Preliminary results from an investigation of software evolution in industry

Odd Petter N. Slyngstad, Anita Gupta, Reidar Conradi, Parastoo Mohagheghi, Thea C. Steen, Mari T. Haug
Department of Computer and Information Science (IDI)
Norwegian University of Science and Technology (NTNU)
{oslyngst, anitaash, conradi, parastoo} at idi.ntnu.no
Harald Rønneberg, Einar Landre
Statoil KTJ/IT
Forus, Stavanger
{haro, einla} at statoil.com

Abstract

In the SEVO (Software EVOlution) project, we explore the field of software evolution in terms of software quality attributes, their characteristics and possible relations between them. Currently, we have explored preliminary data from a software engineering program in a Norwegian company (Statoil ASA), on the frequency of defects and changes of reused components. These measures are the stated quality metrics of the program, and our results indicate that while defect-density evolves decreasingly over time, change-density does not exhibit a conclusive behavior. This is part of on-going research, and the results will be expanded and verified in future following publications. Overall, we aim to use the collected data towards discovering and explaining characteristics related to software evolution.

Keywords: CBSE, software evolution, quality attributes

1. Introduction

The purpose of the SEVO (Software EVOlution) project [SEVO, 2004] is to explore software evolution in Component-Based Software Engineering (CBSE) through empirical research. Aiming to increase our knowledge and understanding of underlying issues and challenges in software evolution, one long-term purpose of the project is to provide possible solutions to these problems. Another goal is to help industrial software engineers to improve their efficiency and cost-effectiveness in developing software based on reusable components, as well as in their ability to develop and use reusable assets. Underlying all this is the need for evidence to support or reject existing and proposed hypotheses, models, design decisions, and the like. Such evidence is best obtained through performing empirical studies in the field, and experience from such studies will be possible to incorporate into a knowledge base for use by the community.

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. The research questions are obtained from the existing literature, and include how the defect-density in reusable components evolves over time, as well as how the number of changes per reusable component evolves over time. Based on these issues, we have defined and explored several research questions and hypotheses through an empirical study. Here, we perform a preliminary analysis of data on defect-density and change-density of reusable components from a software engineering program in Statoil ASA, a major international petroleum company. We have chosen these two attributes for measuring software quality as they are part of the stated quality focus for the program in Statoil ASA. The purpose of this study is to gain initial understanding of software evolution from the viewpoint of these quality attributes.

The number of change requests and trouble reports is to some extent small, and future studies will be used to refine and further investigate the research questions and hypotheses presented here. This study is therefore a pre-study. This paper is structured as follows: Section 2 introduces terminology, Section 3 discusses our contribution to Statoil ASA, as well as the research context at the company. Furthermore, Section 4 introduces our research questions and preliminary data analysis, and

\(^1\) ASA stands for “allmennaksjeselskap”, meaning Incorporated.
Section 5 summarizes and discusses these preliminary results. Section 6 contains planning for further data collection and future work, while Section 7 concludes.

2. Terminology

CBSE is a new style of software development, emphasizing component reuse, which involves the practices needed to perform component-based development in a repeatable way to build systems that have predictable properties [Bass et al., 2001]. An important goal of CBSE is that components provide services that can be integrated into larger, complete applications.

Software evolution can be defined as: “...the dynamic behaviour of programming systems as they are maintained and enhanced over their life times...” [Kemerer & Slaughter, 1999]. The first studies found in literature on software evolution, were undertaken by Lehman on an OS360 system at IBM [Lehman et al., 1985]. Software evolution is closely related to software reuse, since reuse is often employed to achieve the aforementioned positive effects when evolving a system. It should be noted that several alternative uses of the term software evolution exist; some use the term to encompass both the initial development of the system and its subsequent maintenance, while others use it exclusively about the events after initial implementation, in concurrence with its original focus [Kemerer & Slaughter, 1999]. Lastly, there is some work on software evolution taxonomy [Verhoef, 2004], the author sees software maintenance as subpart of software evolution.

Software maintenance is the updating incurred on already existing software in order to keep the system running and up to date. During their lifetime software systems usually need to be changed to reflect changing business, user and customer needs [Lehman, 1974]. Other changes occurring in a software system’s environment may emerge from undiscovered errors during system validation, requiring repair or when new hardware is introduced.

Software maintenance can hence be:
- **corrective** (correcting faults)
- **preventive** (to improve future maintainability)
- **adaptive** (to accommodate alterations related to platform or environment) or
- **perfective** in response to requirements changes or additions, as well as enhancing the performance of a system [Sommerville, 2001] [Pressman, 2000].

In summary, some see the perfective and adaptive parts of software maintenance as part of software evolution [Sommerville 2001]. That is, that it encompasses both aspects of modified and added scope, as well as environmental adaptations. This does not include platform changes, which are commonly referred to as porting, instead of software evolution [Frakes & Fox, 1995]. There is, hence, no clear agreement on the definition of software evolution. Although there seems to be more agreement on the definition of the different types of software maintenance, a clear distinction between software maintenance and software evolution remains elusive.

Statoil ASA [Statoil ASA O&S Masterplan, 2006] has chosen to use defect-density and change-density (stability) as indicators of software quality. A lowered defect-density shows an increased quality, while stability in terms of change-density means a stable level of resources are needed towards adaptation and perfection of the software. In this study, it is these two measures (defect-density and change-density) we will be focusing on, in order to show how the reusable components evolve over time.

3. Our contribution to Statoil ASA, and The context

Our direct contribution is helping Statoil ASA central software development unit in Norway with defining metrics, collecting data and analyzing it. We will also be contributing towards reaching a better understanding and management of software evolution, by exploring whether that employment of reusable components can lead to better system quality2. Finally, we expect that our results will be possible to use as a baseline for comparison in future studies on software evolution.

Statoil ASA is a large, multinational company, in the oil & gas industry. It is represented in 28 countries, has a total of about 24,000 employees, and is headquartered in Europe. The central IT-department in the company is responsible for developing and delivering software, which is meant to give key business areas better flexibility in their operation. They are also responsible for operation and support of IT-systems at Statoil ASA. This department consists of approximately 100 developers worldwide, located mainly in Norway. Since 2003, 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 JEF (Java Enterprise Framework) components. This IT strategy was started as a response to the changing business and market trends, and in order to provide a consistent and resilient technical platform for development.

---

2 The quality focus at Statoil ASA is defect density and change-density (stability).
and integration [14]. The strategy is now being propagated to other divisions within Statoil ASA. The JEF framework itself consists of seven different components. Table 1 gives an overview of the three JEF releases, and the size of each component in the three releases.

<table>
<thead>
<tr>
<th>Component</th>
<th>Release 2.9</th>
<th>Release 3.0</th>
<th>Release 3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEF Client</td>
<td>7871</td>
<td>8400</td>
<td>8885</td>
</tr>
<tr>
<td>JEF Dataaccess</td>
<td>181</td>
<td>181</td>
<td>268</td>
</tr>
<tr>
<td>JEF Integration</td>
<td>958</td>
<td>958</td>
<td>958</td>
</tr>
<tr>
<td>JEF Security</td>
<td>1588</td>
<td>1593</td>
<td>2374</td>
</tr>
<tr>
<td>JEF Util</td>
<td>1312</td>
<td>1359</td>
<td>1647</td>
</tr>
<tr>
<td>JEF Workbench</td>
<td>4187</td>
<td>4515</td>
<td>4748</td>
</tr>
</tbody>
</table>


These JEF components can either be applied separately or together when developing applications. In total, we will be studying the architectural framework components, as well as two projects which use this framework. Here, we present a pre-analysis, reporting on preliminary results of studying defect-density and stability (change rate) of 6 of the 7 reusable architectural framework components, over three releases. These three releases exist concurrently, and the data is mainly from system/integration tests. The limited dataset used in this preliminary analysis is due to current data availability.

4. Research questions and Preliminary data analysis

All the statistical data presented in this study are based on valid data, as none were missing data. The statistical analysis tools we used were SPSS version 14.0 and Microsoft Excel 2003. Our preliminary research questions are regarding defect-density and stability, and are formulated as follows:

RQ1: How does the defect-density in reusable components evolve over time?

Defect-density (number of Trouble Reports/KLOC) may be seen as belonging to the corrective maintenance category by some researchers, but maintenance can also be seen as part of evolution [Verhoef, 2004]. Therefore, measuring defect-density may help characterize the evolution of the different JEF components over time. The following are the related hypotheses for RQ1:

- H10: The defect-density in JEF components do not change with time.
- H1A: There is a difference in defect-density for JEF components over time.

RQ2: How does the number of changes per reusable component (stability) evolve over time?

Research has demonstrated that reusable components are more stable (has a lower change-density) and that this does improve with time [Mohagheghi et al., 2004]. We have chosen to use change-density (number of Change Requests/ KLOC) as an indication of the stability, as this is the defined quality focus of Statoil ASA. The following are the related hypotheses for RQ2:

- H20: The change-density in JEF components does not change with time.
- H2A: There is a difference in change-density for JEF components over time.

4.1. RQ1: How does the defect-density in reusable components evolve over time?

For RQ1 we want to see how the defect-density in JEF components evolves over time, so we decided to use ANOVA test, as this is suitable for comparing the mean defect-density between the three releases. With this test, we wanted to investigate whether there is a difference in defect-density for JEF components. To investigate this research question, all submitted defects for each component were counted, per release. We then calculated the defect-density, as the number of trouble reports (TR’s) divided by kilo lines of code (KLOC) for each component. Table 2 shows the results of this calculation for three releases, all involving major changes to the software components.
Here, we want to test if there is a significant difference in the mean-values of the different releases, which we are using as groups in the analysis. Table 3 shows that the average defect-density decreases with time. The significance level is 0.05, and the data was checked for normality.

### Table 3: Average defect-density per release

<table>
<thead>
<tr>
<th>Groups</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 2.9</td>
<td>7.058</td>
</tr>
<tr>
<td>Release 3.0</td>
<td>0.510</td>
</tr>
<tr>
<td>Release 3.1</td>
<td>0.055</td>
</tr>
</tbody>
</table>

The ANOVA test we performed yielded a $F_0$ value of 7.749, and the critical value was computed to be $F_{0.005, 2, 15} = 3.682$, with a $P$-value of 0.0049. Since 7.749 > 3.682, it is possible to reject the null hypothesis. In summary, we can reject $H_0$ in favour of our alternative hypothesis $H_A$, and hence support the notion that the defect-density decreases with time.

The data trend for RQ1 reveals a declining defect-density, possibly caused by a corresponding decrease in change-density. We will, however, be expanding and verifying our hypothesis on defect-density with more empirical data in future work.

### 4.2. RQ2: How does the number of changes per reusable component (stability) evolve over time?

For RQ2 we want to see how the number of changes per JEF component evolves over time, so we again decided to use ANOVA test, as this is suitable to compare the mean change-density between the three releases. With this test, we wanted to investigate whether there is a difference in change-density for JEF components. To investigate this research question, all change requests were sorted according to JEF component and then counted, per release. We then calculated the change-density, as the number of change requests (CR) divided by kilo lines of code (KLOC) for each component. Change requests in this context mean new or changed requirements. Table 4 shows the results of this calculation.

### Table 4: Change-density per JEF component in #CR/KLOC

<table>
<thead>
<tr>
<th>Component</th>
<th>Release 2.9</th>
<th>Release 3.0</th>
<th>Release 3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>JEF Client</td>
<td>13.4672</td>
<td>0.8333</td>
<td>0.2251</td>
</tr>
<tr>
<td>JEF Dataaccess</td>
<td>0.0000</td>
<td>0.0000</td>
<td>11.1940</td>
</tr>
<tr>
<td>JEF Integration</td>
<td>3.1315</td>
<td>1.0438</td>
<td>0.6072</td>
</tr>
<tr>
<td>JEF Security</td>
<td>9.4458</td>
<td>1.8832</td>
<td>0.0000</td>
</tr>
<tr>
<td>JEF Util</td>
<td>4.5732</td>
<td>0.7358</td>
<td>0.0000</td>
</tr>
<tr>
<td>JEF Workbench</td>
<td>8.3592</td>
<td>1.1074</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

Here too, we decided to use an ANOVA test, to see if there was a significant difference in the mean-values of the different releases. Table 5 shows the variation in mean change-density over time. The significance level is 0.055, and the data were checked for normality.
Table 5: Average change-density per release

<table>
<thead>
<tr>
<th>Groups</th>
<th>Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release 2.9</td>
<td>6.496</td>
</tr>
<tr>
<td>Release 3.0</td>
<td>0.934</td>
</tr>
<tr>
<td>Release 3.1</td>
<td>2.004</td>
</tr>
</tbody>
</table>

As seen from Table 5, Release 3.0 has a lower change density than Release 3.1, indicating that the change-density may not simply decrease with time. The ANOVA test we performed yielded gave a $F_0$ value of 3.540, and the critical value was computed to be $F_{0.055, 2, 15} = 3.682$, with a $P$-value of 0.055. Since $3.540 < 3.682$, it is not possible to reject the null hypothesis. In summary, we cannot reject H20 in favour of our alternative hypothesis H2A.

Nevertheless, upon inspection of the data from Table 4, we see that for all components the change-density is lower in the following release, except for JEFdataaccess. In fact, the value JEFdataaccess has in Release 3.1 differs considerably compared to the other results. This may have specific explanation(s), which will be explored in later analysis.

5. Summary and Discussion of preliminary results

In Table 6, we have summarized our analysis results, along with corresponding research questions and hypotheses.

Table 6: Summary of the results

<table>
<thead>
<tr>
<th>Research Questions</th>
<th>Hypotheses</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1</td>
<td>$H_{10}$: The defect-density in JEF components do not change with time.</td>
<td>$H_{10}$: Rejected</td>
</tr>
<tr>
<td></td>
<td>$H_{1A}$: There is a difference in defect-density for JEF components over time.</td>
<td>$H_{1A}$: Not rejected</td>
</tr>
<tr>
<td>RQ2</td>
<td>$H_{20}$: The change-density in JEF components does not change with time.</td>
<td>$H_{20}$: Not rejected</td>
</tr>
<tr>
<td></td>
<td>$H_{2A}$: There is a difference in change-density for JEF components over time.</td>
<td>$H_{2A}$: Not rejected</td>
</tr>
</tbody>
</table>

On change-density, the data indicate that a decrease over time for five of the six components investigated. However, we are unable to conclude without further empirical data and analysis. When it comes to defect-density, our results indicate a distinct difference over subsequent releases of the JEF components. The data trend here is towards a sharp decrease. Additional trends in the size data vs. the data on change-density and defect-density exist (e.g. that some of the components have zero change density while their code size still shows an increase, or that some have high change-density while still zero defect-density) will be investigated further with more empirical data in future work. An additional possible relationship to be explored is whether large increases in change-density affect defect-density negatively, though such an effect is not indicated in the data from our preliminary analysis.

Lower defect-density means less correction are needed, and thereby a higher quality level is achieved for the reusable JEF components. When it comes to change-density, stability is important to achieve stable evolution and hence allowing for stable resources being assigned to adapt and perfect the reusable JEF components. In this way, these quality attributes can be used to partially model evolution, as they show how the quality of the reusable JEF components evolves over time.

5.1. Threats to validity

We here discuss the possible threats to validity in our study, using the definitions provided by [Wohlin, 2002]:

Construct Validity: The metrics we have used (defect-density and change-density) are thoroughly described and used in literature. Nevertheless, our definition and use of the term change-density is different from that in other studies. All our data are of pre-delivery change requests and trouble reports from the development phases for the three releases of the reusable components.
**External Validity:** The object of study is a framework consisting of only seven components, and the data has been collected for 3 releases of these components. Our results should be relevant and valid for other releases of these components, as well as for similar contexts in other organizations.

**Internal Validity:** All of the change requests and trouble reports for the JEF components have been extracted from Statoil ASA by us. Incorrect or missing data details may exist, but these are not related to our analysis of defect-density and change-density. We have performed the analysis jointly with the Microsoft Excel and SPSS tools.

**Conclusion Validity:** This analysis is performed based on an initial collection of data. This data set of change requests and trouble reports should nevertheless be sufficient to draw relevant and valid conclusions.

### 6. Planning for further data collection and Future work

So far, Statoil ASA has collected data on Trouble Reports (TR’s) and Change Requests (CR’s) for the reusable JEF components over several releases. They are also going to collect data on TR’s and CR’s for systems developed with the JEF components – so far two systems are reusing JEF components in development. Further releases of JEF components will also follow, and data will be collected on these.

In this article we have seen that there are differences in defect-density and change-density over subsequent releases, without further analysis on the differences or their causes. In further work we will be exploring these issues in more detail, as well as the possible cause-effect relation between defect-density and change-density, as well as the relation to other quality attributes. The focus may also change to encompass towards reuse and maintenance, in addition to evolution.

### 7. Conclusions

We have performed a preliminary investigation of how the quality attributes defect-density and change-density evolves over time for reusable components. While prior research has shown reusable components to be more stable (having a lower code modification rate) across releases [Mohagheghi et al., 2004], change density as defined in this context has not been studied before.

The overall results from our study are:

- For RQ1, “How does the defect-density in reusable components evolve over time?”, our results show a clear difference over releases of the JEF components. The data trend here shows a sharp decline.

- On RQ2, “How does the number of changes per reusable component evolve over time?”, our investigation on change-density shows that we cannot conclude without further data and analysis. However, the general trends in the data indicate that the change-density does decrease with time for five of the six components investigated.

In particular, lower defect-density results in less corrections being needed, hence yielding a higher level of quality of the reusable components. Such a reduction is expected if components undergo few changes between releases, e.g. in our dataset, some components have zero change-density between releases. A stable change-density is a factor towards allowing a stable evolution, hence resources used in adapting and perfecting the reusable components can be better allocated. Hence, we see that evolution can be partially modelled by looking at defect-density and change-density, as they show how the quality of the reusable components evolves over time. Our results cannot currently support these thoughts to the full extent, as the results of studying defect-density shows a decline, but the results from studying the change-density so far cannot be used to conclude, despite the apparent trends in the data. We presume that more empirical data will remedy this problem.

The SEVO project is ongoing research and this paper is meant to present preliminary results, while results will come later. We ultimately aim to look at how to reach higher quality, by demonstrating that understanding and managing software evolution can lead to better system quality.

### 8. Acknowledgement

This work has been done as a part of the SEVO project (Software EVOlution in component-based software engineering), an ongoing Norwegian R&D project from 2004-2008 [SEVO project, 2004-2008], and as a part of the first and second authors’ PhD study. We would like to thank Statoil ASA for the opportunity to be involved in their reuse projects.
9. References


