An Empirical Study of Developers Views on Software Reuse in Statoil ASA

Odd Petter N. Slyngstad, Anita Gupta, Reidar Conradi, Parastoo Mohagheghi
Dept. of Computer and Information Science (IDI) Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
+4773593440
{oslyngst, anitaash, conradi, parastoo} at idi.ntnu.no

Harald Rønneberg, Einar Landre
Statoil KTJ/IT
Forus, Stavanger, Norway
+4751990000
{haro, einla} at statoil.com

ABSTRACT
In this article, we describe the results from our survey in the IT-department of a large Oil and Gas company in Norway (Statoil ASA), in order to characterize developers' views on software reuse. We have used a survey followed by semi-structured interviews, investigating software reuse in relation to requirements (re)negotiation, value of component information repository, component understanding and quality attribute specifications. All 16 developers participated in the survey and filled in the questionnaire based on their experience and views on software reuse. Our study focuses on components built and reused in-house. The results show that reuse benefits from the developers view include lower costs, shorter development time, higher quality of the reusable components and a standardized architecture. Component information repositories can contribute to successful software reuse. However, we found no relation between reuse and increased rework. Component understanding was generally sufficient, but documentation could be improved. A key point here is dynamic and interactive documents. Finally, quality attribute specifications were trusted for the applications using reusable components in new development, but not for the reusable components themselves.

Categories and Subject Descriptors
D.2.13 [Software Engineering]: Reusable Software- *reusable libraries.*

General Terms: Measurement, Verification.

Keywords: Empirical Study, Software reuse, CBSE.

1. INTRODUCTION
Software reuse can be specified in two directions [14], namely *development for reuse and development with reuse*. The former refers to systematic generalization of software components for later reuse, while the latter deals with how existing components can be reused in existing and new applications and systems. However, when it comes to reusing in-house built components, these two processes are tightly related.

Currently, we are studying the reuse process in the IT-department of a large Norwegian Oil & Gas company named Statoil ASA[1] and collecting quantitative data on reused components. To improve our understanding and collect evidence from several sources, we also performed a survey followed by semi-structured interviews in the organization. The research interests are obtained from the extant literature, and include the major benefits and factors contributing towards reuse, the effect of reuse on rework, as well as understanding and trust of component and quality specifications. Based on these issues, we have defined and explored several research questions through a survey questionnaire.

The results support some conclusions from earlier studies, while contradicting others. The sample size is rather small, and further studies will be used to refine and further investigate the research questions presented here. This study can therefore be seen as a pre-study. This paper is structured as follows: Section 2 discusses software reuse and CBSE, Section 3 has related work, and Section 4 discusses research background and motivation. Furthermore, Section 5 contains the results of our survey, Section 6 discusses these results, while Section 7 concludes.

2. SOFTWARE REUSE AND CBSE
Software reuse can have varying degrees of application, ranging from case-by-case basis (ad-hoc) to fully systematic approaches [6]. The most-inclusive definition of the term encompasses reuse of any and all assets, that is, from design and code through established procedures to documentation and knowledge. Benefits include easier understanding of the functionality, a potential shorter time-to-market, as well as possibly less effort spent on maintenance and future adoption of new requirements [17].

However, over the past decade, several attempts have been made at improving software development practices by design techniques, developing more expressive notations for capturing a system's intended functionality, and encouraging reuse of pre-developed system pieces rather than building from scratch [2]. Already in 1972, Davis Parnas wrote about the advantages of decomposing a system into modules. He mentions benefits such as [12]:

ISESE'06, September 21-22, 2006, Rio de Janeiro, Brazil.

[1] ASA stands for "allmennaksjeselskap", meaning Incorporated.

- shorter time-to-market (development time) because modules can be developed by separate groups,
- increased product flexibility,
- ease of change, and finally
- increased comprehensibility as modules can be studied separately.

A new style of software development based on the principles of Parnas, and emphasizing component reuse, is CBSE; This involves the practices needed to perform component-based development in a repeatable way to build systems that have predictable properties [1]. *Component-Based Software Engineering* (CBSE) and *Component-Based Development* (CBD) are approaches to the old problem of handling the complexity of a system by decomposition, and these two concepts are often used indistinguishably [10]. Although, much effort has been devoted to define and describe the terms and concepts involved, there is some literature that distinguishes between these two concepts. According to Bass [1], CBD involves the technical steps for designing and implementing software components, assembling systems from pre-built software components, and deploying assembled systems into their target environment. CBSE, however involves the practices necessary to perform CBD in a repeatable way to build systems that have predictable properties [1]. An important goal of CBSE is that components provide services that can be integrated into larger, complete applications.

CBSE allows the reuse of common functionality between applications, as well as organization-wide distribution of best practices. This functionality is embodied in components, which provide services that can then be included in new development, hence reused. Research has long investigated the connections between reuse and CBSE in terms of experience accumulated by practitioners on issues related to software reuse. CBSE provides the means to flexibly upgrade or replace parts of a system in order to satisfy the increasing requirements for agility and speed in new development. Another key feature of CBSE is the focus on quality attributes and corresponding testing.

3. RELATED WORK

Lim [8] have conducted a study of the effect reuse have on quality, productivity and economics in Hewlett-Packard. Data was collected from two reuse programs in this company. The results of this study revealed that reuse can provide a substantial return on investment. HP reuse programs documented improved quality, increased productivity, shortened time-to-market, and enhanced economics resulting from reuse.

Frakes & Fox [4] conducted a survey in 1991-1992, where they answered 16 commonly asked questions about reuse. A total of 113 people from 28 U.S organizations and one European organization, with a median size of 25,000 employees, participated in this survey. Some of the results that the study revealed were that education influences reuse, developers actually prefer to reuse instead of building components from scratch, reuse is more common in telecommunications compared to aerospace, and that having a reuse repository does not improve software reuse. Additionally, they found that a common software process may be advantageous.

A large reuse project was the REBOOT (Reuse Based on Object-Oriented Techniques) project [14], where the focus was on the importance of the organizational aspects of reuse in addition to the traditional technical perspective. These issues include organization and processes, as well as business drivers and human factors. The availability of more experience with industrial reuse may show the importance of these non-technical factors to be at least equal to that of the technological aspects [5] [9].

Another interesting survey is one performed by Morisio, Ezran and Tully [11]. They analyzed 24 projects in both large and small companies in Europe performed in 1994-1997 involving reuse. Their results revealed that successful component reuse was achieved when the organizations had a potential for reuse because of commonality among applications, management committed to introducing reuse process, modifying non-reuse processes, and addressing human factors.

Although the aforementioned studies cannot be directly compared to our survey, they are important to see the general trends in software reuse, and provide some of the motivation for our research questions. We will also compare with these studies where applicable, as shown in section 6 where we discuss our results.

In [7], Li et al. investigated developer attitude towards reuse of in-house components, collecting data from 26 respondents in 3 companies. They found that the concerns were the same among those reusing components built in-house, as among those using Components-Off-The-Shelf (COTS), when it comes to (re)negotiation of requirements, documentation and the specification of quality attributes for components. Also, their results lend support to the claim that repositories do not contribute towards success in software reuse, and show that informal communication between developers can be very valuable, due to shortcomings in the component documentation. This is the work which is most applicable to our research in this survey, and a number of the questions in the questionnaire have been adapted to our use, as seen in section 4. Our contribution is hence to confirm or decline the results from the aforementioned studies, in addition to possibly reveal new results.

4. RESEARCH BACKGROUND AND MOTIVATION

Over the last decades a large push has been towards understanding the issues involved in reuse and discover the benefits and disadvantages of different approaches within the field. In CBSE, a key point of utilizing software reuse is to be able to manage software evolution through reusing components systematically, to take into account new requirements when they appear. Successful introduction and propagation of a software reuse program can be characterized by three overall points [11]:

- Commitment from management at all levels,
- Process modifications in the following manner:
  - Starting reuse processes,
  - Altering non-reuse processes,
  - Taking human factors into account, and
- Awareness of the organization's context.

When it comes to development *with* reuse, being able to match the requirements to existing reusable components is important. Also, being able to obtain sufficient knowledge of these components, as well as being able to reuse them with little or no modification, are paramount issues [9].

Our motivation is to reevaluate the issues surrounding software reuse from the perspective of developers involved in a reuse program. In particular, we want to explore the possible benefits, disadvantages and contributors towards successful reuse of software components. We also want to look at the documentation and quality specifications of reusable components that is available to the developers, who reuse them in new development. In the following section, our research questions for the survey are presented.

4.1 Research Questions

RQ1: What are the key benefits of reuse? The existing literature on software reuse claims that reuse has a positive effect on quality, productivity and time-to-market [8]. These benefits appear to be present from the second reuse occurrence; hence a positive return on investment can easily be seen over a relatively short period of time. The purpose in our case is to confirm whether these positive effects can be seen in the same way from the perspective of the developers.

RQ2: Which factors contribute to facilitate reuse? As aforementioned, key factors towards facilitating reuse in industry are management commitment, necessary process modifications, and organization context awareness [11]. These are factors on a higher level, which developers may not have a large amount of influence on, although they affect developers directly. It may therefore be interesting and beneficial to investigate developer's opinions on this question, while still extracting extra qualitative information.

RQ3: Does reuse increase rework? Statoil ASA has it as a goal to keep rework as low as possible. Rework in the company is concerned with fixing problems (due to changes in requirements or misunderstood/ambiguous requirements), and may be more for reused components if extra effort is needed to analyze and fix such problems for components developed earlier or by other teams. It is therefore important for them to be aware of the causes and possible remedies surrounding this issue.

RQ4: Do developers have sufficient information to understand the relevant components? If the answer is no, how can they solve this problem? Component information for developers should at least encompass requirements and functional specifications. In addition, lower level details such as use cases, tests and the like can be valuable, but depending on the individual area of responsibility the needs may be different between developers. A key issue noted in literature is the need for the developers to have enough relevant information available. A noted problem within this issue is the inability to express quality attribute information on a per component basis [3].

RQ5: Do developers trust the relevant quality specification of the components? If the answer is no, how can they solve this problem? Trust is paramount in CBSE in the sense of developing trustable systems from components for which the developers may only have partial information. While CBSE allows the construction of systems from individual components, there is only a low focus on integration and quality attribute issues [18]. Here, we want to check the current status, and obtain the developers opinions about what can be done to remedy the situation, if problems exist.

The questionnaire used in this study is extended and adapted from that of an earlier study, also on reuse [7]. Some of the questions and sections have been modified and added due to different research questions.

4.2 The Questionnaire

Based on our aforementioned research questions we have decided to use a quantitative survey, supplemented with a semi-structured interview.

Table 1. The questions in our questionnaire

General on software reuse[2]	**Comments**
Q6. What is your highest level of education? Q7. How many years of experience do you have with ***software development*** after completed education? Q8. How many years of experience do you have with ***reusing components*** after completed education? Q9. How important do you consider software reuse for achieving the following benefits? (5 point scale; answer alternatives: Lower development costs, Shorter development time, Higher JEF[3] component quality, A more standardized architecture, Lower maintenance costs (including technology updates), Increased knowledge/knowledge sharing, Other (please specify)) Q10. What software artifacts are most important to be reused? (Requirements, Use Cases, Design, Code, Test data/documentation ranked 1 to 5) Q11. Does Statoil ASA have a training program about software reuse? Q12. If "Yes" in Q11, have you attended this training program? Q13. Do you use a formal software reuse process for ***developing*** JEF components? Q13b. If "No" in Q13, would the availability of such a formal software reuse process be beneficial for you? Q14. Do you use a formal software reuse process for ***reusing*** JEF components? Q14b. If "No" in Q14, would the availability of such a formal software reuse process be beneficial for you?	All questions on general software reuse are customized specifically for Statoil ASA.
Requirements (re)negotiation	
Q15. Are requirements changed/(re)negotiated due the development process (DCF and S&A[4])? (5 point scale) Q16. Are requirements misunderstood / ambiguous? (5 point scale) Q16b. If "Very often" or "Often" in Q16, what are the consequences (e.g. excessive effort)?	Q15, Q17, and Q18 are adapted from [7], while the remainders

[2] Questions Q1-Q5 deal with the general respondent information, and the results from these questions are used towards characterizing the respondents.

[3] JEF is the name used to refer collectively to the reusable components in use at Statoil ASA.

[4] DCF and S&A are two development projects at Statoil ASA, which utilize reuse in new development.

Q17. Are requirements flexible (by flexible we mean that requirements are easy to change, modify, etc.) in the development projects? (5 point scale) Q18. The requirements (re)negotiation processes related to the JEF components work efficiently in the projects (DCF and S&A)? (5 point scale) Q19. Rework (by rework we mean extra effort to fix problems due to changes in requirements or misunderstood/ambiguous requirements) has increased after introducing JEF components? (5 point scale)	are customized specifically for Statoil ASA.
Value of component information repository	
Q20. Availability of a JEF repository (e.g. for storing information about JEF components) would be beneficial for me? (5 point scale) Q20b. If "Agree" or "Strongly agree" in Q20, please specify why: Q20c. If "Strongly disagree" or "Disagree" in Q20, please specify why:	Q20 is adapted from [7].
Component understanding	
Q21. Which of the following JEF components do you find the most difficult to develop and / or reuse? (The JEF components are listed, respondent is asked to rank them for both cases in terms of difficulty) Q22. How well do you know the SJEF[5] architecture? Q23. How well do you know the interface of the components? (5 point scale) Q24. How well is the design / code of the reusable components documented? (5 point scale) Q24a. If the answer of Q24 is "Poorly" or "Very poorly", is this a problem (please specify why)? Q24b. If the answer of Q24 is "Poorly" or "Very poorly", what are the problems with the documentation? Q24c. If the answer of Q24 is "Poorly" or "Very poorly", how would you prefer the documentation? Q25. What is your main source of documentation about JEF components during implementation (e.g. documentation, ask the JEF team)? Q26. Please specify if there are any other problems with understanding JEF components Q27. How do you usually reuse a JEF component? (alternatives: as is, with modifications, with modifications performed by the JEF team, Other (please specify), No relevance)	Q22, Q23, Q24, Q25, Q27 are adapted from [7], while the remainder are customized specifically for Statoil ASA.
Specification of components quality attributes (non-functional requirements)	
Q28. How are the specifications for JEF components' quality attributes defined? (5 point scale) Q29. How are the specifications for the (developed) system quality attributes defined? Q29b. If "Poorly" or "Very poorly" in Q28 and / or Q29, what can be done to improve the situation? Q30. Are the components tested for their quality attributes before integrating them with other components?	Q28, Q29, Q30 are adapted from [7].

Table 2. Research questions vs. questionnaire questions

Questions	RQ1	RQ2	RQ3	RQ4	RQ5
Q9-Q10	X				
Q6-Q8, Q11-Q14, and Q20.		X			
Q15-Q19			X		
Q21-Q27				X	
Q28-Q30					X

Questions Q1-Q5 deal with general respondent information, and the results from these questions are used towards characterizing the respondents.

Our questionnaire consists mainly of check boxes, but gives also the individual respondent to contribute their own qualitative input. From the respondents own personal qualitative data we hope to obtain information, which can be supplemented with the rest of the data material. Due to our research questions, we think that a standardized survey with semi-structured interviews seems most appropriate for our data collection. This is because this research method gives us the opportunity to obtain satisfactory amount of information from each respondent with the help of a structured survey. A quantitative survey also gives us the possibility to sort out the collected data in a least time-consuming way. It gives us the opportunity to analyze data with statistical tools and analysis techniques.

Once the questionnaire was formed, it was first pre-tested on two separate occasions among 6 academic colleagues to obtain comments and to ensure that we were asking the questions understandably and would obtain the desired information. The final questionnaire is 11 pages long and contains 30 questions, which are grouped in five parts. These parts are General on software reuse, Requirements (re)negotiation, Value of component information repository, Component understanding and Specification of components quality attributes (non-functional requirements). Each question in the questionnaire has been used to study one of the research questions. Table 1 describes the questions in more detail, and the correspondence between research questions and the questions in the questionnaire is in Table 2.

4.3 The Context

Statoil ASA is a major oil and gas operator on the Norwegian continental shelf. They are headquartered in Europe, present in 28 countries, and have 24 000 employees worldwide. Within the company, the central IT-department is responsible for developing and delivering software which is meant to give key business areas better flexibility in their operation. This department consists of approximately 100 developers worldwide, located mainly in Norway and Sweden. The 16 developers we selected are located in Norway, specifically in Stavanger, Trondheim and Oslo. Since

[5] SJEF is the name given to the architecture which the reusable components are built on.

2004, a central IT strategy of the O&S (Oil Sales, Trading and Supply) business area has been to explore the potential benefits of reusing software systematically, in the form of a framework based on Java Enterprise Framework components. The actual JEF framework (Java Enterprise Framework) consists of seven separate components (these are: JEF Client, JEF Workbench, JEF Util, JEF Dataaccess, JEF SessionManagement, JEF Security and JEF Integration), which can be applied separately or together when developing applications. This strategy is now being propagated to other divisions within Statoil ASA. Reuse in Statoil ASA is component-based, with a foundation in an in-house developed architecture and with a related component framework based on Java Enterprise Framework technology.

We are currently studying two projects at Statoil ASA, namely **DCF (Digital Cargo Files)** and **S&A (Shipment & Allocation)**. The **DCF** application is mainly a document storage application. It imposes a certain structure to the documents stored in the application, and is based on the assumption that the core part of the documents is based on cargo (load) and deal (contract agreement) data as well as auxiliary documents pertaining to these information entities. DCF is meant to replace the current practice of cargo files, which are physical folders containing printouts of documents pertaining to a particular cargo or deal. A "cargo file" is a container for working documents related to a deal or cargo, within operational processes, used by all parties in the O&S strategy plan at Statoil ASA. The DCF application consists of 21459 LOC, and has a current used budget to date of 15.7 million Norwegian Kroner (about 2 million Euros). The **S&A** application aims to allow operators to carry out risk analysis on shipments from loading at terminals and offshore, as the current application is not able to take care of complex agreements (i.e. mixing of oil qualities within the same shipment). The S&A application consists of 64319 LOC, and has a current used budget to date of 17 million Norwegian Kroner (about 2.12 million Euros).

4.4 Data Collection

Data collection was carried out by two NTNU PhD students, the first and second author of this paper. We selected Statoil ASA, since they are cooperating with us in our SEVO (Software EVOlution) project and throughout our PhD research. The respondents are developers in Statoil ASA. This survey is, therefore, a non-probability sampling, based on convenience as described in Section 4.5. The developers that participated in the survey currently work with the DCF and S&A projects, reusing the JEF components developed by the JEF Team. Also, some of these developers are part of the JEF Team, that is, they both develop and reuse the JEF components. The survey was distributed among the developers, who were then allowed a few hours within which to complete it. We had contacted and agreed upon the date with the relevant department and project managers beforehand, to ensure that enough time was allotted for this purpose. The developers answered the questionnaires separately, and they were filled out by hand. Filling out the questionnaire took 12-14 minutes, as estimated from the test runs. None of the actual respondents used more time than the allotted time to finish answering the questions. After the developers had completed the questionnaire we performed short semi-structured, one-on-one interviews with each of the developers for 10-15 minutes. This was done for providing support with possible misunderstandings in answering the questionnaire, as well as obtaining more thoroughly qualitative information around the issues presented in the questionnaire.

4.5 Respondents

All respondents in our survey are developers that are involved in the DCF and S&A projects, and the JEF team. They all belong to the central IT-department at Statoil ASA which utilizes development *for/with* reuse in their own development projects. The software is developed mainly for other units within Statoil ASA as customers, and aims to be at the forefront of what technology can offer. In total, there are 16 developers working with the DCF project, the S&A project and the JEF Team at Statoil ASA in Stavanger, Trondheim and Oslo. We asked all these developers to participate in the survey, and got 16 filled-out questionnaires back. These 16 developers were selected, since their work is related to JEF component reuse. There are a total of 100 developers in the IT-department, as aforementioned. However, these 16 developers are the only one's currently specifically involved with software reuse at Statoil ASA, and the remainders would therefore be less relevant for us in this survey.

All of the respondents have an IT background and education, seven of them have a Master of Science degree, while the other nine have education on the Bachelor degree level. A total of 22 roles were identified; 14 had a role as developer, 4 had a role as designer, 2 had a role as an architect and 1 had a role as a test manager. In addition, there was 1 respondent who filled the responsibility roles of maintenance and support. Therefore, several of the respondents had multiple roles within and also between the projects / teams. Seven of them had been working in software development between five and ten years, while the majority of the remaining respondents had more than ten years of experience. Only three respondents had less than five years overall experience. The majority expressed having less than ten years of experience in working with reuse.

5. PRESENTATION OF RESULTS

In this section, we summarize the survey results. All the statistical data presented in this study are based on valid answers, and *no relevance* answers are not included in the analysis. The statistical analysis tool we used is SPSS version 1.0 and Microsoft Excel 2003.

5.1 RQ1: What are the Key Benefits of Reuse?

First, we wanted some general information about software reuse in Statoil ASA, and questions Q6-Q14 were asked to get this information. However, Q9 and Q10 were asked to provide answer to RQ1 and are based on developers' subjective opinion related to this issue. The result of Q9 is shown in Figure 1, and the result of Q10 is shown in Figure 2. These figures are boxplots, showing the upper and lower 25% and 75 % quartiles, as well as the median, and outliers where applicable [15][16].

The numbering along the vertical axis in Figure 1 is the individual ranking; where 2=Low, 3=Medium, 4=High and 5=Very high. None of the respondents gave *very low* to the benefits. The abbreviations cq, ik, ldc, lmc, sa and sdt along the horizontal axis in Figure 1 are subsequently higher JEF component quality, increased knowledge/knowledge sharing, lower development cost, lower maintenance cost (including technology updates), a more

standardized architecture and shorter development time. From Figure 1, we can see that most developers think that the component quality, lower development costs, a more standardized architecture, and shorter development time are seen as equally important benefits of software reuse, while increased knowledge/knowledge sharing is less important.

The numbering along the vertical axis in Figure 2 is the priority given by each developer; where 1=most important and 5=least important. The abbreviations co, ds, rq, te, uc along the horizontal axis in this figure are subsequently code, design, requirements, test data/documentation and use cases. From Figure 2 below, we can see that most developers think that design/code is the most important to be reused, while requirements and use cases are less important to be reused.

In summary, most of the developers think that lower development cost, shorter development time, higher JEF component quality, and a more standardized architecture are the most important benefits of reuse, while the artefacts that are important to be reused are design/code in order to achieve the benefits.

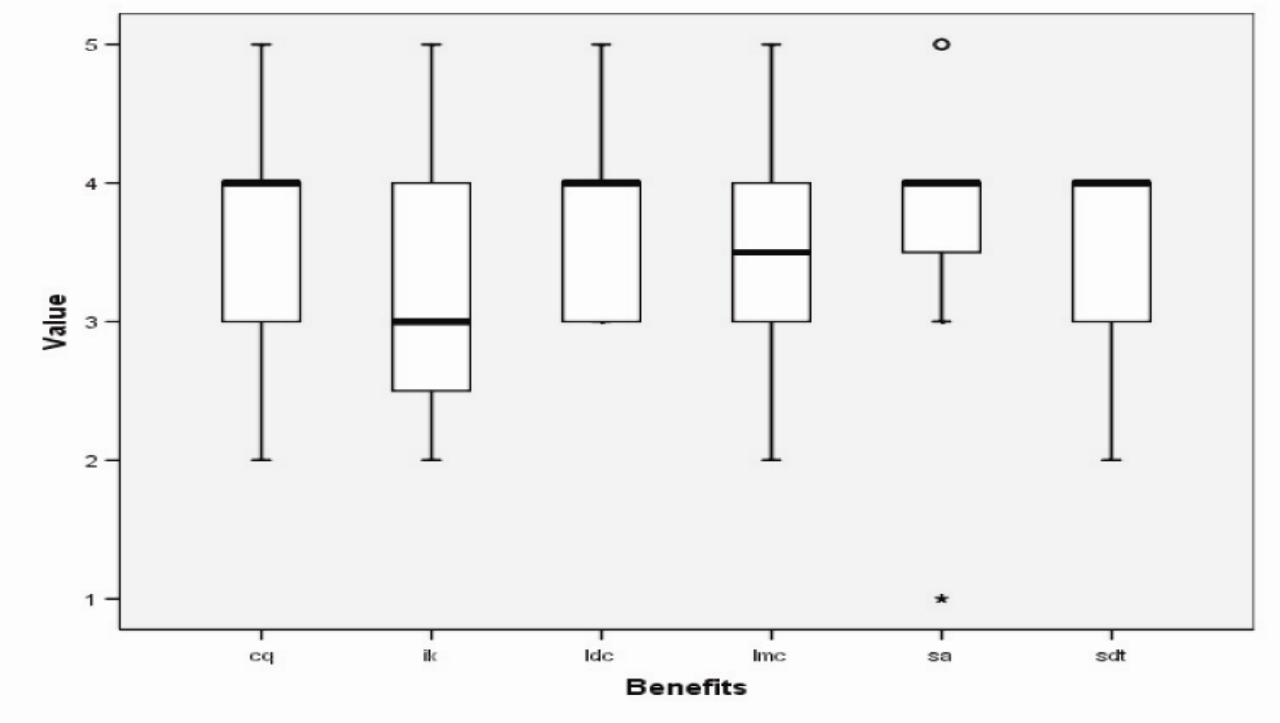

Figure 1. Benefits of software reuse

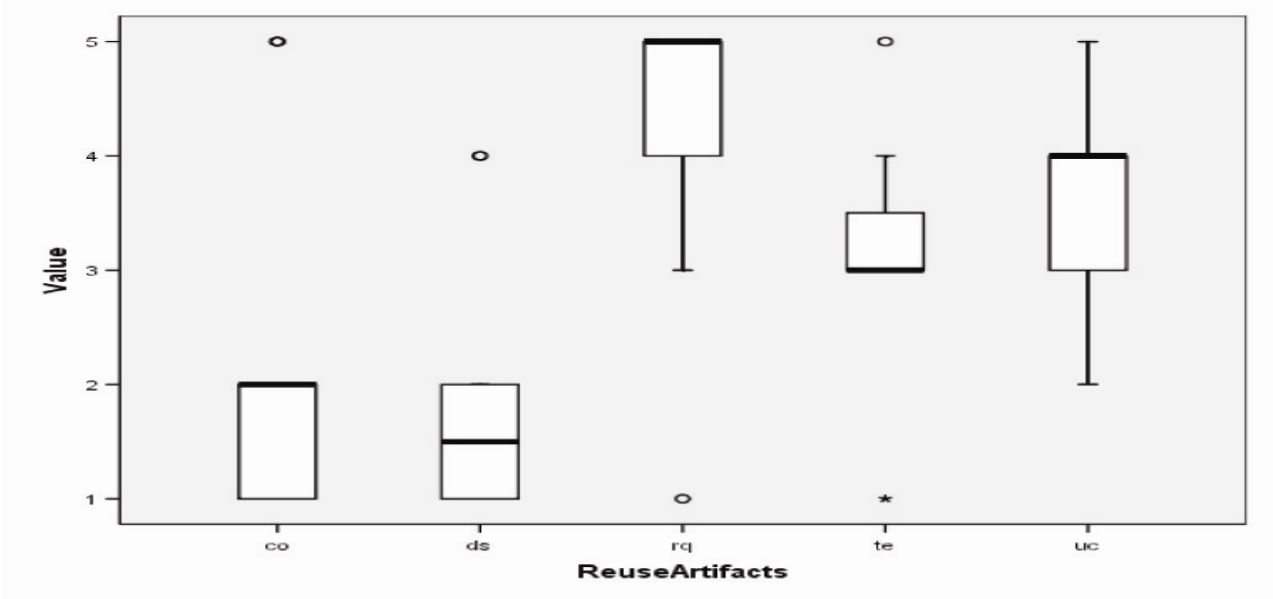

Figure 2. Software artefacts important to be reused

5.2 RQ2: Which Factors Contribute to Facilitate Reuse?

The questions used to evaluate this research question were Q6-Q8, Q11-Q14 and Q20. The results from Q6-Q8 (level of education and experience) are briefly presented in Section 4.5 above. From this, we can see that mean experience with software development is 8.6 years, while mean experience with software reuse is 6.9 years. Q11-Q14 revealed that 12/16 of the developers *don't know* whether Statoil ASA has a training program on software reuse. We know from communication with upper level management that Statoil ASA does not have a formal process for developing and/or reusing reusable components. Nevertheless, the questionnaire revealed some disagreement between developers on this issue, as 3/16 answered *Yes* and 5/16 answered *No* to question Q13. Likewise, in Q14, 5/16 answered *Yes* while 6/16 answered *No*.

It has already been shown thoroughly in other studies that a component repository for storing the reusable components themselves is not a contributor for reuse. We therefore decided to explore the value of a repository of information about the components. From Q20, we see that the vast majority of the respondents think that such a repository would be beneficial for them; none of them disagree, and only two said they were neutral on this issue. This may be partly due to the poor documentation of reusable components as discussed in RQ4. It should be noted that the main function of a traditional reuse repository is search and retrieval of reusable components, which is not relevant here.

The factors that contribute towards facilitating reuse can hence be summarized as follows. With education, the slight majority have a bachelor's degree, while the rest have a higher education, but we have seen no evidence that this contributes towards reuse. The same is true for experience (with software development as well as with software reuse) – here too, we have seen no indication that experience promotes reuse. Also, the majority of the developers have no knowledge of possible training programs at Statoil ASA on software reuse, and there is confusion surrounding the issue of whether formal process(es) for developing and/or reusing software is actually in use. Finally, a repository for *information about the reusable components* would be advantageous.

5.3 RQ3: Does Reuse Increase Rework?

Questions Q15-Q19 were used to investigate whether reuse leads to increased rework levels. We found that for Q15, 11/16 developers feel that the requirements are changed/(re)negotiated *somewhat* in the development projects (DCF and S&A). Further, regarding Q16, 5/16 think that the requirements are *often or very often* misunderstood/ambiguous, and the same amount think that this is *seldom* the case, while another 6/16 think is *somewhat* true. 10/16 of the developers think that requirements are *somewhat* flexible in Q17 (while the remainders said this was only *seldom* the case), and 7/16 *agree* that the process of (re)negotiation of requirements towards the reusable components works efficiently for Q18 (Here, another two developers *disagree*, while the rest are *neutral)*. A plausible reason for this may be that requirements naturally change often in typical development projects.

Finally, when asked directly in Q19, 3/16 think that the introduction of reuse has not caused increasing rework, while 3/16 are of the opposite opinion, and the remaining are neutral. In summary, we have not found any significant evidence that reuse leads to an increase in rework, hence our results remain inconclusive.

5.4 RQ4: Do Developers Have Sufficient Information to Understand the Relevant Components? If the Answer is No, How can they Solve this Problem?

Questions Q21-Q27 were used to investigate this research question. From Q21, the most difficult components appear to be JEF Client and JEF Integration, while the easier one's are JEF Util and JEF Dataaccess.

Q22 revealed that 8/16 know the architecture *well or very well*, while 7/16 know it *somewhat*, and only 1/16 replied knowing it *poorly*. Q23 shows that 5/16 know the component interfaces *well or very well*, while 8/16 know it *somewhat*, and only 2/16 answered that they know it *poorly.*

In Q24, 2/16 think that the design/code is *well* documented, while 9/16 wrote *somewhat* and, again, 3/16 wrote *poorly.* In Q25, developers answered that their main sources of information on the JEF components are typically *the JEF team, javadoc/source code, colleagues and the JEF homepage*. Furthermore, in Q26, when asked about other problems with component understanding, the answers were that *the reusable components are immature/unstable as they are changing, and that documentation is insufficient, as well as that it is unclear how different components cooperate and the dependencies between them*. Finally, Q27 shows that 8/16 of the developers reuse the JEF components *"as-is"*, while 3/16 *reuse with modifications by the JEF Team*. Only 2/16 replied that they *"reuse with modifications"* (performed by themselves).

The most difficult components are JEF Client and JEF Integration, and about half of the developers have a good understanding of the architecture. However, 50% of the developers only know the component interfaces somewhat, and 56% think that the design/code is somewhat well documented. Another 50% reuse the JEF components *"as-is"*.

It hence appears that while the majority of the developers have knowledge of the component architecture as well as the component interfaces, they're still unhappy with the documentation that is available. Currently, the qualitative answers from the questionnaire and the semi-structured interviews reveal that developers would like a website with overview, tutorial, sample code and good javadoc of the component code, which is as interactive as possible for the developers.

5.5 RQ5: Do Developers Trust the Relevant Quality Specification of the Component? If the Answer is No, How can they Solve this Problem?

In this research question, we used Q28-Q30 to elicit the answers. Q28 reveals that while 8/16 developers think that the quality attributes for the JEF components are *poorly or very poorly* defined, 3/16 think that they are *well or very well* defined. However, Q29 shows that when it comes to the projects DCF and S&A, 12/16 think that the respective quality attributes are *well or very well* defined, while only 2/16 think they're *poorly or very poorly* defined. Lastly, in Q30, 6/16 developers *do not know* whether the components are tested for fulfilment of their quality attributes before integrating them with other components, while 5/16 think that this is only *sometimes* done. 4/16 also think that this is *always* done, while 1/16 think that this is *not done at all*.

In summary, 75% of the developers say that the quality attributes (non-functional requirements) for the development projects (DCF and S&A) are well defined, while 50% think that these quality attributes are not well defined for the reusable JEF components. Hence, the developers trust the quality specifications for the development projects, but not for the reusable JEF components. In order to remedy this problem, the qualitative answers from the questionnaire and the semi-structured interviews show that the specification and publication of quality attributes for the reusable components should be improved in terms of realism and clarity. Additionally, more consistent component testing was also suggested as a way to handle this problem.

6. DISCUSSION OF THE RESULTS

We now discuss our research questions based on the results from our survey, as well as the inherent limitations and validity threats.

6.1 RQ1: What are the Key Benefits of Reuse?

Lim [8] showed that key benefits of reuse can be seen in terms of higher quality, higher productivity, and shorter time-to-market as well as economic benefits. Our results confirm that in the view of the developers, lower development costs (economic benefit), shorter development time (productivity – hence shorter time-to-market), higher JEF component quality (quality), are perceived as the key benefits of reuse. Additionally, a standardized architecture is also seen as a benefit.

6.2 RQ2: Which Factors Contribute to Facilitate Reuse?

Frakes & Fox [4] asked about whether reuse education both in academia and in industry, influences reuse. They found that though reuse education in academia and industry helps towards reuse, it is still uncommon in academia, as well as in industry. Our results show that many of the developers do not know about the existence of a reuse training program, so Statoil ASA must become better at promoting such training programs where they exist.

They [4] also wrote that although their respondents say that a common software process does not promote reuse, it may nevertheless contribute indirectly. Our results show that though Statoil ASA does not have a formal process specifically for developing/reusing reusable components, they do have one for general software development, which can implicitly affect reuse positively. Here too they must also become better at informing their developers about this.

When it comes to a repository, literature has concluded that a reuse repository does not increase levels of code reuse [4] [11]. We investigated the question of whether the availability of an JEF repository (e.g. for storing information regarding JEF components, rather than the components themselves) would be beneficial. On this issue, our results show an overwhelming agreement from the developers. The qualitative reasons given include easier information sharing, easier learning, improved level of documentation, better overview of the documentation and functionality, as well as of typical existing problems and troubleshooting.

We would like to emphasize again here that we have investigated the issue of documentation through an *information* repository, not for "finding" reusable components.

6.3 RQ3: Does Reuse Increase Rework?

The theoretical foundation behind this research question is that because extra effort may be needed towards analyzing and fixing problems with reusable components, reuse could potentially increase rework (as discussed in section 4.1). The analysis here is inconclusive, as we cannot show any link between reuse and increased rework. Possible reasons for this are as follows. The development projects DCF and S&A are meant to reuse the JEF components developed by the JEF team, however, our results show that developers often have multiple responsibility roles that often cross project and team lines. This means that there is no clear division between development *for/with* reuse in Statoil ASA. Reused components are developed internally and the organizational flexibility improves knowledge and compensates for the lack of a specific reuse process.

As aforementioned in Section 5.3, we have not seen any indications that reuse leads to an increase in rework, and our results here are therefore inconclusive.

6.4 RQ4: Do Developers Have Sufficient Information to Understand the Relevant Components? If the Answer is No, How can they Solve this Problem?

Li [7] found that developers understood the components well, despite a lack of related documentation, and that the knowledge was instead gotten through prior experience and local experts. Our results support these findings, in that the majority of the developers have sufficient understanding about the relevant components. They too think that the documentation could be better (see section 5.4), and use the JEF team or previous experience to achieve the necessary component understanding.

6.5 RQ5: Do Developers Trust the Relevant Quality Specification of the Component? If the Answer is No, How can they Solve this Problem?

Here, literature reports that most developers are unhappy with the quality specification of the components [7], and therefore cannot use this information. Our results, however, show that the relevant quality specification for the development projects DCF and S&A are well-defined, while that for the JEF components are not. This could be caused by more rapidly changing requirements, resources, personnel involved in the JEF team, and poor documentation. It may also be difficult to define quality specifications on the component level.

6.6 Threats to Validity

We here discuss the possible threats to validity in our survey, using the definitions provided by Wohlin [19]:

Construct Validity: Most of the research questions and the actual questions in the questionnaire have their origin from the research literature. From these, 12 of the survey questions were adapted towards our survey. Further, through pre-testing among local colleagues, most of the questions were refined additionally. Also, terms that may be unfamiliar to the respondents have been defined in the questionnaire handout.

External Validity: Another threat is that our survey is done completely by convenience sampling. That is, we chose this group of developers specifically because they are working with software reuse in the two projects DCF and S&A, as well as the JEF Team, which we are already involved in studying. It should be noted that the company's IT-department has a total of 100 developers, and that we have only sought answers from 16 of these. Nevertheless, these 16 are all the developers that are currently involved with software reuse at Statoil ASA. Also, the applications (DCF and S&A), with JEF development included, are representative of typical applications developed in-house at Statoil ASA in terms of size and allocated resources. Nevertheless, our limited sample size should be kept in mind. This means at least that we cannot generalize outside the context.

Internal Validity: The respondents were asked to answer the questionnaire by their project leader, and a contact relationship with them as well as with upper level management already existed at the time the questionnaire was carried out. The company itself has an expressed interest in gaining knowledge from the answers to the survey. We therefore are of the opinion that the respondents have answered truthfully to the best of their ability. In addition, we also provided support for possible ambiguities of the questions in the questionnaire.

Conclusion Validity: This analysis is performed based on an initial collection of data. Though too small a sample to be statistically significant, it still yields interesting and valuable insights for us and for Statoil ASA.

7. CONCLUSION AND FUTURE WORK

We have investigated the opinions of developers on software reuse, related to the five main areas: *benefits of reuse, factors contributing towards reuse, possible relations between reuse and increased rework, component understanding and quality attribute specification.* Overall, our results can be summarized as follows:

- When it comes to the *benefits of reuse*, the results of RQ1 show that the benefits of reuse can be seen in terms of lower costs, shorter development time, higher quality of the JEF components and a standardized architecture. These results support those found in literature [8].
- In terms of *factors contributing towards reuse* (RQ2), we found no link to education. We also found no evidence that experience contributes towards reuse. When it comes to formal processes, our findings support the literature [4] in that though the formal process in use is only for software development in general (not specifically for software reuse), this may still have an implicit positive effect. The results also show improving documentation of the reusable components would have been largely beneficial towards achieving successful reuse.
- On RQ3, we found no relation between *reuse and increased rework,* hence we cannot come to a conclusion. This is possibly caused by the mandate of reuse in the company, along with the multiple responsibility roles that often cross project and team lines, and that there hence is no clear division between development *for* reuse and development *with* reuse in the company.
- The results of RQ4 showed most developers have sufficient *understanding of the components*, but the documentation could be improved, as they largely use the JEF team or prior experience to achieve the required component understanding. A key point here is dynamic and interactive documents.
- *Quality attribute specification* (RQ5) was shown to be trusted for the development projects developing with reuse, but insufficient for the reusable components. This may be caused, in our case, by the rapid changes in the team that develop the reusable components, in terms of requirements, resources and personnel.

Our investigation is dependent on the subjective opinions of the developers as respondents. The results are presented to Statoil ASA and contribute to improving their processes. One interesting question raised from their side is whether the results of this work can be used as input to future larger reuse programs. The results will be combined with other research in the company to explain findings regarding reuse. We also plan to expand our dataset with more respondents, to refine the research questions based on our initial findings, and to compare our survey results with the actual results of the company.

8. ACKNOWLEDGMENTS

This work has been done as a part of the SEVO project (Software EVOlution in component-based software engineering), and ongoing Norwegian R&D project from 2004-2008 [13], and as a part of the first and second authors' PhD study. We would like to thank Statoil ASA for the opportunity to collect data from their reuse projects. We also thank our local colleagues for feedback and all of the respondents.

9. REFERENCES

[1] Bass, L., Buhman, C., Comella-Dorda, S., Long, F., Robert, J., Seacord, R., Wallnau, K. *Volume I: Market Assessment of Component-based Software Engineering* in SEI Technical

Report number CMU/SEI-2001-TN-007, 2001, (http://www.sei.cmu.edu/)

[2] Brown, A. W., Wallnau, K.C. The Current State of CBSE. *IEEE Software, 15,* 5 (Sept/Oct 1998), 37-46.

[3] Crnkovic, I. Component-based Software Engineering – New Challenges in Software Development. In *Proc. 25th Int'l Conference on Information Technology Interfaces* (Cavtat, Croatia, June 16-19, 2003). IEEE Press, 2003, 9-18.

[4] Frakes, W. B. and Fox, C. J. 16 Questions on Software Reuse. *CACM, 38,* 6 (June 1995), 75-87.

[5] Kim, Y., Stohr, E. A. Software Reuse: Survey and Research Directions. *Journal of Management Information Systems, 14*, 4 (Spring 1998), 113-147.

[6] Kruger, C. Software Reuse. *ACM Computing Surveys, 24*, 2 (June 1992), 131-138.

[7] Li, J., Conradi, R., Mohagheghi, P., Sæhle, O. A., Wang, Ø, Naalsund, E., Walseth, O. A. A Study of Developer Attitude to Component Reuse inside IT industries. *In F. Bomarius and H. Iida (Eds.): Proc. 5th Int'l Conf. on Product Focused Software Process Improvement (PROFES'2004)* (Kansai Science City, Japan, April 5-8, 2004). Springer Verlag LNCS 3009, 2004, 538-552.

[8] Lim, W. C. Effect of Reuse on Quality, Productivity and Economics. *IEEE Software, 11*, 5 (Sept./Oct. 1994), 23-30.

[9] Mili, H., Mili, F., Mili, A. Reusing Software: Issues and Research Directions. *IEEE Transactions on Software Engineering, 21*, 6 (June 1995), 528-561.

[10] Mohagheghi, P. *The Impact of Software Reuse and Incremental Development on the Quality of Large Systems*. PhD Thesis, NTNU, Trondheim, Norway, 2004.

[11] Morisio, M., Ezran, M., Tully, C. Success and Failure Factors in Software Reuse. *IEEE Transactions on Software Engineering, 28*, 4 (April 2002), 340-357.

[12] Parnas, D. L. On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM, 15*, 12 (December 1972), 1053-1058.

[13] The Software EVOlution (SEVO) Project, 2004-2008, http://www.idi.ntnu.no/grupper/su/sevo/

[14] Sindre, G., Conradi, R., and Karlsson, E. The REBOOT Approach to Software Reuse. *Journal of System Software, 30*, 3 (September 1995), 201–212.

[15] Stevens, S. *Psycholoical Scaling: Theory and Applications*. Wiley, 1951.

[16] Tukey, J. W. *The Collected Works of John W. tukey, Vol. III: Philosoply and Principles of Data Analysis: 1949-1964*. Wadsworth & Brooks/Cole Advanced Books & Software, 1986.

[17] Vliet, H. V. *Software Enginering, Principles and practice,* Wiley, 2nd edition, 2001.

[18] Councill, B., Heineman, G. T. Component-Based Software Engineering and the Issue of Trust. In *Proceedings of the 22nd International Conference on Software Engineering* (Limeric, Ireland, June 4-11, 2000). ACM Press, 2000, 21-31.

[19] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A. *Experimentation in Software Engineering – An Introduction.* Kluwer Academic Publishers, 2002.