 3.2, while a description of how to plan a survey is included in section 3.3.

3.1.3 Data Collection

Data collection is the next step (step 4). The first choice to be made is whether data collection is necessary. Sometimes data has already been collected, removing or reducing the need to collect data. There are many ways to collect data, some of which are represented in table 3.1.

Technique		*Used in*
Questionnaire		Survey design
Interview	Telephone interview	Survey design and quantitative investigations
	Visitational interview	
Observation	Field observation	Quantitative investigations
	Laboratory observation	

Table 3.1: Data Collection Techniques

Before data is collected, however, it is important to make a selection of subjects. There are two kinds of reasoning behind the choice of subjects. If it is desirable to have a selection which is representative of the population, the selection has to be performed using statistical criteria. If, however, only a few cases are to be selected, it is better to make the selection based on strategic reasons.

3.1.4 Data Analysis

After the data collection, the data should be analyzed (step 5). Before the data can be analyzed, however, it has to be registered, preferably electronically. The way in which the data is registered depends on the purpose of the data collection. Registering data from surveys might be done simultaneously with the data collection during interviews where the questions are both read from and registered directly to a computer. For qualitative research, the data matrix is not quite as useful. Often, recording tapes or video is used during the interviews, or the interviewer takes notes during the interview. In any case, the data from the interviews are stored electronically.

3.1.5 Reporting

The final step of the research process is reporting (step 6). The results may be reported in a journal, report, Master's thesis, dissertation, or book. The most important way of reporting research results is presently journals. There is an important divide between professional publications and popularizations; the latter gives a wider audience than the former, whose audience mainly is other scientists.

3.2 General Information on Surveys

The information in this section is taken from [6].

Surveys are conducted when the use of a technique or tool already has taken place, or before it is introduced. It could be seen as a snapshot of the situation, used to capture the current status. Surveys could, for example, be used for opinion polls and market research.

When performing survey research the interest may be, for example, to study how a new development process has improved the developer's attitudes toward quality assurance. A sample of developers is selected from all the developers at the company. A questionnaire is constructed to obtain information needed for the research. The questionnaires are answered by the sample of developers. The information collected is then arranged into a form that can be handled in a quantitative or qualitative manner.

Survey characteristics

The purpose of sample surveys is usually to understand the population from which the sample was drawn. For example, by interviewing 25 developers on what they think about a new process, the opinion of the larger population of 100 developers in the company can be estimated.

Surveys have the ability to provide a large number of variables to evaluate, but it is necessary to aim at obtaining the largest amount of understanding from the fewest number of variables since this reduction also eases the analysis work.

Survey purposes

The general objectives for conducting a survey is one of the following:

- Descriptive
- Explanatory
- Explorative

Descriptive surveys can be conducted to enable assertions about some population. This could for instance be determining the distribution of certain characteristics or attributes. The concern is not about why the observed distribution exists, but what that distribution is.

Explanatory surveys aim at making explanatory claims about the population. For example, when studying how developers use a certain inspection

technique, we might want to explain why some developers prefer one technique while others prefer another. By examining the relationships between candidate techniques and several explanatory variables, we may try to explain why developers choose one of the techniques.

Finally, *explorative* surveys are used as a pre-study to a more thorough investigation to assure that important issues are not forgotten. This could be done by creating a loosely structured questionnaire and letting a sample from the population answer it. The information is gathered and analyzed, and the results are used to improve the later, full investigation. The explorative survey does not answer the basic research question, but it may provide new possibilities that should be analyzed and should therefore be followed up in the more focused or thorough survey.

Data collection

The two most common means for data collection are questionnaires and interviews. Questionnaires could both be provided on paper or in some electronic form, e.g. e-mail or Web pages.

Letting interviewers handle the questionnaires by telephone or face-to-face instead of letting the respondents fill them in themselves, offers a number of advantages:

- Interview surveys typically achieve higher response rates than, for example, mail surveys.
- An interviewer generally decreases the number of "do not know" and "no answer", because the interviewer can answer questions about the questionnaire.
- It is possible for the interviewer to observe the interviewee and ask questions.

3.3 Key Steps in Planning a Survey

[3] presents seven steps to be performed when planning a survey. These key steps contain much of the same as the research process described in [2]. The seven key steps are:

- Definition of objective: Clearly specify the aims and purpose of the survey and decide the study design.
- Definition of target population: State the target population of the survey in terms of place, time and other relevant criteria and determining

who will be excluded.

- Sample: Decide on the sample selection and sample size.
- Survey methods and quality assurance: Activities directed at ensuring data quality before data collection begins; validity and reliability are important keywords. Design the questionnaire and test it, develop the manual of operation.
- Implementation of survey and quality control procedures: Monitor and maintain the quality of the data while conducting the study.
- Statistical analysis: Analyze the collected data.
- Reporting and dissemination of results

Chapter 4

Research Planning

I initially chose to use surveys as my main source of information. One of the reasons for this was the description of surveys I found in [6]:

> Surveys are conducted when the use of a technique or tool already has taken place [...] It could be seen as a snapshot of the situation to capture the current status.

This seemed to be exactly what I wanted. There are many ways to conduct a survey and I felt that I would be able to plan and conduct one which would fit my constraints of time and money and still provide the information I wanted. The detailed aspects of research focus, research questions and research process are described later in this chapter.

I found the key steps in planning a survey presented in [3], described in section 3.3, to be quite appropriate for my study, with some adjustments. Hence I started following the "key steps recipe", described in the following section. In order to make the connection visible, I several places in this chapter refer to the research process steps introduced in the previous chapter (section 3.1).

4.1 Following the Key Steps

As [3] states,

> [...] choices are made at each step of the survey. These depend on scientific, practical, and financial constraints.

The choices made during the survey have to be justified and there has to be records of the decisions in the protocol of the study, the document describing

the study and all procedures used in the survey. The "protocol of the study" is in this case the present chapter.

The main steps of planning a survey are Definition of objective (covered in section 4.1.1), Definition of target population (section 4.1.2), Sample selection (section 4.1.3) and Survey methods and quality assurance (section 4.1.4). This section will explain how I planned the survey by describing each of these steps in turn.

4.1.1 Definition of Objective

In order to conduct a survey, the aims and purposes of the survey had to be specified. I also had to decide on the study design. This is similar to step 2 and 3 in [2] (see section 3.1.1 and 3.1.2). The following two sections might also be considered as parts of step 3.

The purpose of this survey was to investigate the current solutions for reuse of object oriented code in the Norwegian software industry. The investigation concerned various aspects of the solutions in use, as described in the section Goals (1.3):

- Get an overview of how common code reuse actually is. (Does "everybody" reuse code?)
- Get to know why people choose to reuse code and what they get out of it. (What are the desired and achieved effects?)
- Learn about the tools which are specifically developed for code reuse, and which software developers are using in practice:
  - What kind of tools and how they are used
  - How the developers feel about using the tools
  - How useful the tools are
- Learn about the procedures which are specifically developed for code reuse, and are used in practice:
  - What kind of procedures are used and how they are used
  - How the developers feel about using these procedures
  - How useful the procedures actually are

I did not aim at making generalizations from the results of the survey; the survey was carried out primarily in order to get an overview of the situation concerning code reuse in a sample of companies in Norway at the present time. The results of the survey will be used as a basis for my Master's thesis

during spring 2005. This meant that the study is an explorative survey. (The different types of surveys are described in section 3.2.) Due to the nature of the project, the study took place during a three month period in the fall 2004.

4.1.2 Definition of Target Population

As described earlier, the survey should answer questions about object oriented code reuse in the Norwegian software industry. The target population of this survey was therefore the current software developers and software development departments in companies in Norway which produced software using (and reusing) object oriented code. The sample would contain only companies producing software for sale, not companies producing software for internal use only. The reason for this limitation was that the software development process when the software is to be used internally differs from what it is when the software is produced to be sold to external customers. I wished to concentrate on code reuse in the process leading to a software product for sale.

4.1.3 Sample Selection

As a survey of the complete population was not feasible in this project, I had to use only a sample from the population. If I wished to be able to generalize the results of the survey, I would have needed to ensure that the sample was representative of the target population. As I simply wished to get an overview of the situation to get a basis for my Master's thesis, I didn't have to be quite as strict when selecting the sample. There were still several aspects to consider;

- size of the company (measured by for example the number of employees, annual profit, or market share)
- company type (e.g. software product domains)
- maturity of the company (e.g. number of years the company has existed)
- presence of a software development department
- size of the development department, if one exists (e.g. measured by number of employees)
- maturity of the development department, if one exists (number of years the development has existed, experiences in different domains)
- programming languages used (object oriented languages)

Before I selected a sample of the population, I had to contact several companies. The number of companies I initially contacted had to be substantially larger than the number of companies I wanted to have in the final sample. I decided to initially select participants quite randomly, based on the category "Computer software and development"[1] in "Gule Sider®"[2]. This selection of participants would hopefully result in a wide variety of companies, giving me a sample containing companies which differ with respect to the above mentioned aspects.

4.1.4 Survey Methods and Quality Assurance

This subsection concerns the survey methods and the activities directed to ensure data quality before data collection begins. The activities might be considered as parts of step 4 in [2] (see section 3.1.3).

I planned to create a questionnaire which would be administered over the telephone. I chose using the telephone for the following reasons:

- It is an effective method for collecting the information. This is important, as the time period in which I had to conduct the survey is short — less than three months.
- The "presence" of an interviewer can increase the cooperation rate and make it possible for respondents to get immediate clarifications.[4] This means that the number of "do not know" and "no answer" will decrease, hence the quality of the survey will increase.
- Being able to talk to the respondents means that I could ask questions and get a better understanding of the respondents' answers.

One way to influence the respondents' understanding and participating in the survey, and thereby the quality of the survey, is to have a well constructed and designed questionnaire. As the survey would be conducted via the telephone, the appearance of the questionnaire was not important, but things such as wording and logical structure were. Before the actual survey, I planned to test the questionnaire and my interviewing skills via the telephone. My subjects would be a few fellow students who had the right level of knowledge of programming and tools to assess wording and logical structure.

At this stage in the survey planning, the manual of operation would be created. This is a document which contains a detailed description of the procedures to be used in the survey. However, I started calling software producing companies to get participants for the survey before this work was

[1] In Norwegian: "Dataprogramvare og -utvikling"
[2] http://www.gulesider.no

finished. In this process I realized that a survey with a strict questionnaire would not be the best way to achieve the project goals (see section 1.3). The reason for this change of heart is explained in the chapter Evaluation, section 7.1.3. In short, one of the developers I spoke to practically functioned as a test person for the preliminary questionnaire, and the result was that I decided that an interview guide would be more appropriate for my purposes.

4.2 Interview Guide

I planned to use my unfinished questionnaire design (included in appendix B) as the basis for an interview guide. [2] states that interview guides vary from keywords ordered by subject to a set of formulated open questions. I chose to use keywords, as this would enable a more open conversation than previously formulated questions. During the interviews, I planned to write notes using my laptop computer.

4.3 Plans for Data Analysis

Step 5 in [2] is Data analysis (see section 3.1.4). [1] and Kvale ([7]) present a method for analysis which consists of categorizing the contents of the interviews. Long statements are reduced to simple categories, and are used to indicate the existence (or the opposite) of a phenomenon (e.g. “no tools”, “procedures used”), and possibly the strength of the phenomenon (e.g. on a scale from 1 to 5). This method will structure and reduce a longer text into a few tables and/or figures. I felt this would be a good way to extract the essences from the interviews, and decided to use this method for analyzing the collected data. To ensure the correctness of the collected data, I planned to write interview transcriptions and send them to each interviewee for verification and possibly correction of misunderstandings etc.

This concludes the planning. The implementation of the plans are described in the following chapter.

Chapter 5

Research Implementation

In this chapter, I will describe the implementation of the research plans described in the previous chapter. Modeling the questionnaire is described in section 5.1, while sample selection is covered in section 5.2, and making the interview guide is described in section 5.3. Performing the interviews is the topic of section 5.4, and the process of writing interview transcriptions and getting them verified by the interviewees is covered in section 5.5.

5.1 Modeling the Questionnaire

To prepare for the survey, I started making a questionnaire. I used the list in section Definition of Objective (4.1.1) as a starting point, and developed the unfinished questionnaire included in appendix B. As I was modeling the questionnaire, I was also trying to find participants for the survey. After a while, I did no longer wish to use a questionnaire, and the unfinished questionnaire was never completed. For an explanation of this, see the following section.

5.2 Sample Selection

According to the plan, I used Gule Sider®to find participants for what was then planned to be a survey. I started out with 53 possible subjects, selected randomly from the list generated by selecting the "Computer software and development" category, except for one criterion: I had to be able to determine from the information in the generated list and the companies' Web pages that the companies actually were producing and selling software. I contacted the companies by telephone and explained why I contacted them.

In most cases the first person I spoke to was a switchboard operator, who referred me to a software developer/person in the software development department. I explained why I called, and if the person seemed interested in participating, I offered to send an explanatory e-mail (included in appendix A).

About half of the companies I contacted didn't wish to participate for different reasons, generally too little time was the reason given. After contacting all the companies on my list of 53, I had appointments with 25 participants. One of the participants did, however, not answer my call at the scheduled time or at any later time. Nor did he answer e-mails, and so he did not participate in the interview process. This left me with a final list of 24 interviewees.

During the period in which I made appointments, one of the people I spoke to explained to me that he didn't have the time for an appointment later, but he had the time to participate at the time I contacted him. This, as explained more thoroughly in the chapter Evaluation (section 7.1.3), caused me to revert from the plan of using a questionnaire. I decided that an interview guide wold be a better help when talking to the participants.

5.3 Making the Interview Guide

As planned, I used the unfinished questionnaire design (see appendix B) as the basis for creating the interview guide. As described in the research plan (section 4.2), I created an interview guide consisting of keywords. This would enable me to ask questions freely, but the guide would still remind me of the information I wanted. I put the keywords in the order in which I planned to discuss them, starting with the "simplest" information (which programming language(s) are used). The last topic, how reuse is done in practice, might have been covered earlier in the interview, and thus might not need to be addressed separately. The interview guide is included in appendix C.

5.4 Performing the Interviews

The interviews were conducted during a four week period in the autumn of 2004. I tried to conduct the very first interview using the unfinished questionnaire, but during the interview, I realized that a questionnaire would be too specific and not have enough room for follow-up questions etc. The rest of the interviews were conducted using the interview guide (see appendix C). At the beginning of each conversation, I informed the interviewee

that I was taking notes as we spoke, and that this might cause some pauses in the conversation, and I might ask them to repeat what they just said. I used my laptop computer for writing the notes. The interviews lasted from 10 minutes to about one hour. A summary of the interviews is included in chapter 6, and the transcriptions of the interviews are included in appendix E together with some more information on the transcriptions and a list of the companies.

At the end of each conversation, I asked the interviewee if I could contact them by e-mail or telephone at a later time if I had any questions, and none of the interviewees had a problem with this. In the e-mails sent to most of the interviewees (see appendix A), I explained that the information I gathered would be anonymized. At the end of the interviews, I asked the interviewees how anonymous the interviewee and the company wished to be. Most of the interviewees told me I could use as much information about them and the company as I wished in the report, but some told me they wanted to see the transcription before I put it in the report. This was actually my next, and last, question to the interviewees; if I could send the interviewee the transcript from the interview for verification. All interviewees agreed to this. The verification would consist of two parts; verifying that the information was correct, and verifying that it was OK for the interviewee and the company that the information was included in the report.

5.5 Interview Transcriptions

As the notes I made during the interviews were highly informal and probably not understandable to anyone but me, I had to "translate" them in some way to make them understandable to others. I decided to translate them into English, as the report is written in English. I also chose not to include everything which was said during the interviews; I excluded comments such as "C++ really is not the way to go, if you want to be successful, you have to use C#", which I felt would not be useful in this context. The original notes were saved, although not included in this report. The interview transcriptions are included in appendix E.

When I had finished translating the interviews, I sent the results (both the transcriptions themselves and the information about the company which accompanies the transcriptions in appendix E) by e-mail to each interviewee for validation. In this e-mail I asked again how anonymous the information from the interview should be. I also asked them to check that the information was correct. As I was running out of time, I had to receive the validations in a relatively short amount of time (for some, one and a half

weeks, for others, four days). I had set a deadline for the latest time I had to receive the verifications, and after this deadline I phoned the ones who had not answered, to get their verification. The final result was that all the transcriptions were verified, just in time to be included in the report. None of the interviewees or companies wanted to be anonymous.

Chapter 6

Results

In this chapter, a summary of the interview transcriptions (section 6.1) and a discussion of the results (section 6.2) are included. The full interview transcriptions are included in appendix E.

6.1 Interview Summary

To enable easier access to the results of the interviews, I have chosen to make a summary of the interviews, in the form of a table. Table 6.4 contains the essential information gathered from each company during the interviews. *Column 1* of the table contains the company names. International companies are marked with a star (*), and consultancy firms are marked with a plus sign (+). Consultancy firms are generally less domain specific than the other companies, which also makes the plus sign a possible indicator of a low level of domain specialization. *Column 2* indicates the number of employees in the company and the number of software developers. The two numbers are separated by a forward slash (/). For international companies, only the number of employees in Norway are listed. *Column 3* contains the extent of reuse, indicated by the abbreviations given in table 6.1. *Column 4* describes which effects of reuse were mentioned during the interview, indicated by the abbreviations given in table 6.2. Some information on effects of code reuse is given in the Prestudy, section 2.1. As all the interviewees stated that the achieved effects were the same as the desired effects (to different extents), I have chosen not to list the desired and achieved effects separately. I have tried to make the effect categories general enough to include similar statements; both the statement "we reuse to save time" and the statement "a desired effect of code reuse is higher efficiency in software development" are included in category E: Increased efficiency. Still, I have tried not to make

Abbreviation	*Extent of reuse*
1	Reuse is left up to each developer
2	There is a development department policy on reuse
3	There is a company policy on reuse
A	There is no common base of reusable code which is separated from other code
B	There is a common base of reusable code which is separated from other code

Table 6.1: Extent of Code Reuse — Codification

the categories too wide. I have for example chosen to list stability and quality as two different effects, although stability might be considered a form of quality. This is because I wished to express the importance of stability over (or in addition to) general quality for some interviewees.

Abbreviation	*Effect*
E	Increased efficiency, i.e. saved time (and thereby money)
Q	Improved quality of software
S	Improved stability of software/fewer errors
T	Simplified testing
U	Uniformity (of how problems are solved, of appearance and functionality)
P	More accurate time and price estimates
M	Marketing advantages (experience and ease of software development)

Table 6.2: Effects of Code Reuse — Codification

The first part of *column 5* indicates whether a tool specifically produced for reuse is employed, denoted by "No" for no tools employed and "Yes" when tools are employed. The second part indicates the use of one or several reuse specific procedures. "No" indicates no presence of specific reuse procedures, "Yes" indicates the presence of formal procedures, while "yes-" indicates the presence of some kind of reuse procedure which is not formalized. The two parts of column 5 are separated by a forward slash (/). *Column 6* contains a classification of reuse; how reuse is organized. This is based on the models of reuse described in appendix D, and it is codified as described in table 6.3. If neither of these models fit the reuse organization of a company, this is denoted by a single dash (-).

Note: Fundator and Lydia are both in a special situation, as they are in the final stages of adopting new systems for software development. This

Abbreviation	*Reuse model*
PO	Project oriented reuse
SP	Separate reuse project
CP	Reuse with component producer
DP	Reuse with domain producers

Table 6.3: Models of Code Reuse — Codification

summary describes the situations in the companies before these systems were adopted.

6.1.1 Worth Noting

Some of the interviewees made statements which are worth noting, but aren't of such a nature that they fit into the interview summary table. I have chosen to include these statements here, to make the reader aware of their existence.

The first quote is taken from section E.13.2, the interview with Tor Haugen at Electric Farm:

> Reusable components can also improve the process of starting a new project. When the members of the development team meet to start a project, they have to get an idea of what is to be done. Sometimes the developers may feel "lost" and it may seem like they have to start from scratch. In this situation, it is useful to have reusable components from a previous project which resembles the new project. This makes things easier for the developers by reducing the need for discussions and clarifications of how functionality is to be implemented et cetera, as some parts already are implemented. The resemblance to the older project and reusing code from that project help make the new project more "recognizable" as well, and in this way, reuse helps create continuity between projects.

The following quotes were taken from section E.21.2, the interview with Thomas Lünell at ErgoEphorma:

> A negative effect of reuse is possible loss of competence. How something works is easily forgotten, and as components are reused over and over, the developers might forget the reasoning behind the component (how does it work, why is a functionality implemented in a certain way, etc.). In this case, documentation is essential. It might also be an advantage to sometimes rewrite code in order to maintain a certain level of knowledge.

Company	*Employees*	*Ext.*	*Effects*	*Tool/ Procedure*	*Org.*
Ajour Media	6/2	3B	EQS	No/yes-	PO
Antares Gruppen+	30/17	1A	EQS	No/No	-
ErgoSolutions+	300/150	3B	ET	No/No	PO
Fundator+	17/14	1A	EQTPM	No/No	-
TietoEnator+*	1100/500	3B	EQS	Yes/Yes	SP
Fronter*	24/6	3A	EQSU	No/No	-
QS Manager	3/2	1A	E	No/No	-
Escio	7/2	3B	E	No/yes-	PO
Cintra Software Engineering+	2/2	3A	EQ	No/No	-
Lydia	11/4	1A	ET	No/No	-
adramatch*	35/3	2A	ET	No/No	-
Electric Farm+	23/19	3B	EQSTUM	No/yes-	PO
MaXware*+	33/10	1A	E	No/No	-
Finale Systemer	14/8	2B	EQ	No/No	PO
Well Diagnostics	10/5	1A	EQT	No/No	-
AKVAsmart*	100/12	2A	EQS	No/No	-
Vega SMB	9/4	1A	EQ	No/No	-
Geodata	80/22	2B	EQU	No/No	PO
Egroup+	200/5	2A	EQS	No/No	-
ErgoEphorma+	200/20	1A	EQS	No/No	-
Auticon*	37/5	2B	ETU	No/No	PO
SYSteam*	60/25	2B	ESU	No/No	PO
Deriga	8/3	2B	ES	No/No	PO
NetIT+	9/2	1A	ES	No/No	-

Table 6.4: Interview Summary

> Tools/utilities are often placed in separate directories (util). This is done within each project, but seldom between projects, because nobody trusts the other developer(s) to have done a good enough job making the utilities. This somewhat defensive attitude is due to the fact that each developer stands responsible for the code he writes and reuses, and so each developer is careful when reusing other people's code.

6.2 Discussion of Results

I will not try to make any generalizations to the entire population from the results, but instead discuss these results alone. I will first look at the effects of code reuse mentioned by the interviewees, then the extent of code reuse and the organization of reuse. Finally, I will discuss the use of tools and procedures.

6.2.1 Effects of Code Reuse

All of the 24 interviewees mentioned increased efficiency or saved time and money (E) as an effect of code reuse. This was not surprising, as time is an important factor in software development, and efficiency is the one effect which is mentioned most often in the information I have gathered.

A total of 17 interviewees acknowledged some form of quality as an effect of code reuse: 6 cited higher general quality (Q) as an effect of reuse, 8 spoke of both stability (S) and quality (Q), while 3 mentioned improved stability (S). This makes quality (Q) the second most often mentioned effect of code reuse, being spoken of by 14 interviewees, and stability (S) the third in the list, with 11 references. As software is becoming increasingly important in the society, software quality is imperative, and the high "ratings" of quality aspects are perhaps both a result of and an advantage for this development.

7 interviewees named simplified testing (T) as an effect of reuse. 5 interviewees considered uniformity as a positive effect, while 2 mentioned that code reuse gives marketing advantages. A single interviewee spoke of more accurate time and price estimates as effects of reuse.

6.2.2 Extent of Code Reuse

14 of the 24 companies/development departments did not have a shared base of reusable code separated from other code (A), while 10 did (B). 9 interviewees reported that reuse is left to ach developer (1). 8 development

departments have a policy on reuse (2), while 7 companies have a policy on reuse (3).

The combination which seems to be the most interesting is a company policy on reuse (3), but not a common base of reusable code which is separated from other code (A). My impression before the interviews was that "serious" reuse would include some sort of a separate reuse storage. A company policy on reuse indicates a serious take on reuse, but the lack of a separate base of reusable code might seem a bit less "serious". Only 2 companies have this combination of policy and storage of code.

A combination which seems natural is a lack of policies on reuse (1) and a lack of a common, separate base of reusable code (A). This occurs in 9 companies. Another natural combination is a company policy on reuse (3) accompanied by a separate reuse storage (B). This is the case for 5 companies, which seem to be the most conscious of code reuse. There are 8 development departments with policies on reuse (2). 3 of these have no common base for reusable code (A), while there are 5 where a common base of reusable code exists (B).

6.2.3 Use of Tools and Procedures

When it comes to the employment of tools and procedures, there is a single company which stands out. TietoEnator is the only company which has a tool specialized for reuse. This tool, which was developed by the company, imposes certain procedures on code reuse. For a description of the system, see the interview transcription (section E.6). TietoEnator is an international company, and is also the only company with more than a thousand employees in Norway alone. The company has more than 15000 employees world wide. The size of the company explains their need for a formalized system for code reuse.

No other companies have tools which are created specifically for code reuse. Some companies, however, have informal procedures for code reuse. There were 3 interviewees who stated that the developers followed procedures for reuse.

6.2.4 Organization of Code Reuse

Again, there is a single company which stands out. TietoEnator is the only company which organizes reuse following a slightly advanced model; the company has a separate reuse project. The organization doesn't completely follow this model, described in section D.2; the reusable code is developed as parts of the projects, not by a separate, temporary reuse project. The

organization at TietoEnator is, however, more complex than the first model, where the projects identify and develop components and submit them to the reuse storage: At TietoEnator, there is a separate department which deals with code reuse in the sense of approval or disapproval of the component as fit for reuse and possibly authorizing investments.

There are no companies following the two more advanced models of reuse organization. 14 companies have no separate storage for reusable code, thus not following any of the reuse models listed in appendix D. As mentioned earlier, I feel this indicates a lack of serious commitment to code reuse. Most of the companies with no separate base for reusable code have no company or department policy of reuse, which strengthens the impression of less commitment to code reuse.

Finally, there are 9 companies where code reuse follows model 1; project oriented reuse (see section D.1). These companies separate reusable code from other code in a reuse storage. The reusable code is identified and developed by the projects, and it is stored in a separate reuse storage. This is the least complex way of reusing code in the set of models listed in appendix D, and it is the way I beforehand had assumed most of the companies reused code.

Chapter 7

Evaluation

In this chapter, I will evaluate the research process (section 7.1) and the process of extracting information from the notes I wrote during the interviews (section 7.2).

7.1 Research Process

As this research has been performed to form a basis for my Master's thesis and thus mainly for my own sake, I probably haven't been as diligent in ensuring the quality of the research process as I might otherwise have been.

7.1.1 Time, or Lack of It

An important aspect for all research is time. A few times during the planning and implementation of the research process, I was too optimistic when estimating the time needed to complete a task. Each time I had mis-estimated, the task at hand included contact with the interviewees or other people (e.g. switchboard operators). There were mainly two occasions where I underestimated the time need. One occasion was when I tried to get participants for the research; explaining the reason for my call and getting in touch with the right person took more time than I had thought. The other occasion was when I needed the interviewees to verify the interview transcriptions; some answered very quickly, while some used more time before returning an answer. There was also one occasion where I made a good time estimate: I scheduled the interviews one hour apart. Few of the interviews lasted that long, and I could spend the extra time left before the next interview making additional notes or preparing for the next interview.

7.1.2 Sample Selection

When I was trying to get participants for the research, I started with a list of 53 companies. This was reduced to a list of 25 companies where someone had agreed to participate. The most often cited reason for not participating was a lack of time. I had expected a higher rate of participation, although I was aware that time is a valuable resource in software development. The process of getting participants for the research was time consuming, and most of the time was spent (unsuccessfully) trying to reach the right person at the right time.

The list of 25 participants was reduced to 24, as one of the interviews could not be conducted. This was because I was unable to reach the interviewee over the telephone at the scheduled time and date, over the telephone at subsequent times and dates, and via e-mail. I don't know why this happened. In spite of several messages left at his answering machine and messages given to the switchboard operator, and one e-mail message, I have not yet received any response from the person in question.

7.1.3 Selecting the Right Research Method

When I planned the research, I wanted to conduct a survey. As I was planning the survey I contacted several software developing companies in Norway, and I tried to convince them to participate in the survey and set a date and time for a telephone call. One of the people I spoke to explained to me that he didn't have the time for an appointment later, but he had the time to participate at the time I contacted him. In a way, he helped me test the unfinished questionnaire I had created. The conversation made me realize that I had to re-think the way I communicated with the participants. One of the most important reasons for this realization was that the interviewee's company didn't have tools specifically developed for code reuse or formalized procedures for code reuse, although the developers considered reuse to be highly important.

My reason for choosing to perform a survey and develop a questionnaire was an assumption that reuse was performed in organized ways, with tools and procedures which are developed for code reuse. This assumption was based on the fact that reuse is not a new concept, and I believed that reuse was so common and important that many companies/development departments would have carefully considered the way reuse was performed. If reuse was performed in an organized, formalized manner, a questionnaire would be a good way of capturing information regarding code reuse. However, the conversation with the first participant made me realize that reuse was not commonly as organized as I had believed. Hence, I decided to use

an interview guide instead of a questionnaire. This enabled a more open conversation than the previously formulated questions of the questionnaire.

7.2 Extracting Information From Interview Notes

One of the things which is often more difficult in qualitative research than in quantitative research, is extracting information. I had written notes during the interviews, and I had to extract information from these notes to better be able to present and discuss the results. I chose to extract a minimum of information, as this would enable a more surveyable summary of the results. I could have included more information in the summary, such as which programming languages were used, and a classification of how domain specific the software development of the companies were, but I felt this would make the summary less surveyable and not contribute much to the understanding of how reuse is performed. The reasoning behind the selection of categories for extent of code reuse and effects of code reuse is similar; I have extracted information at a level which allows a general understanding.

Chapter 8

Conclusion and Further Work

As the final part of the report, I include the conclusion (section 8.1) and some suggestions for further work (section 8.2).

8.1 Conclusion

After performing the research, I feel that the process has been informative and I have reached the goals I had when I started working on the project (listed in section 1.3). My knowledge is of course specific to the group of companies and people I interviewed.

Code reuse is very common, but not commonly achieved by using tools and/or procedures especially developed for reuse. There are many reasons why software developers choose to reuse code; the most common reason is increased efficiency. Improved quality and stability, simplified testing, and uniformity are also reasons which are mentioned by many developers. All the interviewees feel that these reasons for code reuse are actually achieved effects as well.

This work has left me with a lot of questions and ideas for further work, which is the topic of the following section.

8.2 Further Work

During the course of this project a lot of new questions have appeared, which have given several ideas for further work (for example for my Master's

thesis):

- Do research on available tools and/or procedures specialized for code reuse (e.g. low-cost or free tools/procedures), and perhaps find out how the interviewees from this study would feel about using such tools.
- Use focus groups consisting of interviewees in this project as a means of gaining a deeper understanding of the results of this research.
- Look at patterns in relation to code reuse; e.g.: Are patterns more common than code reuse? Are the use ofs patterns perceived as simpler than code reuse?
- Take a closer look at the two companies which recently had started using new tools/procedures (Lydia, Fundator); how are things different after the change?
- Develop a system for code reuse, perhaps specialized for small to medium sized enterprises.
- Few people seem to be conscious of all the possible effects of code reuse. Contact the same interviewees and present them with a list of effects, and ask them to rate the effects on two scales; to what extent is the effect desired, and to what extent is the effect achieved.

Appendix A

E-mail Information

This appendix contains the information I e-mailed to the companies who wished information on the project. As I was loosely considering using focus groups at the time, I included information on this in the e-mails. All the e-mails I sent were in Norwegian (section A.1), but I include an English version in section A.2.

The contents of the e-mails varied slightly. As I realized that the time spent on each interview would be shorter, I reduced the time estimate to "about 30 minutes". Also, not all information was necessary to include in every e-mail, as I already had made arrangements with some of the companies when the e-mail was sent.

A.1 Norwegian

Hei.

Jeg heter Lisa Wold Eriksen og studerer datateknikk på NTNU (Norges Teknisk- Naturvitenskapelige Universitet) i Trondheim. Jeg utfører et prosjekt som del av studiene til Master of Science (Sivilingeniør). Dette prosjektet handler om gjenbruk av kode. I den forbindelse kontakter jeg deres bedrift i håp om at dere kan bidra med informasjon til mitt prosjekt.

Prosjektbeskrivelsen lyder som følger:
Gjenbruk av kode i objektorientert programvareutvikling.
"Fornuftig gjenbruk av kode øker hastigheten i utviklingsprosjekter."
(Rickard Öberg) I så måte er verktøy og metoder for gjenbruk av kode og erfaringer med dette viktig. Prosjektet går ut på å undersøke og vurdere dagens løsninger når det gjelder gjenbruk av objektorientert kode.

Denne beskrivelsen finnes også på http://www.idi.ntnu.no/undervisning/-prosjektbeskrivelse.php?id=389. Min veileder heter Tor Stålhane og er professor ved IDI (institutt for datateknikk og informasjonsvitenskap).

Hoveddelen av prosjektet vil basere seg på telefonintervju med representanter for ulike IT-bedrifter som utvikler programvare i Norge, og det er som nevnt derfor jeg kontakter deres bedrift. Telefonintervjuet vil ta mellom 30 og 60 minutt. Jeg ønsker å vite hvorvidt gjenbruk av kode eksisterer i bedriften, hva slags verktøy og/eller metoder som brukes for å gjenbruke kode og hvilke erfaringer bedriften har med dette. Den innsamlede informasjonen vil bearbeides og anonymiseres før den brukes i prosjektrapporten.

I tillegg ønsker jeg å ha nærmere kontakt med 5–6 bedrifter i Oslo og Trondheim for å få en dypere forståelse av deres erfaringer og forhåpentligvis også få anledning til å ha et personlig møte med ansatte i disse bedriftene.

I første omgang ønsker jeg fra deres bedrift å vite om det er aktuelt å bidra med informasjon. En betingelse for at informasjonen fra bedriften skal være interessant i prosjektet er at det brukes et objektorientert språk(*) i utviklingen av programvare. Dersom det er aktuelt for deres bedrift å bidra med informasjon ville jeg sette pris på å få oppgitt navn, e-postadresse og telefonnummer til en kontaktperson.

På forhånd takk!

Med vennlig hilsen
Lisa Wold Eriksen
E-post: lisaer@stud.ntnu.no
Telefon: 73939242
Mobiltelefon: 97070382

* Eksempler på objektorienterte språk er Java, C++, C#, Smalltalk etc. For en liste, se http://en.wikipedia.org/wiki/Object-oriented_programming_language

A.2 English

Hello.

My name is Lisa Wold Eriksen and I study computer science at NTNU (Norges Teknisk- Naturvitenskapelige Universitet, Norwegian University of Science and Technology) in Trondheim. I am carrying out a project as a part of my studies to become a Master of Science (Sivilingeniør). This project concerns reuse of code. To this end I am contacting your company in the hope that you will be able to contribute with information for my project.

The project description is as follows:
Reuse of code in object oriented software development.
"Proper reuse of code increases the speed of software development projects." (Rickard Öberg) In this respect, tools and procedures for code reuse and experiences with reuse are important. The project is about examining and assessing the present solutions for reuse of object oriented code.

This description can be found at http://www.idi.ntnu.no/undervisning/-prosjektbeskrivelse.php?id=389. My supervisor's name is Tor Stålhane and he is a professor at IDI (Department of Computer and Information Science).

The main part of the project will be based on telephone interviews with representatives from different IT companies which develop software in Norway, and this is as previously mentioned why I am contacting your company. The telephone interview will last from 30 to 60 minutes. I wish to know whether reuse of code exists in the company, what kind of tools and/or procedures are used to reuse code and what experiences the company has with this. The gathered information will be edited and anonymized before it is used in the project report.

In addition I wish to get more in touch with 5–6 companies in Oslo and Trondheim to achieve a deeper understanding of their experiences and hopefully also get the chance to have a personal meeting with employees of these companies.

To start with I wish to know if it is possible for your company to contribute with information. A condition to make the information from the company interesting for the project is that an object oriented language(*) is used in software development. If your company can contribute with information I would appreciate it if I could be given the name, e-mail address and phone number of a contact.

Thank you in advance!

Sincerely,
Lisa Wold Eriksen
E-mail: lisaer@stud.ntnu.no
Phone: 73939242
Mobile phone: 97070382

* Examples of object oriented languages are Java, C++, C#, Smalltalk etc. For a list, see http://en.wikipedia.org/wiki/Object-oriented_programming_-language

Appendix B

Unfinished Questionnaire

This appendix contains my first draft for a survey questionnaire. The original version is in Norwegian and is included in section B.1, while an English translation is included in section B.2.

B.1 Norwegian

- Hvilke programmeringsspråk bruker dere?
- Gjenbruker dere kode?

Dersom dere gjenbruker kode:

- Bruker dere noen spesielle verktøy for gjenbruk?
- Dersom dere bruker verktøy:
  - Kommer disse verktøyene med noen slags metode for gjenbruk? (Gå til metode-punkt)
  - Hva slags verktøy bruker dere:
    - Navn
    - Prodsert internt:
      - Er det til salgs? Hvis ja: Lisens, pris?
    - Produsert av tredjepart:
      - Produsent
      - Pris
      - Lisens

- Bruker dere noen spesielle metoder for gjenbruk?
- Dersom dere bruker metoder:
  - Hva slags metode(r) bruker dere:
    - Utviklet internt i selskapet
    - Utviklet av tredjepart - hvis ja: Pris?
- Hva er de ønskede effektene av gjenbruk? (Presentér liste)
- Hva er de opplevde effektene av gjenbruk? (Presentér liste)
- Hvor omfattende er gjenbruket?
  - Pålagt eller frivillig
  - Personlig eller felles
  - Enkelte utviklere, hele utviklingsavdelingen eller enkelte prosjekt

B.2 English

- What programming languages do you use?
- Do you reuse code?

If you reuse code:

- Do you use any specific tools for reuse?
- If you use tools:
  - Do these tools come with some kind of procedure for reuse? (Go to procedure point)
  - What kind of tools do you use:
    - Name
    - Produced internally:
      - Is it for sale? If so: License, price?
    - Produced by third party:
      - Producer
      - Price
      - License
- Do you use any specific procedures for reuse?

- If you use procedures:
  - What kind of method do you use:
    - Developed internally in company
    - Developed by third party - if so: Price?
- What are the desired effects of reuse? (Present list)
- What are the experienced effects of reuse? (Present list)
- What is the extent of reuse?
  - Imposed or voluntary
  - Personal or common
  - Individual developer, individual projects, or development department

Appendix C

Interview Guide

This appendix contains the interview guide I used. My interview guide was written in Norwegian, as the interviews were conducted in Norwegian. The original interview guide is included in section C.1. Section C.2 contains an English translation of the Norwegian interview guide.

C.1 Norwegian

- Programmeringsspråk
- Hvor omfattende gjenbruk (pålagt/frivillig, personlig/felles, enkelte utviklere/hele utviklingsavdelingen/kun enkelte prosjekt)
- Ønsket og oppnådd effekt(er)
- Verktøy?
- Metode/rutiner/prosedyrer?
- Generell beskrivelse av hvordan gjenbruk utføres i praksis

C.2 English

- Programming language
- Extent of reuse (imposed/voluntary, personal/common, individual developer/entire development department/individual projects)
- Desired and attained effect(s)
- Tools?

- Methods/routines/procedures?
- General description of how reuse is actually carried out

Appendix D

Reuse Models

Hauge ([8]) presents four basic models on how reuse could be organized, taken from Davenport & Probst ([9]). In this appendix I will describe each model briefly.

D.1 Model 1 — Project Oriented

Shown in figure D.1, model 1 is the most straightforward reuse model. A shared reuse storage is used to exchange reusable components between projects. Each project is responsible for identifying and developing components for this reuse storage.

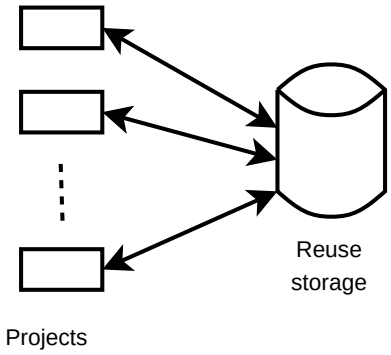

Figure D.1: Reuse Model 1 — Project Oriented

D.2 Model 2 — Reuse Through a Separate Project

Model 2 (figure D.2) is similar to model 1, as the projects identify possibly reusable components to add to a shared storage. Model 2 adds a level of complexity: The development of the reusable components is performed by a temporary reuse project, which then adds the components to the shared storage. When a component is added to the reuse storage, other projects can retrieve it from this shared storage, similarly to the previous model.

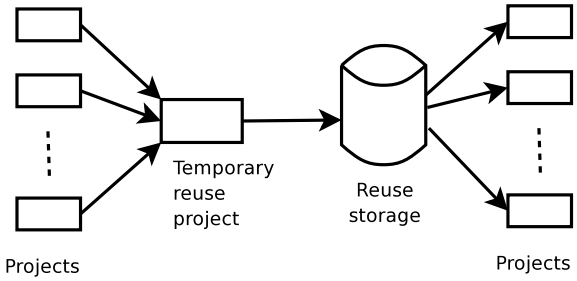

Figure D.2: Reuse Model 2 — Reuse Through Separate Project

D.3 Model 3 — Component Producer

In model 3 (figure D.3), all parts of the development of reusable components are the responsibility of a permanent department; the component producer. As in the two previous models, the reusable components are added to a shared storage and thus made available to the projects.

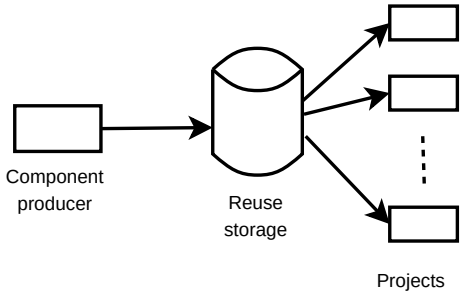

Figure D.3: Reuse Model 3 — Component Producer

D.4 Model 4 — Domain Producers

The most complex reuse model is model 4 (see figure D.4). This model is used in larger organisations who develop products within several domains. Domain specific reusable components are produced by a component development department for one or more domains.

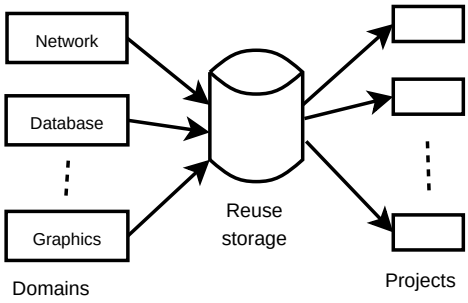

Figure D.4: Reuse Model 4 — Domain Producers

Appendix E

Interviews

In this appendix, the interviews are included. The appendix also includes some information about the interview transcriptions, including a list of the companies involved, in addition to the interview transcriptions themselves.

E.1 Interview Transcriptions

The following sections contains the transcribed interviews. First I present some information on the interview, such as company name, the name of the person I interviewed and the time I spent conducting the interview. Then I describe the company in brief; what the company does, where it is located, and number of employees. Finally the transcription of the interview is presented.

The text has been edited somewhat, as the original notes made during the interviews was highly informal (and in Norwegian). The original notes are not included, as they would not be useful/understandable to others than the author. If the original notes are needed for some reason, please contact the author to obtain them.

The overall elements of the transcription are as follows: First I list the programming language(s) used. The rest of the transcription covers the effects of reuse, both desired and experienced effects. The way reuse is carried out (including use of tools and/or procedures) is also described. Finally I have included other interesting pieces of information where appropriate.

E.1.1 List of Companies

When I made the appointments for interviews and conducted the interviews I used a list where I assigned each company a letter from A to Y. This helped me organize my work, and would also make it easier to make the information about some or all of the companies anonymous. I assigned the companies letters in the same order as I was going to conduct the interviews; the first company I interviewed was A, the second was B, and so on. Where some letter in the series is missing, I had an appointment for an interview but did not get to conduct the interview. The list of companies is included in table E.1.

ID	*Company*	*Web address*
A	Ajour Media AS	www.ajourmedia.no
B	Antares Gruppen AS	www.antares.no
C	ErgoSolutions AS	www.ergosolutions.no
D	Fundator AS	www.fundator.no
E	TietoEnator AS	www.tietoenator.no
F	Fronter AS	www.fronter.no
G	QS Manager AS	www.qsmanager.no
H	Escio AS	www.escio.no
I	Cintra Software Engineering AS	www.cintra.no
J	Lydia AS	www.lydia.no
K	adramatch asa	www.adramatch.no
L	Electric Farm ASA	www.electricfarm.no
N	MaXware AS	www.maxware.no
O	Finale Systemer AS	www.finale.no
P	Well Diagnostics AS	www.welldiagnostics.com
Q	AKVAsmart ASA	www.akvasmart.no
R	Vega SMB AS	www.vegasmb.no
S	Geodata AS	www.geodata.no
T	Egroup ASA	www.egroup.no
U	ErgoEphorma AS	www.ergoephorma.no
V	Auticon AS	www.auticon.no
W	SYSteam AS	www.systeam.no
X	Deriga AS	www.deriga.no
Y	NetIT AS	www.netit.no

Table E.1: Participating Companies

E.2 A — Ajour Media AS

The first interview I conducted was with Morten Ellingsen at Ajour Media AS. As mentioned in section 7.1.3, Mr. Ellingsen functioned as a test person for my survey questionnaire and the way I conducted the interviews. This interview took about one hour.

E.2.1 About the Company

Ajour Media AS is a small, Norwegian company located in Frei. The company develops a suite of software products for newspapers. The products are based on the Microsoft©.NET platform with XML and Web-interface. The staff consists of six persons, of which two persons handle the software delvelopment. Ellingsen is both the founder of the company and the development manager.

E.2.2 Transcription

Java and C# is used for developing client software and C# for server software. Ellingsen explained that Ajour Media exercises reuse to save time and, more importantly, to create stable software. The software becomes more stable because the code already has been verified and debugged. He stressed the importance of code reuse and told me that he spends more time on quality assurance than he does on the coding, as it is imperative to have products with as few flaws as possible.

No tools specialized for reuse are employed. Reuse is carried out by creating and using libraries and is chiefly module based. At first, Mr. Ellingsen explained the systems Ajour Media sells as consisting of a user part and a server side set of C# libraries which are made as general as possible. Later he explained that many of the software products they produce are based on Web services. C# is used for programming the Web services, which are the basis for Web applications. The Web applications, which he labels client software, consist of a Java or C# front-end and C# code-behind. There is less code reuse on the client side than in other parts of the system.

Ellingsen stated that he reuses as much as possible. He has been developing software for 13 years and has created many libraries. Some of them have been discarded, for example old libraries which are programmed in C and are not in use any more. When developing software he tries to find and collect general code which can be reused, for instance by searching through files and adding the code to libraries. This is also used as a quality assurance procedure; he goes through all code looking for code which has been written

twice. If he finds such code he creates a new library or uses an existing library.

E.3 B — Antares Gruppen AS

The second interviewee was Mr. Jon Bråtømyr who is a senior system consultant at Antares Gruppen AS. The interview lasted for about a quarter of an hour.

E.3.1 About the Company

Antares Gruppen AS is located in Oslo. The company employs approximately 30 persons, of which 17 work with software development. Antares Gruppen offers software engineering with main focus on portal solutions for intranet, customers and partners, handheld solutions and interactive TV systems. The company also does consulting, mainly focused at eServices for the public sector and eBusiness. Antares is in addition providing services within systems development based on Java/J2EE/J2ME, C++ and Oracle development tools and are moreover providing consulting services related to Microsoft Windows 2000/2003 as a certified Microsoft partner.

E.3.2 Transcription

Several programming languages are used, mainly Java; J2EE. Mr. Bråtømyr informed me that at the present most of the software development is done in connection with their consulting services, and for this reason re-use is mainly up to each individual developer. He also informed me that the company in time will be developing more software which is not client specific. As a consequence of this, re-use will have to be more coordinated and organized.

Bråtømyr mentioned several reasons for re-using code; it saves time, it helps achieve a better solution, the developer knows that the code works and most of the brainwork has already been done. He observes that these are not only reasons for re-use, but also the effects of re-use.

Currently no specific tools or procedures for re-use are in use. The re-use is highly disorganized, with some code placed on a CVS server and some on network disks. There are no standard libraries. Re-use is achieved by "cut and paste" or by using entire classes. When re-using a class, however, oftentimes modifications to the code are needed, and so also class re-use involves some level of "cut and paste". There have been frequent unsuccessful attempts at obtaining and using routines for re-use. Mr. Bråtømyr believes the lack of success is due to the fact that there is no great need for such procedures as of yet. As mentioned earlier, this will probably change in the future. To this end he spoke of a framework and libraries for re-use and routines for accessing and modifying these.

E.4 C — ErgoSolutions AS

At ErgoSolutions AS I interviewed Ken Dahle, who is a senior consultant. We spoke for half an hour.

E.4.1 About the Company

ErgoSolutions is located in Oslo and develops, integrates and administers business critical IT systems for larger businesses. The company offers services throughout the IT project's life cycle; from consultancy and analysis to project management, development and efficient administration. ErgoSolutions is a part of ErgoGroup which is owned by Posten Norge AS. The company has 300 employees. 150 of these work with software development.

E.4.2 Transcription

Several object oriented programming languages are used in software development; mainly Java, but also Delphi and C++. There is no company policy regarding re-use, but the developers find it natural to re-use code. On a higher level, it's up to the architect or designer of the program to decide which patterns should be used. The Integrated Development Environment (IDE) used by the developers helps develop frameworks for these design patterns.

There are several reasons for reuse on different levels. Designers are mainly concerned with patterns, while code reuse is employed by the software developers to avoid doing the same thing twice. Mr. Dahle explicitly stated that they do not reuse to save time on testing, as testing is important. There is, however, a positive effect from reuse on testing: Some parts which are reused are familiar and the developers and testers have already developed tests for them. Hence it is not necessary to make new unit tests for that specific part. ErgoSolutions experiences that the time saved due to reuse in relation to the total time of developing a software product is marginal, but measurable.

Reuse is common in software development at ErgoSolutions. The developers start with pre-existing code and adjust it, and the structure is maintained while the business logic is altered. Each developer decides for himself where he wants to start and how he wishes to reuse. Commonly the developers use components that are already known to them. Mr. Dahle explained that the developers have generated libraries for specific functionalities. The libraries are maintained in one location and used by several applications. They contain functionality which is so specific to one area that it is not necessary or

desirable that each developer or development department has deep knowledge of that area. One example of such functionality is packages for Public Key Infrastructure. Mr. Dahle explained that he feels optimal reuse is when something is produced once and then used by others as a utility. An example of this is, as previously mentioned, when someone with expertise in a certain area produces a module which can be reused by other systems, e.g. a module for certificates which was produced for one system and then used by other systems.

Frameworks are often reused as they consist of components with functionality which is familiar, always needed, and which there is no need to redo. When it comes to tools, different versions of JBuilder is used in development projects. No tools or procedures developed especially for reuse are employed.

E.5 D — Fundator AS

Interview number four took 20 minutes. This time was spent interviewing senior consultant Geir Domaas at Fundator AS.

E.5.1 About the Company

Fundator AS is located in Trondheim. The company has 17 employees of which 14 are working with software development. Fundator delivers IT solutions and provides consultancy services in system development and system integration.

E.5.2 Transcription

As Fundator is a consultancy firm, the programming languages used by the developers vary with the customers' needs and demands. .NET, Java, and PHP were most commonly used for software development at the time of the interview.

At Fundator the developers reuse mainly to save time, provide the customer with a cheaper solution, and produce a solution with higher quality. They feel that these are both desired and achieved effects of reuse. Time is saved during tests and by spending less time writing the code. The quality is improved as the reused modules are commonly more thoroughly tested and the code is improved as the developers go through it several times. Another positive effect is that the company can provide a more accurate and possibly lower price for a system if it is similar to something they have already produced. Also, the experience with similar systems and the possibility of reuse can be used in marketing the company.

The level of reuse varies with the projects. Unless the customer has demands on reuse, the decision to reuse is entirely the developer's own. Mr. Domaas mentioned that generally reuse is complicated and often isn't applied. Fundator is, however, in the final stages of adopting an internal quality control system which accentuates reuse. This system was developed by Fundator and is called "Fundator Unified Process" (FUP). FUP is based on Rational Unified Process (RUP), which is quite extensive. Instead of including everything from RUP, Fundator has selected the artifacts from RUP which are the most suitable for the kind of projects carried out at the company. Before the creation of this system, there wasn't much structure or a common system for reuse. Libraries existed, but to get hold of code to reuse, the developer wishing to reuse the code had to call on the developer who had produced the code. No tools or procedures specific to reuse

were employed company-wide, and reuse was achieved by reusing text (i.e. code snippets), files (e.g. classes), file structures (e.g. packages) or entire modules.

E.6 E — TietoEnator

At TietoEnator AS I interviewed Morten Brurberg, Head of Department. This interview took 20 minutes.

E.6.1 About the Company

TietoEnator is an international consultancy firm which provides consulting, developing and hosting services for its customers' digital businesses. They employ more than 14000 people world wide, in more than 20 countries. 1100 employees are located in Norway, of which about 500 work with software development.

E.6.2 Transcription

Most of the programming is performed in Java. Reuse is not imposed on the developers, but it is part of the company strategy to reuse as much as possible. As Mr. Brurberg stated: Everybody depends on reusing what they've got to be able to compete both among countries and for customers.

The reasons for reuse are desired efficiency and quality; when reusing a component the developers already know something about the qualities of the component, such as stability, scalability and performance. The achieved effects of reuse mostly match the desired effects, but sometimes a component might not be as suitable for reuse as it initially seemed, resulting in lower rather than higher efficiency.

TietoEnator has produced an internal company-wide system for code reuse, called "Digitalizing Framework". The company has invested substantial amounts of resources in this system. It is rooted in the senior management and the company has a separate organizational unit in charge of the system. At the base of the system is a repository, while the users access the system via a web interface. This interface allows the users to access existing reusable components and add suggestions for new components. The components are mainly large and solve general tasks, but the repository also contains smaller components and classes. The repository consists of several sections, for example one section for general company-wide components and different sections for different countries with components which in some way are specific to that region.

The Digitalizing Framework is a tool for reuse which also introduces some procedures. When a developer considers a component to be reusable, he or she commits a suggestion for review. This is done by filling out an electronic

form with a description of the component. The review is performed by someone in a central part of the organization and the review process ends with approval or disapproval of the component as fit for reuse and possibly authorizing investments (for example allowing the developer to spend extra time developing the component). There are no rules concerning which of the existing components in the Digitalizing Framework that should be reused or which new components should be committed; this is up to the developers working on each project.

E.7 F — Fronter AS

The fifth interviewee was Are Pedersen, who works as a Development Teamleader at Fronter AS. The interview took about a quarter of an hour.

E.7.1 About the Company

Fronter AS was founded in Norway in 1998, and is now represented by partners in several European countries. Fronter provides Virtual Learning environments which enable creation, management and sharing of knowledge and help students and teachers to work together regardless of their physical location. Fronter has 24 employees and 1 software developing department with 6 employees.

E.7.2 Transcription

At Fronter most of the programming is done in PHP, but PHP's object oriented features are only partially utilized. Some Java is also used for programming applets. The developers try to reuse whenever possible, and Mr. Pedersen remarked that reuse is easiest when dealing with object oriented code. The desired effects of reuse are a consistent manner of operation when it comes to appearance and functionality and also reducing the number of errors. In addition to these effects, the developers experience that reuse can save some time.

Fronter is mainly developing one single application system, which means there are a lot of basic libraries which have to be used by everyone. Whenever something new (an addition to the system) is produced, the developers try to reuse as much as possible in addition to the common libraries that always have to be used. Every time something is produced, the developers perform a code review to ensure that the right code has been (re)used. In addition to the reviews the developers hold meetings to plan new additions to the system, and a part of this planning is done to discover what can be reused. Both these procedures include reuse, but neither of them are specific to code reuse. The developers don't employ any procedures specific to code reuse.

No tools for reuse are utilized, the developers either "cut and paste" code (which is generally avoided) or expand the functionality of already existing code. The developers have, however, created a Website which functions as a work of reference, a place where they search for information and documentation when they are programming. The Website can provide information as to whether something has been done earlier and presents a description of

how it can be done, possibly with a hint to where reusable code might be found.

E.8 G — QS Manager AS

At QS Manager AS, I spoke to Tor Sætrang, who is the founder and general manager of the company. The interview lasted for ten minutes.

E.8.1 About the Company

QS Manager AS is a Norwegian company located at Hamar. They have 3 employees, of which 2 work exclusively with software development. Due to turbulence in the trade, the company is organized as flexible as possible, allowing 5 people in total to work with their products. The company has a product line of products for quality assurance, documentation and "Asset/Resource/Inventory Management". Their products support the administration of important resources in businesses and increase the efficiency of the processes involved in quality assurance.

E.8.2 Transcription

The developers at QS Manager mostly use the programming language Clarion, but Visual Basic and Java are also occasionally used. Mr. Sætrang explained that the decision to reuse is natural to the developers. A good programmer is "lazy" — the less time and energy wasted, the better. In this way, reuse is a result of each programmer's desire to achieve his/her goals quickly, and it is up to each programmer to make the decision to reuse.

No tools or procedures specific to reuse is employed in software development. As previously mentioned, the decision to reuse is left up to each programmer. The developers use each other's code as well as their own, often by "cut and paste". The fact that the company produces primarily one application means that all the developers have access to the source code of this application and (re)use components either directly or by "cut and paste".

E.9 H — Escio AS

At Escio AS, I spoke to Espen Holje, who works as a Product/Development Manager. This interview lasted circa ten minutes.

E.9.1 About the Company

Escio is a Norwegian company located in Hamar, and is an internet company in the concern called "Sulland Gruppen". Their main focus is on developement of webbased solutions based on their content management product EasyPublish CMS. Although the company has a strong focus on developing their core products, they also provide consultation and project management services. Escio has 7 employees, including 2 who develop software.

E.9.2 Transcription

The programmers at Escio use PHP when developing their browser-based content management platform, which is their main product. Mr. Holje informed me that reuse saves time, both in terms of having to write less code and already having used and tested the reused components.

Reuse is the basis of Escio's business, reuse is the rule rather than the exception. Experience indicates that the degree of reuse should be about 90-95%, mainly based on the fact that their customers have similar problems. The developers always consider how they can produce their software as "off the shelf" software and parameterize as much as possible to enable reuse. In a project developing software for a small to medium size Web site, the company can deliver a solution which is 100% "off the shelf". In larger projects, there is about 5–10% development of new code.

As for the remaining 5–10% code which is not "off the shelf" the developers analyze the code to see if it has general interest for generalization. Code that may be reused is then generalized and implemented into their content management product by parameterizing the functionality and placing it into an object repository. In a 3-tier model, with separated business logic, this enables the programmers to easily reuse the components. Components which produce output, e.g. an object displaying an article or a list of articles on a website, extends a base class which covers data transformation services for easy deployment on any site. All source code is under control of a versioning system.

E.10 I — Cintra Software Engineering AS

At Cintra Software Engineering AS, Tomm Scüller was the interviewee. He is Head of Development. This interview took about ten minutes.

E.10.1 About the Company

Cintra Software Engineering AS is a two man software development firm located in Porsgrunn. Most of the software products developed at Cintra are produced for clients (and may be sold by clients instead of directly from Cintra). The company also offers counselling, problem-solving, courses et cetera.

E.10.2 Transcription

The programming languages used at Cintra are Borland Delphi and C# .NET. The company has a policy to reuse code where appropriate. Desired effects from reuse are saved time and higher quality. The developers feel these are not only the desired effects, but also the real effects. Another reason for reuse is that the company's clients have different needs. This means that different functionality is needed, making it easier to have several slightly different parallel systems (reusing code) than assigning all the functionality to one single, larger system.

At the present time inheritance is used to a certain extent, but the developers wish to make even more use of inheritance and use a single code base. In addition to using templates for inheritance, "cut and paste" is also used if appropriate. Previously, component libraries in Delphi have been used with success, but this is not as useful anymore.

No tools and procedures specifically created for code reuse are applied. The developers keep the shared source code in a library with version control. Generally, the two developers keep a running dialogue to ensure that they both know what the other is doing.

E.11 J — Lydia AS

Interview number ten was with project coordinator Monika Lie Larsen at Lydia AS. The interview lasted for nearly a quarter of an hour.

E.11.1 About the Company

Lydia AS is a Norwegian company located in Trondheim. They develop and implement software for administration, operation, maintenance and development of buildings and property. Their software is called Lydia®, and the company performs development, project assignments, installation, training, and other services connected to this software. Lydia AS has 11 employees, of which 4 work with software development.

E.11.2 Transcription

The developers at Lydia program in C#, Tcl, and C. The company is in the process of adopting a new system for development. Ms. Larsen depicted the change as going from spaghetti to lasagna, i.e. a change from disorder to a more organized system. A strategy for reuse is embedded in the organization of the class hierarchy in relatively small groups/assemblies which reflect business areas and layered structure in the application. Previously, reuse has not been common, but this will change with the introduction of the new system.

Ms. Larsen stated that it goes without saying that code has to be reused when working with software development. The effects of reuse are saved development time and more predictability in testing. Also, it is important to make developing fun, and get more focus on development and less focus on error correction.

One of the effects of introducing the new system is that a procedure for reuse will be applied: All developers when developing software will have to check if something similar has been created earlier. If not, they will have to create a class themselves, include documentation in the code, and produce a test class with standard tests for each access class.

No tools specifically produced for code reuse are employed.

E.12 K — adramatch asa

At adramatch asa, I interviewed Ketil Grytten, who is a software developer. The interview lasted for twenty minutes.

E.12.1 About the Company

adramatch is an international company which develops and distributes software for automatic reconciliation (automatiske kontoavstemminger og automatisk oppdatering av innbetalinger uten KID). The company headquarter is in Oslo, and there are subsidiaries in Stockholm, Copenhagen and London. adramatch has a total of about 35–40 employees. 3 of these work with software development, all of them located in Oslo.

E.12.2 Transcription

At adramatch, the developers program in C#, C++, and Visual FoxPro. Mr. Grytten explained that as they make a niche product, the developers want the produced components to be reusable in several programs. For this reason, the developers use the object oriented functionality in C# to be able to reuse components.

Effects of reuse are saved time on development and testing. When a reusable component exists, there is usually no need or desire for further development or changes to the component. The developers try to keep further development to a minimum, as this will ensure that the reusable components in the long run will become highly stable.

Mr. Grytten stated that the developers reuse to the best of their abilities. No procedures or tools created specifically for code reuse are employed, but generally the developers make the components they produce as general as possible to avoid changing them at a later time, and a pattern called Abstract Factory is employed to create flexible software products.

E.13 L — Electric Farm ASA

Interview number twelve was with Tor Haugen, a senior developer at Electric Farm ASA. The interview took about 35 minutes.

E.13.1 About the Company

Electric Farm ASA is located in Oslo, and is a part of ErgoGroup which is owned by Posten Norge AS. Electric Farm is a consultancy firm specializing in development of Web-based services and solutions. The primary products of the company are Web solutions for interaction with and administration of larger databases, electronic commerce, finance and stock services, project management, and content management systems. The company has 23 employees, of which 19 are software developers.

E.13.2 Transcription

The programming language used at Electric Farm is C#. The company has a relatively general policy on reuse; the developers should try to reuse code. Beyond that, deciding how to reuse and what to reuse is left to the development department and the individual developers.

Previously, reuse was not common; mainly a few developers with an initiative advocated code reuse. These developers have collected general code in the form of reusable components in a joint framework project. As this base of reusable components has evolved, the desire to reuse has spread throughout the development department, and at the present time, the reusable components are utilized in the company's .NET projects. The transition from ASP and Visual Basic previously used to C# and .NET technology was a contributing factor to the growing success of the base of reusable components, as it has made reuse easier. It was only when the developers started using object orientation that any significant level of reuse was reached.

The reusable components in the framework project are open for anyone to develop further, and everything is subject to version control. There is also a folder of already compiled components and help files/API documentation to be used when no changes to the source code are necessary. No tools specifically created for reuse are employed, primarily due to the lack of central initiative. Because of this lack of initiative, reuse has been introduced gradually, at first in individual projects and later in the common framework project, and time has not been invested in researching and selecting a tool for reuse.

There are no established procedures specifically for reuse. The developers have a rudimentary code standard (naming conventions, basic patterns, et cetera) which is partially verbal and partially in writing. The code standard is based on a common understanding of how things are organized in namespaces (file structures and ways of organizing code in functional areas) and what kind of patterns should be used to solve individual problems. Before anything is added to the reuse framework, it has to go through peer reviews by several developers and consensus has to be reached that the code follows the code standard and is reusable.

There are several viewpoints on the desired effects of reuse. The management focuses mainly on shortening the development time in order to deliver the products faster and/or improve profits. The reuse of components means that the company has modules for and experience with tasks and functionalities. This is an advantage for the developers saving time and the management saving money, but is also seen as an advantage from a marketing point of view.

Mr. Haugen has realized that there is another aspect of reuse which is just as, or even more, important: Reuse results in components of higher quality. Some errors are always made when the components are developed, but as the components are reused, these errors are fixed, and the result is more stable components. This also means that time can be saved during tests: Often development and preliminary testing of components happen simultaneously as the developers produce the components, and this preliminary testing is not necessary for the reused components.

Reusable components can also improve the process of starting a new project. When the members of the development team meet to start a project, they have to get an idea of what is to be done. Sometimes the developers may feel "lost" and it may seem like they have to start from scratch. In this situation, it is useful to have reusable components from a previous project which resembles the new project. This makes things easier for the developers by reducing the need for discussions and clarifications of how functionality is to be implemented et cetera, as some parts already are implemented. The resemblance to the older project and reusing code from that project help make the new project more "recognizable" as well, and in this way, reuse helps create continuity between projects.

E.14 N — MaXware AS

At MaXware AS I spoke with CTO Tor Even Dahl. The interview took nearly 20 minutes.

E.14.1 About the Company

MaXware is an international company, delivering vendor neutral Identity Management solutions utilizing the customers' existing infrastructure. The company's targeted markets are telecommunications, military/defense, government, postal, oil & gas, and finance & insurance. The company has 33 employees, of which about 10 work with software development. All of the software developers are located at the headquarter in Trondheim, Norway.

E.14.2 Transcription

The developers at MaXware primarily use Java, but also Visual Basic, C, C++ and PHP when producing software. As they develop software for different markets, there isn't extensive reuse of components. Hovewer, they try to reuse third party open source software where appropriate. No internal library of reusable components exists, and when code actually is reused it is done by "cut and paste", as they reuse small parts of code more often than entire classes or components. The developers know each other well and they have good communication internally in the development department regarding what each developer is working on. This is one of the reasons why they feel it is not necessary to create a library of reusable components which they have produced themselves.

Mr. Dahl reported that the effects (both desired and achieved) of code reuse are saved time and money, and optimization of the code. To the developers at MaXware, speed is important; everything has to be done rapidly. The developers try to reuse both competence and code, and it is common to reuse the program logic instead of the program code itself.

There are no tools employed which are specific to code reuse. The developers feel their department is so small that they don't need a specialized tool, and that such a tool would be expensive and not very useful to them. There aren't any reuse procedures either; it is up to the individual developer to reach his/her goals as quickly as possible, based on what has been produced earlier. There are, however, directions as to how general development should be done, and during the development, the developers meet to discuss how problems should be solved.

E.15 O — Finale Systemer AS

At Finale Systemer AS I interviewed Head of Development Frank Mikalsen. The interview lasted for a quarter of an hour.

E.15.1 About the Company

Finale Systemer AS is a Norwegian company with headquarters in Tromsø and subdivisions in Oslo, Årviksand, and Skjervøy. Finale Systemer produces software for account reporting and for tasks in connection with tax calculations. Their software products are independent applications which consist of modules. These modules offer functionality through interfaces, and a module can be used in several applications. Examples of applications are FINALE Årsoppgjør (annual settlement), FINALE Skatt (tax), FINALE Konsern (concern), and FINALE Investor. The company has 14 employees, of which 8 work with software development.

E.15.2 Transcription

At Finale Systemer, the developers use Borland Delphi and C++ when producing software. The software has a modular component based structure. There are two levels of reuse; an application core which is accessed via an interface, and a code repository where the reusable code is separated from non-reusable code. The development department has started a project to document code (functions, what is available) which has been reused up to this point.

It is up to the individual developers to reuse as much as possible. The effects of reuse are fewer (re)written lines of code and higher quality of the code — both new and reused code. Mr. Mikalsen stated that while there haven't been any measurements of this, he feels that over time, reuse results in higher quality.

No tools or procedures which are specialized for code reuse are employed. It is left to each developer to assess whether the code or component in question is of interest to the rest of the application field. The components which are easiest to recognize as reusable (or not) are components or functions which belong in the architecture; it's easily spotted whether the component is usable in other applications.

Using Extreme Programming (XP) and working in pairs help spread knowledge of what code already exists. When the developers reuse code, it is very

rarely done by "cut and paste". Mostly, the developer uses a previously produced component which is made available through an interface.

E.16 P — Well Diagnostics AS

Interview fifteen was conducted with Senior System Developer Hårek Ryeng at Well Diagnostics AS. This interview lasted for a quarter of an hour.

E.16.1 About the Company

Well Diagnostics AS is a Norwegian company, located in Tromsø. They develop, market, and sell solutions which enable secure exchange of sensitive patient information. Their main product is Well Communicator. The company has 10 employees, and 5 of these are software developers.

E.16.2 Transcription

The developers at Well Diagnostics primarily use Object Pascal, but they also use Java, C++ and C# when developing software. The prevailing approach to reuse is to reuse to one's best abilities. Mr. Ryeng stated that software developers are "lazy"; all programmers using object orientation wish to be successful, and they thus produce code which is reusable.

An effect of reuse is saved time, which is partially due to the ability to use automated tests; if changes are made to the reusable piece of code, automated tests can be used to check if the altered code performs as required. Time and effort are also saved when the developers don't have to rewrite the code several times. Another effect of reuse is higher quality of the end product. All the aforementioned effects are both desired and achieved.

Mr. Ryeng explained that there are some problems connected to reuse as well. A problem the developers might have, is that it often takes more time to produce reusable code which is general and of high quality than non-reusable code. When, in addition to this, there is little time, the result might be that the code is not made reusable. Often it is difficult to reuse code across several products. One aspect of this is changes; if there is one component which is being reused across several products and the component is changed, this affects all the products. This could be a good thing if it is desirable to have the same functionality in all the products, but it turns into a problem when the developers wish to change the component in only one of the several products using it. He also feels that it is often merely parts of a reusable component which are actually reused.

No tools specifically made for reuse are used. The developers have procedures for testing the code: The code has to be subjected to tests which are both detailed unit testing at class level and tests of functionality, and

the tests are to be executed once every 24 hours. The developers have a library of code which they have produced themselves, but they also reuse code produced by a third party. The library consists of both binary and textual content (compiled code and source code).

E.17 Q — AKVAsmart ASA

At AKVAsmart ASA I interviewed program developer Allan John Alfheim. The interview lasted for about 20 minutes.

E.17.1 About the Company

AKVAsmart ASA is a technology company with activities in the fish farming industry. The company's two areas are "Farm Process Technology" and "IT & Consulting". The main products delivered by the company are feeding technology and production management systems. The head office of AKVAsmart is located in Bryne, Norway. "IT & Consulting" is managed from Trondheim. The company has established subsidiaries in all major markets outside Norway; Chile, Scotland, and Canada. The company has 100 employees, of which 12 are software developers.

E.17.2 Transcription

The software developers at AKVAsmart primarily use C# (.NET) and Delphi when they produce software. Mr. Alfheim explained that each developer has a free hand when it comes to reuse, but they have tried to reuse as much as possible and to produce general code.

The desired and achieved positive effects of reuse are the same. According to Mr. Alfheim, the most important effect is the prevention of errors; fewer errors are made when code for common functionality is collected in a single place. The developers also save time, as they don't have to write the same code more than once. Another effect is more readable code.

Some general problems with reuse were also mentioned. First, it might be difficult for new developers to know what exists of reusable code. Second, reuse requires more planning than developing from scratch. Also, if time is short, reuse gets less focus.

They use no specific tools or procedures to reuse code. All code is organized in a source control program. Mr. Alfheim explained that in the big picture, the developers try to make the design as good as possible to enable procedures/methods to be reused. On a more detailed level, each developer decides how he/she wishes to reuse code. Both while writing code the first time and afterward, the developers try to make the code as reusable as possible.

When the developers are writing code, they might discover that code with the same functionality exists someplace else. They then do something called

refactoring; extract the given code from both places and add it to a separate routine which is called from both the previous places in the code. Reuse is done with entire classes as well as methods/procedures.

E.18 R — Vega SMB AS

For the seventeenth interview I spoke with Marius Seglsten, who is a Lead Developer at Vega SMB AS. The interview lasted a quarter of an hour.

E.18.1 About the Company

Vega SMB AS is located at Lysaker, and produces a software product which is designed to improve the buyer's customer interactions. The company has 9 employees, of which 4 work with software development.

E.18.2 Transcription

The developers at Vega SMB perform software development using C# .NET. Some programming is also done in Delphi and Visual Basic, but this is mostly maintenance of previously produced software. The desired effects of reuse are that the developers don't have to do the same work more than once, and increased quality of the software. Mr. Seglsten explained that traditionally, the developers have not reused much code. They try to develop code which can be (re)used by others, but this turns out to be difficult in practice.

The company has recently gone through some changes, which have also affected the developers' ability and will to reuse source code. Previously, it was common to reuse third party (compiled) components. This turned out to not work so well, as many of the components were too general to be useful. At the present time, however, the developers try to reuse their own code. They extract code which can be made generic into reusable components, e.g. entire functional areas which can be used in more than one application. The reuse of components is still binary based (compiled code). When it comes to reuse of source code, mostly small chunks of code are reused. This is generally accomplished by "cut and paste".

There are no tools or procedures specifically made for reuse which are employed. The developers communicate a lot, informing each other of what they are working on, which makes the others aware of things that can be reused. Also, when a developer is curious as to whether something has already been produced and can be reused, he/she obtains this information simply by asking the other developers.

E.19 S — Geodata AS

At Geodata AS I interviewed Technical Manager Håvard Tobiassen. The interview lasted for 20 minutes.

E.19.1 About the Company

Geodata AS is a Norwegian company, with its headquarter in Oslo and a department in Hønefoss. The company is a supplier of Geographical Information Systems (GIS), and supplies IT-systems and services for administration and use of geographical information. Geodata has 80 employees, of which 22 work with software development. Including project managers, testers et cetera, 35 people are involved in developing software solutions.

E.19.2 Transcription

The developers at Geodata use Visual Basic .NET and Java. About one third of the developers use Java, while the rest use Visual Basic .NET. The level of reuse differs a bit between the two developer groups and between projects; the Java group has a slightly better track record of reuse. Each group has one person in charge. This person shall take the initiative to guide the developers in the direction of reuse, on an administrative level.

There are several desired effects of reuse. Improved efficiency when it comes to time and getting a better economy in the projects is important. The quality is improved, and the competence and uniformity of the developers' ways of doing things are improved; reuse ensures that even new developers do things the same way as the experienced ones. At the present time, the desired effects are the same as the achieved effects, but this was not always the case. There have been phases where the developers have felt that they've spent a great deal of time on reuse without getting the results. It takes some work before the critical mass of reusable code needed to see the positive results of reuse is reached.

The Web and Java developers have a common code library where components from projects are stored. The Visual Basic .NET group hasn't got a library which is quite as extensive, but they are in the process of creating a framework and code library for reuse. All source code is stored in a version control system, where the reusable libraries are separated from the rest. No tools specialized for reuse are employed. The developers don't use any reuse specific procedures in addition to the normal code and documentation standards.

E.20 T — Egroup ASA

At Egroup ASA, I interviewed Torbjørn Pedersen, Manager Software Development. The interview lasted about a quarter of an hour.

E.20.1 About the Company

Egroup ASA is a Norwegian company with 200 employees who are distributed amongst offices in i Oslo, Brumunddal, Fredrikstad, Sandefjord, and Porsgrunn. The company is a consultancy firm which delivers solutions, services and products for cost effective establishment, operation and management of information and communication technology solutions. Egroup has a dedicated software development department responsible for the company's own Corporate Portal Solution, named eWay. There are five employees in this department.

E.20.2 Transcription

The developers at Egroup primarily program in Visual Basic, and some C# .NET. Mr. Pedersen reported that there is a high degree of reuse amongst the developers, and proceeded to explain that the developers produce library routines which are general enough to be reusable, and which are often gathered in libraries. The units of reuse are typically classes (libraries).

The desired effects of code reuse are efficiency and increased quality; extra work and time spent to develop good solutions result in a higher development efficiency later. Other positive effects of reuse are that the developers don't have to write the code over again, and they know that the code works. Reuse also reduces maintenance, as the components become stable over time. Mr. Pedersen stated that the developers achieve these effects of reuse. Another positive effect, he mentioned, is that the developers can specialize in different fields; every developer doesn't need to know everything about how to solve a task when a component to handle the task already exists.

Reusable code is not separated from other code, all code is stored as libraries in a version control system. No reuse specific tools are employed. When it comes to procedures for reuse, Mr. Pedersen explained that the developers don't use any distinct procedures for reuse, but some procedures are worked into the programming languages and tools they use.

E.21 U — ErgoEphorma AS

The twentieth interview was conducted with Advisor Thomas Lünell at ErgoEphorma AS. The interview lasted about three quarters of an hour.

E.21.1 About the Company

ErgoEphorma AS is a part of ErgoGroup which is owned by Posten Norge AS. The company offers solutions to simplify electronic cooperation in and between administration, local authority (kommuner), industry and inhabitants. These solutions cover communication, administration and message relay. ErgoEphorma also offers analysis, consultation, and help with customers' databases. The company has about 200 employees, of which 20 are software developers.

E.21.2 Transcription

The developers at ErgoEphorma use several programming languages. Mr. Lünell uses Java and Perl, while others might prefer C or .NET languages. There is no company policy on reuse, but reuse is generally accepted as having positive effects on software development. More and more of the reused code is developed by a third party, as the developers try to find existing solutions instead of developing their own. As Mr. Lünell put it, "we don't try to re-invent things". Often things have to be done in a certain standardized way, which means there is no point in doing these things over and over. Patterns might be used to force somewhat better code reuse, e.g. J2EE-patterns. The developers have tried to use patterns, e.g. decoupling of layers, and this results in better components which are easier to reuse.

The threshold to understanding components which are reused might be a bit high, meaning the developer has to spend some time before he/she understands the component/code. This is, however, countered by the fact that the developer saves the time he/she would have spent writing the code him-/herself. When everything is added up, reuse saves time.

The most important effect of reuse is improved stability, as stability is highly important to ErgoEphorma's systems — "100%" uptime is desirable. Efficiency also matters, but not as much as stability. Improved quality is important, and related to stability. These positive effects of reuse are achieved by the developers. But the positive effects are not the only reasons for reuse: It's possible to reach a phase in a project which is so large that the developers *have* to reuse code. The reused code doesn't necessarily have to be

object oriented (Mr. Lünell mentioned COBOL as an example), as long as the logic fits. Reuse is then done by migration or porting.

A negative effect of reuse is possible loss of competence. How something works is easily forgotten, and as components are reused over and over, the developers might forget the reasoning behind the component (how does it work, why is a functionality implemented in a certain way, etc.). In this case, documentation is essential. It might also be an advantage to sometimes rewrite code in order to maintain a certain level of knowledge.

Reuse is often done by copying; developer A can ask developer B to send him/her a zip file containing reusable components, and sometimes a module is good enough to be reused in the system developer A is making with only minor changes. Mr. Lünell mentioned command patterns as a good way of reusing; whatever is common to different classes should be taken care of by a superior class, from which the other classes inherit.

The developers have toolboxes consisting of components, e.g. components which are repeatedly used by several other components. Often simple interfaces are created, a framework is established, and code is cut, pasted, and modified. Tools/utilities are often placed in separate directories (util). This is done within each project, but seldom between projects, because nobody trusts the other developer(s) to have done a good enough job making the utilities. This somewhat defensive attitude is due to the fact that each developer stands responsible for the code he writes and reuses, and so each developer is careful when reusing other people's code.

No reuse specific procedures are employed. Mr. Lünell doesn't have faith in one standard set of procedures, as the situation changes from project to project. No reuse specific tools are used. Mr. Lünell explained that he doesn't believe that such tools work in practice, as the developers often make things too special to be reused. However, if a developer has produced general components, they should be included in a JAR file which then might be included in other applications. One might create a database system around this, but Mr. Lünell feels that it is just as well to simply use Javadoc instead. Documentation is an important issue, and it counts for much to have *one* overview. Mr. Lünell mentioned that he expects that there is too little communication in large companies, which makes it more difficult to keep an overview.

It is important to remember to remove code as well, not only reusing it. People are afraid to throw code away, but often it isn't necessary to include absolutely everything. On the other hand, if the code has been running for a long time, it is stable, and changing the code might make the code less stable. Including only the useful parts vs. maintaining stability is a tradeoff point.

E.22 V — Auticon AS

At Auticon AS the interviewee was System architect (Sticos økonomi) Frode Nilsen. This interview lasted for 20 minutes.

E.22.1 About the Company

Auticon AS is a Norwegian company located in Trondheim. The company offers products for and competence in accounting and wages. Their main product is Sticos oppslag ("book of reference"), but they also develop and sell products for accounting, annual settlement, presentation of results and updates on e.g. changes in the laws governing taxes and wages. Auticon has 37 employees, including 5 software developers.

E.22.2 Transcription

Which programming languages are used varies, but the developers use C++ and Delphi as the main development languages. The developers should reuse whenever possible, but there are no particular policies regarding how reuse should be done. Mr. Nilsen explained that the developers have many libraries which are bought from third parties, as it's better to reuse already existing code than creating reusable code yourself. When the developers do create reusable code, they extract the functionality which is generally useful for the company products.

The desired effects of reuse are saved time, and uniformity. Uniformity greatly simplifies testing. The developers achieve these effects, but there are also downsides to reuse. Especially when libraries are bought, there is a lot of extra code; code which is not useful. This means that when an error is detected, there is a massive amount of code to search through to find the error. In this way, it's simpler to produce reusable code yourself, as this can be more minimalistic. A second problem with third party reusable code is that not every supplier is good at responding to error reports et cetera. They thereby create extra work when they release new versions of their libraries, without Auticon's fixes. Also, updates from the suppliers might cause problems if the updated component isn't backward compatible.

Reusable components often start out as single files in a project, before they are gathered in a library (both internally in the project and externally to a reuse project). No tools which are specifically made for reuse are employed. "Projects" are started for reusable components of several types. For example, there is a project for database functionality, a project for user interfaces

and some projects for specific products. These projects are stored on a common server with a version control system.

No specific procedures are employed in connection with code reuse, apart from the routine of going through third party components to see what has to be added to achieve the desired functionality.

Reuse is often performed at class level, but the reusable code is collected in libraries and categorized. The developers also produce pure functions, code which is not part of a class, e.g. conversion functions. One way of reusing code is to link only parts of the code in a class (e.g. a method and class variables). How reuse is performed varies. For some products the developers use static linking with their own and the third parties' libraries, thereby creating an minimal application file where only the used methods and class variables are linked in. For other products, they use dynamic linking to their complete library set. This creates a somewhat larger distribution, but has advantages when updates or bug fixes are released, as only the changed components have to be distributed.

E.23 W — SYSteam AS

The interview with Nicolay Lae, responsible for developent of Spektra (Utviklingsansvarlig Spektra) at SYSteam AS, lasted for nearly 20 minutes.

E.23.1 About the Company

SYSteam is an international company with divisions in Sweden, Norway, Denmark and Finland. In Norway, SYSteam has about 60 employees in Oslo, Stavanger and Askim. There are about 25 developers located in Norway. SYSteam is an IT partner for small and medium size companies, and they are specialists in Enterprise Resource Planning/Supply Chain Management (ERP/SCM), system development, technical platforms, and management services. SYSteams knowledge of the trade lies in trading, wholesale, and industry.

E.23.2 Transcription

The developers at SYSteam/Spektra[1] use C and C++ when developing software. Mr. Lae explained that the company has developed an ERP system which is sold to Norwegian and Swedish industry. He also told me that he has some code which he has been working on for a long time, and in this context, reuse of code is using methods/functions and adjusting them.

The developers have a foundation library which is used in a host of contexts. The contents of the library is used for a lot of things, and changed to suit different needs. This foundation routine library has existed for 10 years.

One of the most important effects of reuse is speed; Mr. Lae stated that not reusing is not an option, as the developers need to be efficient. Almost just as important is the safety of knowing that components work. Another reason for reuse is uniformity; when several developers work on the same problems, they need to solve them in the same way. The libraries establish a way to solve the problems, and a standard for how things are done. These desired effects are actually achieved.

Mr. Lae remarked that sometimes, it would be better to produce new code instead of reusing. This doesn't happen often, however, it is more usual that code is not reused when it should be. He feels the developers should reuse more often.

The central element of code reuse is as mentioned a common routine library, with accompanying documentation. There are no formalized reuse specific

[1]The contents of this interview are only valid for SYSteam/Spektra.

procedures for reuse (library/remaining code). Tools for text search and comparison (UNIX-like "grep" and "compare" commands) are used in connection with reuse and adjustments of this code, but no tools specifically produced for code reuse are employed.

E.24 X — Deriga AS

At Deriga AS I interviewed head of development, Torje Lundereng. The interview lasted for a quarter of an hour.

E.24.1 About the Company

Deriga AS is a Norwegian company, located in Trondheim. Deriga's goal is to cover the need for secure and reliable information exchange, primarily in the health sector. This is done by offering simple, user friendly and efficient IT tools. The company has 8 employees. 3 of them develop software.

E.24.2 Transcription

The programming language primarily used by the developers at Deriga is C++, but Java is also used occasionally. Mr. Lundereng told me that the development department has a policy on reuse; when code is written, it should be as general as possible, especially if the code might be useful for other projects.

The developers try to keep "cut and paste" to a minimum, and libraries are the most common form of reuse. Code which is not considered reusable is not put in libraries. When a piece of code is considered to be reusable, it could be put in an already existing library, or a new library with an appropriate name has to be created. The code is subject to version control, and is available to all the developers on a common server.

The desired effects of code reuse are saved time and not having to "re-invent the wheel". These effects are truly achieved; Mr. Lundereng stated that the libraries get extensive, and so reuse saves a lot of time, and it increases the stability of the software. The developers try to find existing libraries both internally and externally, but third party reusable libraries are not commonly used. This is due to the fact that not too many third party libraries for their application area exist.

No tools specifically produced for code reuse are employed. When it comes to procedures for reuse, the developers have a policy stating that if code is written which has the same functionality as already existing code, the previously developed version is to be used — possibly with some added functionality.

E.25 Y — NetIT AS

For the final interview I spoke to software developer Joar Holme at NetIT AS. The interview lasted circa 10 minutes.

E.25.1 About the Company

NetIT AS is a Norwegian company located in Oslo. The company delivers professional business systems and operational solutions for Small and Medium sized Enterprises (SME). NetIT has 9 employees, of which 2 develop software.

E.25.2 Transcription

The developers at NetIT recently went from C++ to C# .NET as their primary programming language. Previously, there has also been a lot of Web programming (JScript, VBScript) and a few projects using Visual Basic. Some of the developers' projects are similar, and in these cases it is natural to reuse the code.

The desired effects of code reuse are not having to rewrite code, increased development speed and more stable code. These effects are achieved, if not to the extent the developers wish. They hope the transition to C# might increase both the level of reuse and the positive effects of reuse. A downside to trying to reuse is the possibility of over-generalization; when the developers are eager to reuse, components might have been made too general or the components might not be reused often enough to be worth the effort.

The developers have built what might be called library files for reuse. Mr. Holme gave an example: Scripts for Web pages are separated into different files. Routines for different functionalities (such as database access) are stored in different files. This enables routines with similar functionalities to be reused later. All code is subject to version control. No tools or procedures specific to reuse are employed.

Bibliography

[1] Camilla Fledsberg. Prosessorientert kvalitetssystem i praksis. Master's thesis, NTNU, June 2003.

[2] Kristen Ringdal. *Enhet og mangfold.* Fagbokforlaget Vigmostad og Bjørke AS, 1 edition, 2001.

[3] Bruna Galobardes. Key steps in planning a survey. *Soz.- Präventivmed.*, 47:349–351, 2002.

[4] Fritz Scheuren. What is a survey. Published on the ASA's web page: http://www.amstat.org/sections/srms/pamphlet.pdf, June 2004.

[5] Kevin Boone. How to conduct a survey. Published on author's web page: http://www.kevinboone.com/howto_survey.html, July 2004.

[6] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in Software Engineering - An Introduction.* Kluwer Academic Publishers, 2000.

[7] Steinar Kvale. *Det kvalitative forskningsintervju.* Ad Notam Gyldendal, 1997.

[8] Tor-Erik Hauge. Gjenbruk i it-bedrifter — utvikling og trender. Master's thesis, Høgskolen i Stavanger, June 2003.

[9] Tom Davenport and Gilbert J. B. Probst. *Knowledge Management Case Book: Siemens best practises.* Wiley, John & Sons, Incorporated, 2002.