NTNU
Innovation and Creativity

An empirical study of component-based software engineering in Statoil

Vu Ha
Kiet Ve Tran

Master of Science in Computer Science
Submission date: June 2006
Supervisor: Reidar Conradi, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

Our master thesis is an extension based on our thesis written in the autumn 2005.

Assignment given: 20. January 2006
Supervisor: Reidar Conradi, IDI

An empirical study of component-based software engineering in Statoil

Authors: Vu Ha & Kiet Ve Tran

Teacher of subject: Reidar Conradi
Mentor: Anita Gupta

Date: June 16th, 2006

TDT4735 Software Engineering

Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering

NTNU
Norwegian University of Science and Technology

Abstract

Title:	An empirical study of component-based software engineering in Statoil.
Background:	This project is a continuation from indentation thesis that we wrote in the last semester. This project explores the field of software components reuse in Statoil by an empirical study. The exploration includes the study of architectural framework about SJEF and what that belongs to software components reuse. The work of the project is focusing on effort, defect density and quality attributes used to evolve reuse component.
Presenting	Phase 1: We will conduct a pre-study of the current state-of-the-art and practice and of the context about projects of Statoil. Phase 2: We will declare our focus and problem definition, and define hypotheses and research questions. Phase 3: We will present methodologies and technologies to conduct the study. We will use these methodologies to perform an empirical research based on the resources we have gathered from Statoil. Phase 4: We will present the results, evaluation, discussion and conclusion.
Thesis goals:	The overall goal for this thesis is to evaluate the software components reuse with the architectural framework. The following list presents the final goals: 1. Study the state-of-the-art in component-based software engineering. 2. Study the state-of-the-art in architecture of Statoil, SJEF 3. Explore the reuse component of Statoil, JEF 4. Explore Statoil’s projects, which are based on reuse component 5. Provide an overview of advantage or disadvantage with JEF by using empirical study and methods The final goals of the thesis have emerged through the increasing knowledge to the state-of-the-art and how mature the concepts of component-based software engineering and software product lines were at the organization.
Competence:	We will base us upon existing empirical methodologies, hypothesis testing, statistics, strategies and analysis, such as Goal-Questions-Metric (GQM), ANOVA, graphical visualization, validity evaluation and descriptive statistics.
Keywords:	Statoil, SJEF, effort, incident, defect density, CBSE, CBD, reuse, component, quality, resource, evaluation, methodology, technology.

Preface

This project is written in the course TDT4735 Software Engineering, in the final year and semester of study at the Department of Computer and Information Science at Norwegian University of Science and Technology, NTNU. The result is an evaluation and analysis of component reuse in Statoil with regards to effort, defects and quality attributes.

We would like to thank teacher of subject, Professor Reidar Conradi, who has helped us throughout the study. We would also thank our mentor Anita Gupta and Odd Petter Slyngstad in NTNU for providing us with important information and resources from Statoil. Our conversations with them have proved to be invaluable for this thesis.

Trondheim, 16th of June, 2006.

Vu Ha   Kiet ve Tran

Table of Contents

List of figures

List of tables

List of equations

1 Introduction

This introduction sets the stage for the thesis, describing the motivation, objectives, followed by the work context, report structure and reading guild. This chapter will give details about the motivation behind this thesis and under what kind of environment it has been written.

1.1 *Project context*

This project is a part of INCO (INcremental and COmponent-base development) and it is a continuation from the in-depth study from our last semester. This work has been carried out in last years by professors and scholarship holders from NTNU, who cooperates with employees of Statoil and several other Norwegian companies.

Reuse component-based software has become highly attractive in software industries among development projects, because reuse component insures to reduce the costs, shorter time to market and increase the qualities of software. Statoil was developing a new information systems generation in order to provide support for current and future central business areas. These systems must be easily adaptable to meet a changing and increasingly demanding business environment, and should be developed within a common architecture to address natively built software as well as COTS software. This new architecture is called SJEF.

Our in-depth study last year had a lot of information about state-of-the-art and Statoil context. In this report we will not rewrite the facts, but we will add the relevant and missing facts. The purpose of this project is to illustrate the results according to hypothesizes. Hypothesizes are acknowledged by empirical method. The following parts of this report will be using a lot of empirical methods and strategies to achieve the purposes and goals of our study.

1.2 Motivation

The use of components in software development is generating a huge interest both as a research area, and in commercial use. Component Based Software Engineering, CBSE, shows potential in decreasing development time and product costs, increase product reliability and stability. The system flexibility might be improved by allowing new components to replace old ones and systems could get a standardized architecture.

Our motivation is to carry out the empirical studies of Statoil's developed projects by reusing component based software with the architectural framework JEF. We will try to verify the hypotheses using data from Statoil's projects. From the result what we will get, we will search about reuse component based software in Statoil's developed projects have positive effects and quality impact, or opposite. By doing this work, several researches will be defined and many evaluations will be done to support the conclusion.

1.3 Focus

In the autumn of 2005 we wrote a thesis about the introduction of component-based software development in Statoil. This new development was deployed as something called JEF – a project which uses best practices and guidelines and defines a set of building block and design principles for construction and integration of enterprise class information systems. There we discussed the advantages of SJEF had on Statoil's general business. We focused on the cost of developing new projects in JEF and how accurate the project estimations were.

We wish to continue the work what we had done in our last thesis. We had a difficulty to find enough resources to make a throughout investigation of our hypotheses. We will therefore use this master thesis to complete our research, as well as look into other issues related to the introduction of JEF in Statoil.

The first part of our research will focus on the hypotheses in our thesis. There we used the metric "cost" as a measure for projects in JEF and JEF itself. Although "cost" is an adequate measure for projects, we do not think this metric can fully cover every aspect of a project. We have chosen to do the same investigation of project – to see if JEF has been advantageous – with the metric "effort" instead. Our definition of effort will be:

Effort = [Total cost] / [Total number of physical Line-of-Code]

We have chosen effort as the metric because effort gives a better basis for comparison between several projects. The projects will probably not be in the same range of size, and using the metric cost will not be able to express this differentiation. Cost alone can not express productivity, efficiency and complexity either, while effort can with some interpretations. A pitfall we should be observant of is that the cost of project will not be linear with the size, but rather exponential.

The second part of our research will explore what we had wished to do in our thesis. This part is partially documented in our thesis under "Further Work". We had planned to make a research about defects and quality attributes, but we had to exclude them from our final paper. The reason behind this is that we were running out of time and we did not want to overextend our scope and we did not have access to all the data resources we wanted. In addition report of defects in JEF already exists. We have therefore chosen to focus on defects and quality attributes in our master thesis.

We will use the following metrics for defect: Number of defects, defect density (defects per LOC), defect types, defect severity, defect class, what phase the defect is in (completed, closed, open etc) and defect priority. We wish to see the evolution of stated metrics over time and versions to see if component-based software engineering can improve the versions. We will also make comparisons between different projects in JEF.

We wish to emphasize on stability regarding quality attributes. Here, we will investigate the relationship in change request between the different projects. The metric to define the stability will be presented later in the document.

The following table will give an overview of assumed data resources we will use to make this investigation and assumed focuses we will search.

Focus	Under focus	Details	Resources
Effort (Cost / #SLOC)	JEF and other projects	The comparison is based on effort in JEF, DCF and S&A to each other. In this way to conduct the most advantageous project among them.	- DCF Project Handbook - S&A Project Handbook - JEF Component Overview - JEF Release info
	Versions within a project	The comparison is based on effort in released versions from each project. In this way to conduct the development from the one to another version has been improved or opposite.	
Defect density (Defects / #SLOC)	JEF and other projects	The comparison is based on defect density in JEF, DCF and S&A to each other. Priority to correct the defect and severity of defect can also be compare	- Final test report DCF - Final test report S&A - Final test report JEF
	Versions within a project	The comparison is based on defect density in released versions from each project. In this way to conduct the development from the one to another version has been improved or opposite.	
Stability	JEF and other projects	Number of change request from a project can be compared to another project to conduct the stability of the project.	- DCF change report - JEF change report - S&A change report

Table 1-1: Goal and focus

The following table illustrates the available projects and version we have access to.

Statoil's project	Version of project
DCF	Version 1.0 Version 1.1 Version 2.0
S&A	Version 1.0
JEF	Version 2.9 Version 3.0 Version 3.1

Table 1-2: The versions of projects analyzed in this thesis

1.4 Goal definition

Developing software is not an exact science. There are many ways to reach the goal of implementing a given set of requirements, and there are different demands for different systems. Some systems need a high degree of security, such as medical journals or bank transaction systems. Some need high usability, such as check-in machines at airports or automatic ticket sales machines at train station, and some need high degree of safety, such as airspace control systems. The latter type of systems is called safety-critical system, since a failure in these systems may cause harm to persons or equipment.

In the case of Statoil with SJEF, it is little different developing software as mention above. Developers establish new components (software) with SJEF, and the new component includes previous components functionality. In another word, it tells that Statoil's developing software is reusing components with SJEF. This project is to study the data report of components reuse with architecture SJEF, and to underline the advantage or disadvantage with SJEF. Our work in this project is to provide a research result by using empirical studies of software components reuse with SJEF in Statoil. The main objective with the work presented here will be as follow:

> *"Provide an investigation of components reuse with SJEF and whether it is an advantage or disadvantage by using empirical study."*

This main objective can be parted into more detailed objectives presented in table below, together with their rationale and what research questions they address:

Sub-objectives	Rationale	Research question
Provide a statement to see if reusing software components with SJEF gives an advantage or disadvantage to effort.	As a starting point for our work we need to clarify what SJEF is, how do the system work and what kind of technologies that support SJEF.	• What do we mean by effort? • What do we mean by advantage and disadvantage? • How can we measure the effort?
Provide a statement to see if reusing software components with SJEF gives an advantage or disadvantage to defect density.	We need to know what kind of defect it exists, severity of defect and priority to correct the defect.	• What do we mean by defect density? • What kind of classification is it specified in defect? • How can we measure the defect density?
Provide a statement to see if reusing software components with SJEF gives an advantage or disadvantage to stability.	We need to clarify what is the change request.	• What do we mean by stability? • How can we measure the stability?

Table 1-3: Main goal and sub-goals

1.5 Report process

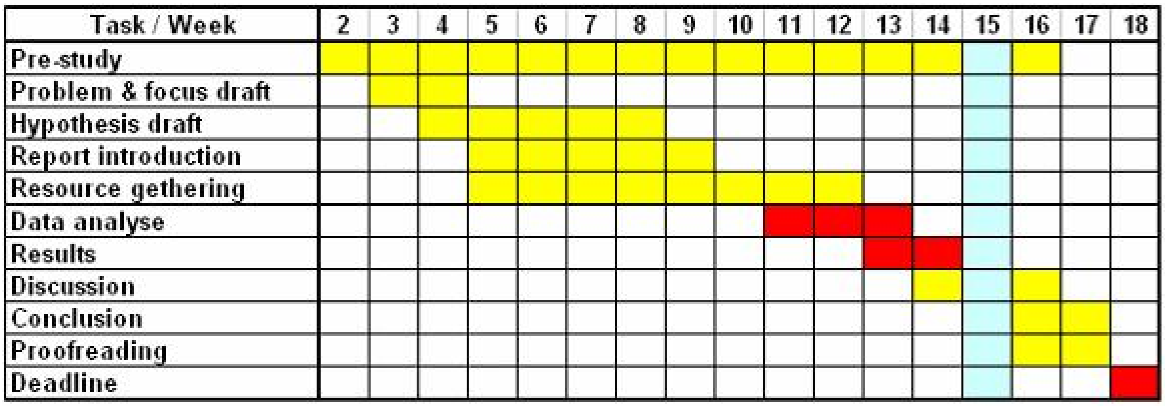
Here present our schedule of work with the report. Our work proceeds in four phases. It is following:

Phase 1
It conducts a pre-study of the current state-of-the-art and practice and the context about Statoil's projects.

Phase 2
It declares our focus and problem definition, and defines hypotheses and research questions.

Phase 3
It presents methodologies and technologies to conduct the study. We will use these methodologies to perform an empirical research based on the resources what we have gathered from Statoil.

Phase 4
It presents results of evaluated and analyzed data, validation and discussion of the work, and conclusion of the project. Here is our schedule of report outline.



Task / Week	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Pre-study	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦	🟨		
Problem & focus draft		🟨	🟨											🟦			
Hypothesis draft			🟨	🟨	🟨	🟨	🟨							🟦			
Report introduction				🟨	🟨	🟨	🟨	🟨						🟦			
Resource gethering				🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨			🟦			
Data analyse										🟥	🟥	🟥		🟦			
Results												🟥	🟥	🟦			
Discussion													🟨	🟦	🟨		
Conclusion														🟦	🟨	🟨	
Proofreading														🟦	🟨	🟨	
Deadline														🟦			🟥

Figure 1-1: Report outline

1.6 Reading Guide

To fully understand to the contents in this thesis, the reader should have a background from the system engineering discipline. It also an advantage if the reader has some background in quality analysis and component-based software engineering.

This sub-chapter will provide the reader with an overview of the master thesis. This report is organized in 5 parts and in each part consists of a number of chapters and subchapters.

Part 1: Introduction – Chapter 1
This part contains the introduction to the thesis and presents the motivation, objective definition, project description, project context, document layout and the reader's guide. This chapter will provide in-depth background information behind the thesis. Readers who which to get a full overview of this thesis should not skip this part.

Part 2: Pre-study – Chapter 2 and 3

Chapter 2 is an introduction to component technology advances that have been utilized. If the reader is interested in the state-of-the-art of CBSE and software product lines, read this chapter. It can be skipped if the reader already is familiar with the concepts.

Chapter 3 describes the background information and the current situation at Statoil. It will explain some of the terms and measures that are used in the resources that will be analyzed later.

Part 3: Hypotheses – Chapter 4

This part illustrates the research focus, hypotheses and methods for our study. Readers are suggested to read this chapter before going on to next part. This part is one of the main points in this report and it provides a more understandable view of the whole report. Our work is based on this part.

Part 4: Results and conclusions – Chapter 5, 6, 7 and 8

Chapter 5 presents the data resources that will be analyzed. The data resources is presented with brief explanations, and visualized through diagrams. This chapter can be skipped if the reader does not have any special interest in the actual data.

Chapter 6 presents the results to the stated hypotheses. The result is provided by studying the data presented in chapter 5. This chapter is recommended as it will relate all the data resources to the hypotheses in chapter 4.

Chapter 7 presents an evaluation of the results given in chapter 6, and discusses the threats to validity. This chapter will then make a discussion about our study and provide suggestions for improvements.

Chapter 8 presents the conclusion of this thesis and provides plans for further work. This chapter is strongly recommended to read as it will conclude this whole master thesis.

Part 5: Appendices – Chapter 9

This part contains references as well as abbreviations used in this thesis.

2 Pre-study: Survey of state-of-the-art and practice

This chapter will present the current state-of-the-art that, and in particular those fields that are related to our work.

2.1 Component-based software engineering

Recently, software component technology, which is based on building software systems from reusable components, has attracted attention because it is capable of reducing developmental costs. In a narrow sense, a software component is defined as a unit of composition, and can be independently exchanged in the form of an object code without source codes. The internal structure of the component is not available to the public.

The characteristics of the component-based development are the following:

- Black-box reuse
- Reactive-control and component's granularity
- Using RAD (rapid application development) tools
- Contractually specified interfaces
- Introspection mechanism provided by the component systems
- Software component market (CALS)

CBSE is the key aspect of reuse. In order to successfully reuse old software, we must have a unit, which is a component. We can derive the following definition of a component from [12]:

> *"A component is a language-neutral, independently implemented package of software services, delivered in an encapsulated and replaceable container, accessed via one or more published interfaces. A component is not platform constrained or application bound."*

CBSE involves designing and implementing these software components and assembling software systems from pre-built components. There are several types of components: COTS (Commercial-Off-The-Shelf), GOTS (Government Off-The-Shelf), MOTS (Modifiable Off-The-Shelf) and OSS (Open Source Software). Together, these components can gather and create an entire software architecture, product line or system family. CBSE and reuse promise many advantages to system developers and users such as:

- Shortened development time, and reduced total cost, since systems are not developed from scratch.
- Reduced maintenance cost, since it is not necessary to recompile multiple systems because an adjustment was made to a single component
- Facilitation of more standard and reusable architectures, with a potential for learning.
- Separation of skills, since much complexity is packaged into specific frameworks.
- Fast access to new technology, since we can acquire components instead of developing them in-house.
- Improved reliability by shared components
- The relations between components are loose. This dampens the ripple effect when changes are needed.

These advantages are achieved in exchange for dependence on component providers, vague trust to new technology, and trade-offs for both functional requirements, and quality attributes.

2.2 Defect reporting

In recent years the emphasis on software quality has increased due to forces from several sectors of the computer industry. In the software development, successful companies are interested in continually improving their software development processes. In order to improve their processes there are many pieces of data that can be examined. One such piece is defect data. Defect data is an important piece of data for software companies because it shows without a doubt the places that thy have made mistakes. Some of those mistakes may have been unavoidable, but many are due to human error. That error may come in the form of commission, omission, misunderstandings, etc. If a software development company is able to determine a specific type of error that is common over many projects, then they have located an ideal place to start improving their process. In order to do this, data visualization can be very useful. What a company is looking for is a correlation between some aspect of the defects and the effort taken of fix those defects.

Here is suggestion for how interpret the classification to describe the structure of the defects:

Classification	Detail	Description
Miss	Design	A certain requirement is covered nowhere in the design. This is stronger than IC where the coverage is present, but incomplete.
	Implementation	A certain part of a design was not implemented.
	Error handling	An error case was not handled in the program (or not handled properly).
	Assignment	A single variable was not initialized or updated. Only one statement needs to be added.
	Call	A single method call is missing. Only one statement needs to be added.
Wrong	Algorithm	The entire logic in a method is wrong and cannot provide the desired functionality. More than one statement needs to be added or changed
	Expression	An expression (in an assignment or method call) computes the wrong value. Only one expression needs to be changed.
	Condition	A Boolean expression was wrong. Only one expression needs to be changed
	Name	Objects or their names were confused. The wrong method, attribute, or variable was used. Only one name needs to be changed.
	Type	Two `similar' types were confused.

Table 2-1: The structure of defects

Although defect reporting is the key to improve software engineering, it exists a lack of empirical studies in this field. Defect data is one of the few direct measurements of software quality. There can be a great variation of defect reporting since there are no standards for defect-reporting systems. And often the defect data is not gathered with the intention of assessing software quality. These are some of the reasons why this field has received less attention that it deserves.

2.2.1 Studies on defect-density in the context of reuse

There are not many studies on defect-density in the context of reuse. Melo, Briand and Basili have studied the impact of reuse in software quality and productivity in object-orientated systems [13][12]. One study was performed on a population that consisted of graduate level students at the Department of Computer Science at the University of Maryland. The students were randomly placed in 8 different groups that each developed an information system based on the waterfall method. Each of these groups had a various degree of reuse, which was divided into four categories [13]:

1) Verbatim – Class 'C' is put into the library and used as it was created without modifications. The reuse rate is 100%.
2) Slightly modified – Class 'C' is a new class which has been created by specializing, through inheritance, a library class. The reuse rate is more than 25%.
3) Extensively modified – Class 'C' has been created by modifying an existing class. The reuse rate was less that 25%.
4) New – Class 'C' has been developed from scratch. The reuse rate was zero.

This study came to the following results.

Project	No. Of LOC	Reuse Rate	Error Density
1	23354	71.83	1.03
2	5068	2.23	6.51
3	9735	31.44	4.31
4	8543	18.08	3.86
5	8173	40.05	3.18
6	6368	48.67	3.93
7	6571	64.01	2.28
8	5068	0.00	8.68

Table 2-2: Relation between LOC, defect density and reuse rate

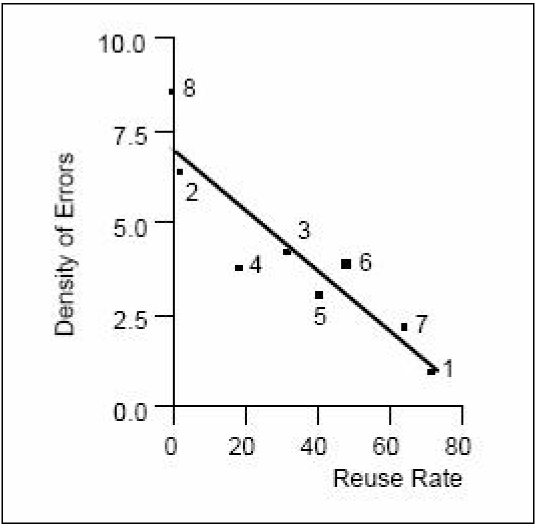


Figure 2-1: Relation between error density and reuse rate

The error density is given as Errors/KLOC.

This, they claim, is very strong evidence that reuse linearly reduces the defect density. Further on, they claim in their study:

> "[…] *that means that, when there is no reuse, error density should be expected to be around 6.55 and each additional 10 percentage points in the reuse rate decreases this density by nearly 1"*

They claim that their results of data analysis and conclusion can be generalized as follows:

> *"Each additional 10 percentage points in reuse rate, within the reuse rate interval covered by our data set, decreases rework by nearly 8.5%."*

Their studies also conclude that increased reuse rate results in linearly increasing productivity and reducing effort. They state the following:

> *"When there is no reuse, productivity should be expected to be around 14 SLOC per hour and each additional 10 percentage points in the reuse rate should increase productivity by 11 SLOC per hour."*

Parastoo Mohagheghi performed another study on the context of reuse. His goal was main goal was to evaluate parameters that are earlier studied in traditional reliability models, in the context of reuse, and assess the impact of reuse on software quality attributes [9]. The quality in focus in this study was stability and defect density. This paper is in particular interest because our goal is quite similar to his, and his results will give us a better understanding of our research agenda.

The paper describes the results of an empirical study on the impacts of reuse on defect-density and stability on a large-scale telecom system developed by Ericsson. The following table shows the hypotheses and results from the study concerning the defect density.

HypID	Hypothesis Text	Result
H1	**H01:** Reused components have the same defect-density as non-reused ones.	Rejected
	H02: Reused components have lower defect-density that non-reused ones.	Accepted
H2	H02-1: There is no relation between number of defects and component size for all components.	Not rejected
	H02-2: There is no relation between number of defects and component size for reused components.	Not rejected
	H02-3: There is no relation between number of defects and component size for non-reused components.	Rejected
H3	H03-1: There is no relation between defect-density and component size for all components.	Not rejected
	H03-2: There is no relation between defect-density and component size for reused components.	Not rejected
	H03-3: There is no relation between defect-density and component size for non-reused components.	Not rejected

Table 2-3: Research hypotheses and results of [9]

The results shows that reused components have lower defect-density that non-reused ones. Further on, the paper claims:

> "*Reused components have more defects with highest severity than the total distribution, but less defects after delivery, which shows that that these are given higher priority to fix.*"

This shows that there is a clear advantage of reusing software components.

2.3 Effort

In the public is often defined the effort and the time are closed together, because they have always been seen close like moneys. That is why, business peoples usually say; time is money. The effort may be able to be seen like the income from a business. The general business profitability is depended on the relationship between business income and cost. The income of the business is represented by the sale. But in our study of the effort from Statoil's project is relationship between the work and the money had invested to the work.

2.4 Quality of system

Quality is defined as meeting the customer's expectations and ensuring that the parts work when assembled in the final product. System's quality facilitates efficient, effective and comfortable use by a given set of users for a set of purposes under specified conditions. In our case, we are interested in two attributes of system's quality. There are modifiability and stability.

2.4.1 Modifiability

The world around software systems is constantly changing. Software that used to function properly turns incompatible because their environments change. For this reason most software systems need to be modified many times after their first release. Another reason for modifying software is when software product are updated and improved to keep the competitive advantage against other products on the market. The result is software products that evolve from release 1, to release 2, to release 3, and so forth. In developing a software product today, it is accepted and expected that there will be subsequent release of the product. The software life cycle of a product is continuous development that only ends when the product is obsolete.

The changes that are implemented from release to release include anticipated change, e.g. following that market plan, and unanticipated ones, e.g. provoked by sudden and unexpected changes from different sources. Additionally, the bug fixes for the previous release are usually incorporated into the code baseline. Based on our understanding about the future of the software system, we can choose the design alternatives that support futures modifications better otherwise equivalent alternatives.

The goal is to increase the productivity in the subsequent release by choosing the most appropriate design solution from the start. Consequently, stakeholders are generally interested in a system designed such that future changes will be relatively easy to implement, and thus increase the maintenance productivity for implementing these change. Typical questions that stakeholders pose during the early design stage, i.e. software architecture design, include:

- Which of the available architecture design requires the lowest cost to maintain, i.e. yields the highest productivity to accommodate future changes?
- How much effort will be needed to develop coming releases of the system?
- Where are the trouble spots in the system with respect to accommodating changes?
- Is there any other construction that is significantly easier to modify than the present one?

These questions concern a system's ability to be modified.

In the software engineering literature a large number of definitions of qualities exist that are related to modifiability. A very early classification of quality [08] includes the following definitions:

> *"Maintainaility is the effort required to locate and fix an error in an operational program. Flexibility is the effort required to modify an operational program."*

A more recent classification of qualities is given in the [02]. This standard includes the following definition, related to modifying system:

> *"Maintainability is the capability of the software product to be modified. Modifications may include corrections, improvements or adaptations of the software to change in environment, and in requirements and functional specification."*

2.4.2 Stability

The definition of stability presents the quality of maintaining a constant character in the presence of forces, which threaten to disturb it, resistance to change. Stability refers the invariability of a specified property of a substance, device, or apparatus with time, or under the influence of typically extrinsic factors. Stability can be categorized as:

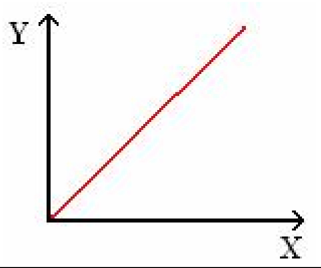
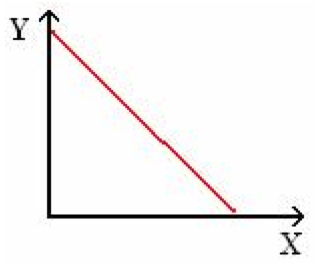
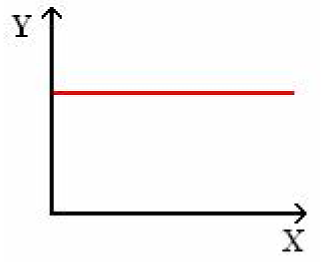
Positive stability	Negative stability	Neutral stability
		
Tends to return to original condition	Diverges away from original condition	Remains at new condition (does not move further away or closer)

Table 2-4: The categorization of stability

Stability is an important factor when software is developed in a waterfall model or developed over several versions or releases. Parastoo Mohagheghi's article [9] also discusses the impact of software reuse on stability. Stability, as the degree of modification, investigated along with defect density in the context of reuse. The following table shows the results of the investigation concerning stability.

H4	H04: Reused and non-reused components are equally modified. HA4: Reused components are modified more than non-reused ones.	Rejected Rejected

Table 2-5: Research hypotheses and results of [9]

Reused components were less modified (more stable) than non-reused ones between successive releases. Again, this shows the clear advantages of reusing software components.

2.5 Empirical theory

This subchapter describes the state-of-the art of measurement in software engineering. Below we will present a brief description of some statistical methods used in this thesis. We will give an introduction to the ANOVA method followed by an overview of different data analysis methods and graphical visualization follows the statistical result presentation.

2.5.1 Threats to validity

A study is *valid* if its measures actually measure what they claim to, and if there are no logical errors in drawing conclusions from the data. There are many labels for different types of validity, but they all have to do with threats and biases which would undermine the meaningfulness of research. We will mention the different types of validity, and we will also mention some questions we should be asking ourselves about the validity of our study.

Then what does validity means? We have chosen the definition given by William M. K. Trochim [14]:

> *"[Validity is] the best available approximation to the truth of a given proposition, inference, or conclusion."*

And one of the first questions we have to ask ourselves is: "validity of what?". Measures, samples and designs do not *have* validity -- only propositions or conclusions can be said to be valid. Technically, we should say that a measure leads to valid conclusions or that a sample enables valid inferences. It is a proposition, inference or conclusion that can *have* validity.

We subdivide validity into four types. Each type addresses a specific methodological question. In order to understand the types of validity, you have to know something about how we investigate a research question. Because all four validity types are really only operative when studying causal questions, we will use a causal study to set the context.

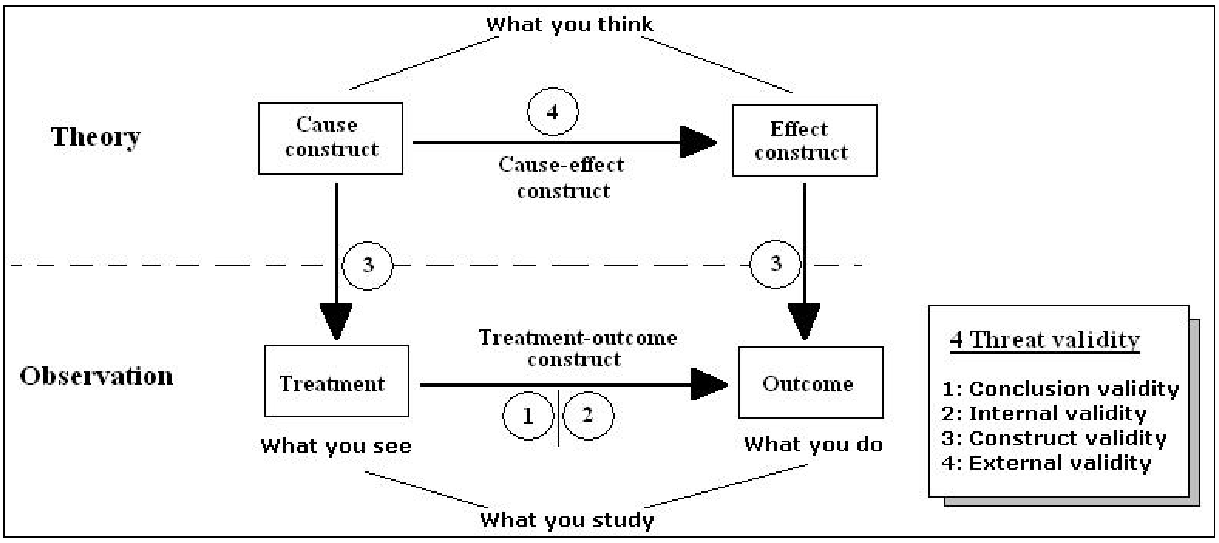


Figure 2-2: The principles of validity (adapted from Trochim)

The figure 2-2 shows that there are two sides that are involved in our study. The first, on the top, is the land of theory. It is what goes on inside our heads as researchers. It is where we keep our theories about how the world operates. The second, on the bottom, is the land of observations. It is the real world into which we translate our ideas -- our programs, treatments, measures and observations. When we conduct a study, we are continually moving back and forth between these two sides, between what we think about the world and what is going on in it.

The four types of validity:
As mentioned earlier, there are four types of validity. They build on one another, with two of them (*conclusion* and *internal*) referring to the land of observation on the bottom of the figure xxx, one of them (*construct*) emphasizing the linkages between the bottom and the top, and the last (*external*) being primarily concerned about the range of our theory on the top.

We will give a further and detailed explanation of each of these four types.

Conclusion validity:

This validity concerns the relationship between two variables. We have to ask ourselves: *"In this study, is there a relationship between the two variables?"*

Threats to conclusions validity are concerned with the issues that affect the ability to draw a correct conclusion about relations between the treatment and the outcome. These issues include, for example, choice of statistical test and choice of sample sizes.

Internal Validity:

This validity will further investigate the relations in this study. It concerns whether or not the relation between two variables is a casual one. We have to ask ourselves: *"Assuming that there is a relationship in this study, is the relationship a causal one?"*

Factors that can influence the internal validity are how the subjects are gathered and sorted into groups, how the subjects are treated and compensated during the study. The figure below will illustrate the factors.

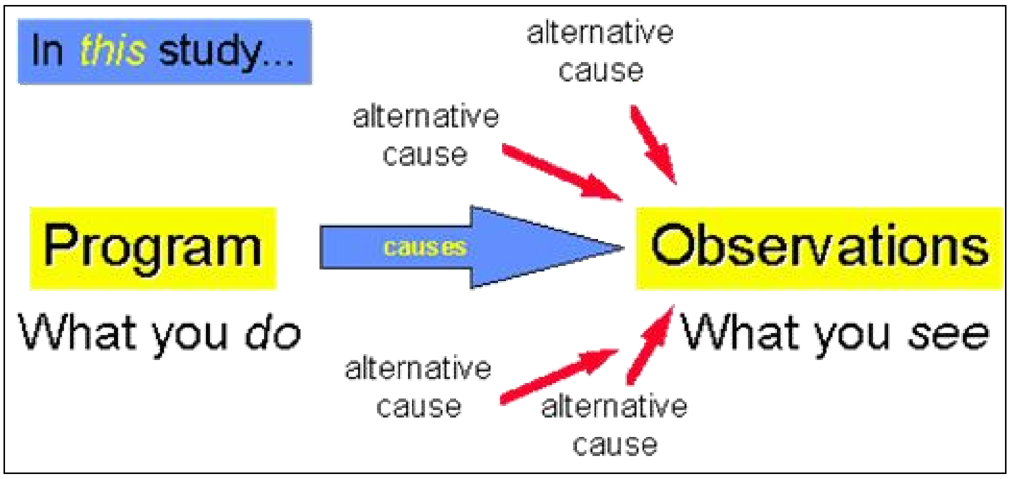


Figure 2-3: Internal validity and the threats

Construct Validity:

This validity concerns the relation between theory and observation. We have to ask ourselves: *"Assuming that there is a causal relationship in this study, can we claim that the program reflected well our construct of the program and that our measure reflected well our idea of the construct of the measure?"*

For example, the number of courses taken at the university in computer science may be a poor measure for experience in a programming language. A better measure might be the number of years with practical use.

External Validity:

We have to ask ourselves: *"Assuming that there is a causal relationship in this study between the constructs of the cause and the effect, can we generalize this effect to other persons, places or times?"*

We are likely to make some claims that our research findings have implications for other groups and individuals in other settings and at other times. When we do, we can examine the external validity of these claims. There are three major risks which can determine whether our study has external validity or not: Having wrong participants, conducting the study in the wrong environment and conducting the study at a timing that affects the results. These are shown at the figure below.

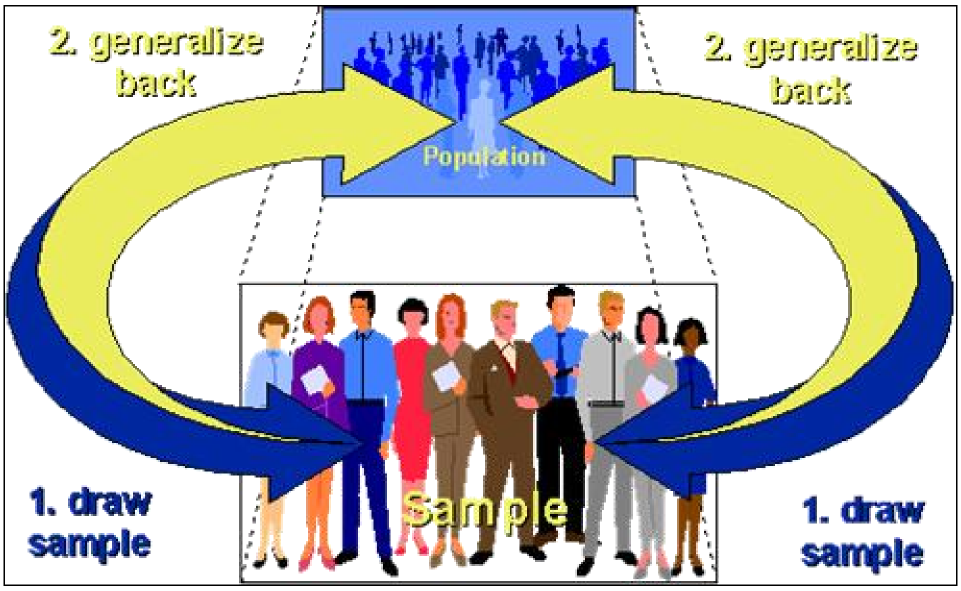


Figure 2-4: External validity and the threats

Notice how the question that each validity type addresses presupposes an affirmative answer to the previous one. This is what we mean when we say that the validity types build on one another. The figure below shows the idea of cumulativeness as a staircase, along with the key question for each validity type.

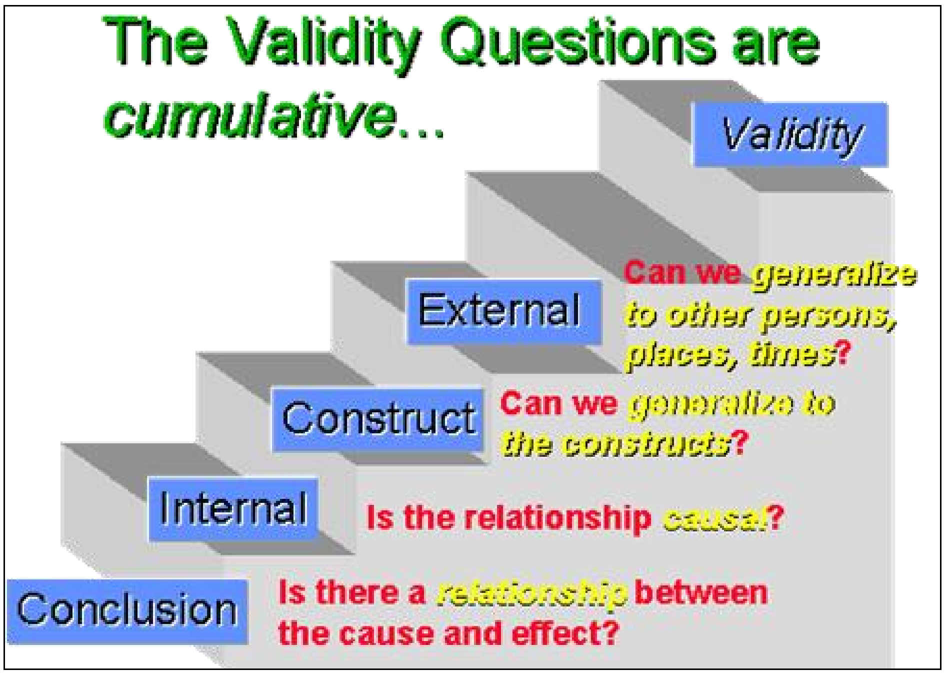

The Validity Questions are cumulative...
Validity
External
Can we generalize to other persons, places, times?
Construct
Can we generalize to the constructs?
Internal
Is the relationship causal?
Conclusion
Is there a relationship between the cause and effect?

Figure 2-5: The questions we must ask ourselves about validity

2.5.2 Measurement definitions

The theory of measurement provides the rigorous framework for determining when a proposed measure really does characterize the attribute it is supposed to. The theory also provides rules for determining the scale types of measures, and hence to determine what statistical analyses are relevant and meaningful. We make a distinction between a measure (in the above definition) and a metric. A metric is a proposed measure. Only when it really does characterize the attribute in question can it truly be called a measure of that attribute. For example, the number of Lines of Code (LOC) (defined on the set of entities "programs") is not a measure of "complexity" or even "size" of programs (although it has been proposed as such), but it is clearly a measure of the attribute of length of programs.

2.5.3 Statistical analysis

Statistical calculations are often a technical sideshow; the primary interest is in some substantive question. Even so, the methodological issues need careful attention, as we have argued. However, in many cases the substantive issues are very close to the statistical ones. For instance of litigation involving claims of racial discriminations, the substantive research question is usually operationalized as a statistical hypothesis. Hypothesis testing is the basis for statistical analysis in an empirical study. Setting up and testing hypotheses is an essential part of statistical inference. In order to formulate such a test, usually some theory has been put forward, either because it is believed to be true or because it is to be used as a basis for argument, but has not been proved. Hypothesis is used to reject the hypothesis if possible. There are two hypotheses we have to formulate:

Null Hypothesis - H_0

The null hypothesis, H_0 represents a theory that has been put forward, either because it is believed to be true or because it is to be used as a basis for argument, but has not been proved. For example, in a clinical trial of a new drug, the null hypothesis might be that the new drug is no better, on average, than the current drug. We would write H_0: there is no difference between the two drugs on average.

Alternative Hypothesis - $H_\propto$

The alternative hypothesis, $H_\propto$, is a statement of what a statistical hypothesis test is set up to establish. For example, in a clinical trial of a new drug, the alternative hypothesis might be that the new drug has a different effect, on average, compared to that of the current drug. We would write $H_\propto$: the two drugs have different effects, on average. The alternative hypothesis might also be that the new drug is better, on average, than the current drug. In this case we would write $H_\propto$: the new drug is better than the current drug, on average.

If we conclude to reject the null hypothesis then it suggests that the alternative hypothesis may be true. Else we suppose that the null hypothesis is true.

Type Error

In a hypothesis test, a type I error occurs when the null hypothesis is rejected when it is in fact true; that is, H_0 is wrongly rejected. For example, in a clinical trial of a new drug, the null hypothesis might be that the new drug is no better, on average, than the current drug; that is H_0: there is no difference between the two drugs on average. A type I error would occur if we concluded that the two drugs produced different effects when in fact there was no difference between them.

In a hypothesis test, a type II error occurs when the null hypothesis H_0, is not rejected when it is in fact false. For example, in a clinical trial of a new drug, the null hypothesis might be that the new drug is no better, on average, than the current drug; that is H_0: there is no difference between the two drugs on average. A type II error would occur if it was concluded that the two drugs produced the same effect, that is, there is no difference between the two drugs on average, when in fact they produced different ones.

The following table gives a summary of possible results of any hypothesis test:

		Decision	
		Reject H_0	Don't Reject H_0
Truth	H_0	Type I error	Right decision
	H_1	Right decision	Type II error

Table 2-6: The possible results of a hypothesis test

A type I error is often considered to be more serious, and therefore more important to avoid, than a type II error. The hypothesis test procedure is therefore adjusted so that there is a guaranteed 'low' probability of rejecting the null hypothesis wrongly; this probability is never 0.

One-sided Test
A one-sided test is a statistical hypothesis test in which the values for which we can reject the null hypothesis, H_0 are located entirely in one tail of the probability distribution. In other words, the critical region for a one-sided test is the set of values less than the critical value of the test, or the set of values greater than the critical value of the test. A one-sided test is also referred to as a one-tailed test of significance. The choice between a one-sided and a two-sided test is determined by the purpose of the investigation or prior reasons for using a one-sided test. Here is an example to facilitate an easier understanding.

Suppose we wanted to test a manufacturer claim that there are, on average, 50 matches in a box. We could set up the following hypothesis:

$$\boxed{H_0 : \mu = 50 \text{ against } H_1 : \mu < 50 \text{ or } H_1 : \mu > 50}$$

Equation 2-1: An example hypothesis for one-sided test

Either of these two alternative hypotheses would lead to a one-sided test. Presumably, we would want to test the null hypothesis against the first alternative hypothesis since it would be useful to know if there is likely to be less than 50 matches, on average, in a box (no one would complain if they get the correct number of matches in a box or more).

Yet another alternative hypothesis could be tested against the same null, leading this time to a two-sided test:

$$\boxed{H_0 : \mu = 50 \text{ against } H_1 : \mu \neq 50}$$

Equation 2-2: An alternate example hypothesis for one-sided test

That is, nothing specific can be said about the average number of matches in a box; only that, if we could reject the null hypothesis in our test, we would know that the average number of matches in a box is likely to be less than or greater than

Two-Sided Test
A two-sided test is a statistical hypothesis test in which the values for which we can reject the null hypothesis, H_0 are located in both tails of the probability distribution. In other words, the critical region for a two-sided test is the set of values less than a first critical value of the test and the set of values greater than a second critical value of the test A two-sided test is also

referred to as a two-tailed test of significance. The choice between a one-sided test and a two-sided test is determined by the purpose of the investigation or prior reasons for using a one-sided test.

Suppose we wanted to test a manufacturer claim that there are, on average, 50 matches in a box. We could set up the following hypotheses:

$$\boxed{H_0 : \mu = 50 \text{ against } H_1 : \mu < 50 \text{ or } H_1 : \mu > 50}$$

Equation 2-3: An example hypothesis for two-sided test

Either of these two alternative hypotheses would lead to a one-sided test. Presumably, we would want to test the null against the first alternative hypothesis since it would be useful to know if there is likely to be less than 50 matches, on average, in a box (no one would complain if they get the correct number of matches in a box or more).

Yet another alternative hypothesis could be tested against the same null, leading this time to a two-sided test:

$$\boxed{H_0 : \mu = 50 \text{ against } H_1 : \mu \neq 50}$$

Equation 2-4: An alternate hypothesis for two-sided test

That is, nothing specific can be said about the average number of matches in a box; only that, if we could reject the null hypothesis in our test, we would know that the average number of matches in a box is likely to be less than or greater than 50.

2.5.4 ANOVA

Analysis of Variance (ANOVA) allows us to extend this to more than two populations or measurements (treatments). The name, ANOVA, is used because the method is based on looking at the total variability of the data and the variability partition according to different components. In its simplest form the test compares the variability due to treatment and the variability due to random error. Below it is described how to use ANOVA in its simplest form. The test can be used to compare if a number of samples has the same mean value. That is, the design is one factor with more than two treatments. The test can be summarized as:

<u>ANOVA, one factor, more than two treatments</u>
Input A samples: $x_{11}, x_{12}, \ldots x_{1n_1}; x_{21}, x_{22}, \ldots x_{2n_2}; \ldots; x_{a1}, x_{a2}, \ldots x_{an_a};$
H_0 We can test the follow: 1. Are all the means from more than two populations equal? 2. Are all the means from more than two treatments on one population equal? (This is equivalent to asking whether the treatments have any overall effect.) $\mu_{x1} = \mu_{x1} = \ldots = \mu_{xa}$; All expected means are equal

Conclusion

The results given this test can we given in a table like down below.

Source of variation	Sum of squares	Degrees of freedom	Mean square	F_0
Between Treatments	$SS_{Treatment}$	a-1	$MS_{Treatment}$	$F_0 = MS_{Treatment} / MS_{Error}$
Error[a]	SS_{Error}	N-a	MS_{Error}	
Totalt	SS_T	N-1		

Formula:

$SS_T = \sum_{i=1}^{a} \sum_{j=1}^{n_i} x_{ij}^2 - \frac{x^2}{N}$	$SS_{Treatment} = \sum_{i=1}^{a} \frac{x_i^2}{n_i} - \frac{x^2}{n}$
$SS_{Error} = SS_T - SS_{Treatment}$	$MS_{Treatment} = SS_{Treatment} / (a-1)$
$MS_{Error} = SS_{Error} / (N-a)$	

N is the total number of measurements

Criterion

Reject H_0 if $F0 > F_{\propto, a-1, N-a}$. Here $F_{\propto, f1, f2}$ is the upper $\propto$ percentage point of the F distribution with f1 and f2 degrees of freedom.

Table 2-7: An example of an ANOVA test

2.5.5 Graphical visualization

Graphical visualization describes a data set of quantitative measures. Graphs are very illustrative and give a good overview of the data set. There are several types of graph it can be use:

Scatter plot

Scatter plot is one simple but effective graph, where pair wise samples (x_i, y_i) are plotted in tow dimension, as shown below:

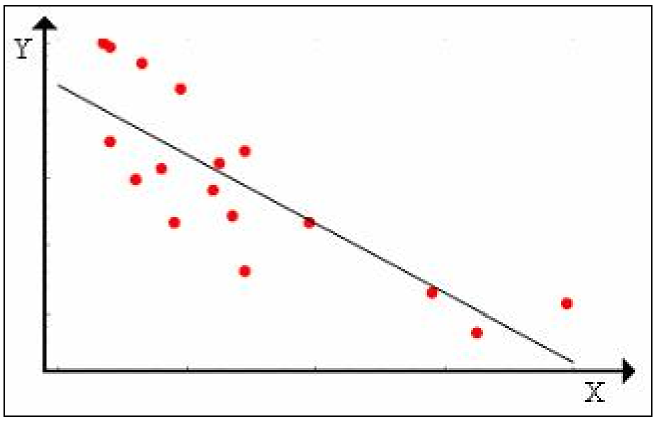


Figure 2-6: An example of scatter plot

Scatter plots are similar to line graphs in that they use horizontal and vertical axes to plot data points Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation. Scatter plots usually consist of a large body of data. The closer the data points come when plotted to making a straight line, the higher the correlation between the two variables, or the stronger the relationship.

If the data points make a straight line going from the origin out to high x- and y-values, then the variables are said to have a positive correlation. If the line goes from a high-value on the y-axis down to a high-value on the x-axis, the variables have a negative correlation.

Line graph

Line graphs compare two variables. Each variable is plotted along an axis. A line graph has a vertical axis and a horizontal axis. So, for example, if you wanted to graph the height of a ball after you have thrown it, you could put time along the horizontal, or x-axis, and height along the vertical, or y-axis. Some of the strengths of line graphs are that:

1. They are good at showing specific values of data, meaning that given one variable the other can easily be determined.
2. They show trends in data clearly, meaning that they visibly show how one variable is affected by the other as it increases or decreases.
3. They enable the viewer to make predictions about the results of data not yet recorded.

Unfortunately, it is possible to alter the way a line graph appears to make data look a certain way. This is done by either not using consistent scales on the axes, meaning that the value in between each point along the axis may not be the same, or when comparing two graphs using different scales for each. It is important that we all be aware of how graphs can be made to look a certain way, when that might not be the way the data really is.

Here is a line graph, which is illustrated my weight according to the years.

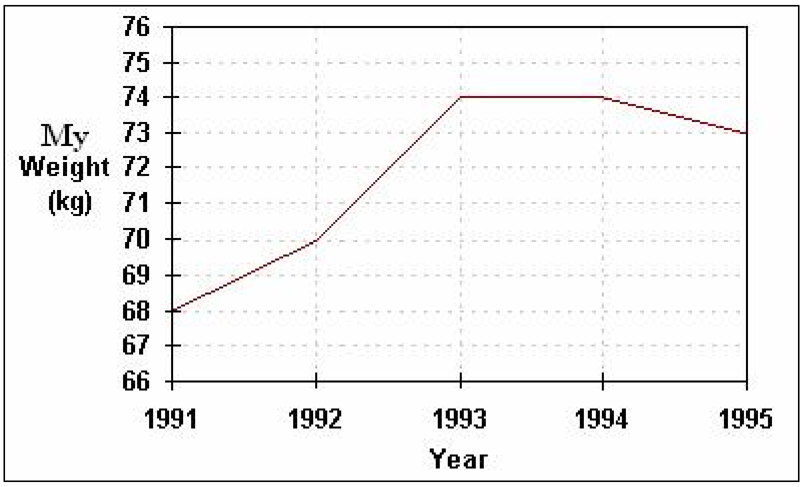


Figure 2-7: Line graph

Histogram

The histogram can be used to give an overview of the distribution density of the samples from one variable. A histogram consists of bars with heights that represent the frequency (or the relative frequency) of a value or an interval of values, as shown below:

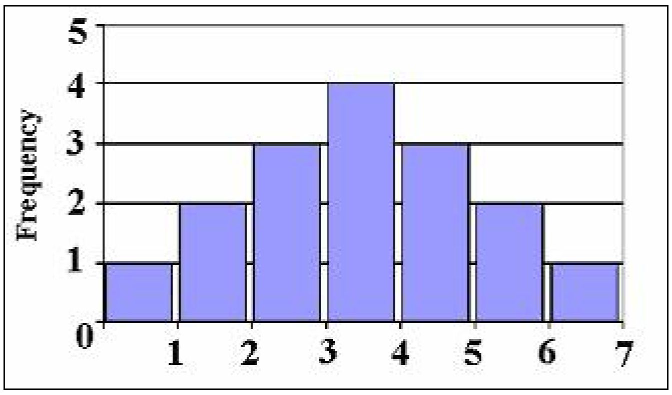


Figure 2-8: An example of histogram

The histogram is thus a graphical representation of a frequency table. One distribution of particular interest is the normal distribution, since it is one aspect that should be taken into account when analyzing the data. Thus, a pot could provide a first indication whether the data resembles a normal distribution or not.

Pie diagram

The pie diagram can also be used to give an overview of he distribution density of the samples from one variable. A pie diagram consists of several pieces, and the size of each pieces presents how large is the part of the totality. The size can be defined in number or in percent.

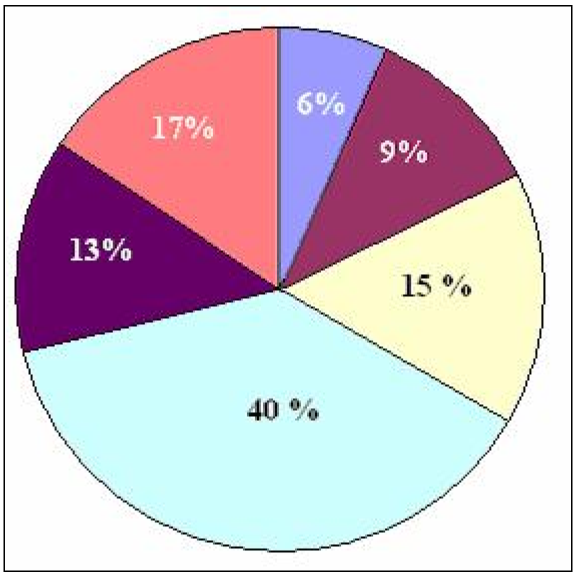


Figure 2-9: An example of pie diagram

3 Statoil context

From our previous report it is written general information about Statoil, and a lot of information about JEF, DCF and O&S Masterplan of Statoil. The information contains the motivation, goal, architecture, and foundation. Here we will add more information about JEF and DCF, and present the new project of Statoil; Shipman & Allocation (S&A). More detail of each project can be found at [01], [11] and [04].

3.1 JEF

[04] JEF version 3.0 is released in September 2005. All components will on utilize the same version numbers for major releases. This should make it easier to figurer out which component version belong together. For example this release will start by using the 3.0 version number. Patch releases for this release will be name 3.0.x (with x being the patch number). JEF version 3.1 was released not a long time ago, and the new version contains more completed data than previous version. JEF version 4.0 will be released later in this year.

JEF contains seven components, which are based on reuse and the components are used in the application of the different projects. Down below shows an illustration of the DCF application, which includes the relationship between JEF components, DCF component and other components.

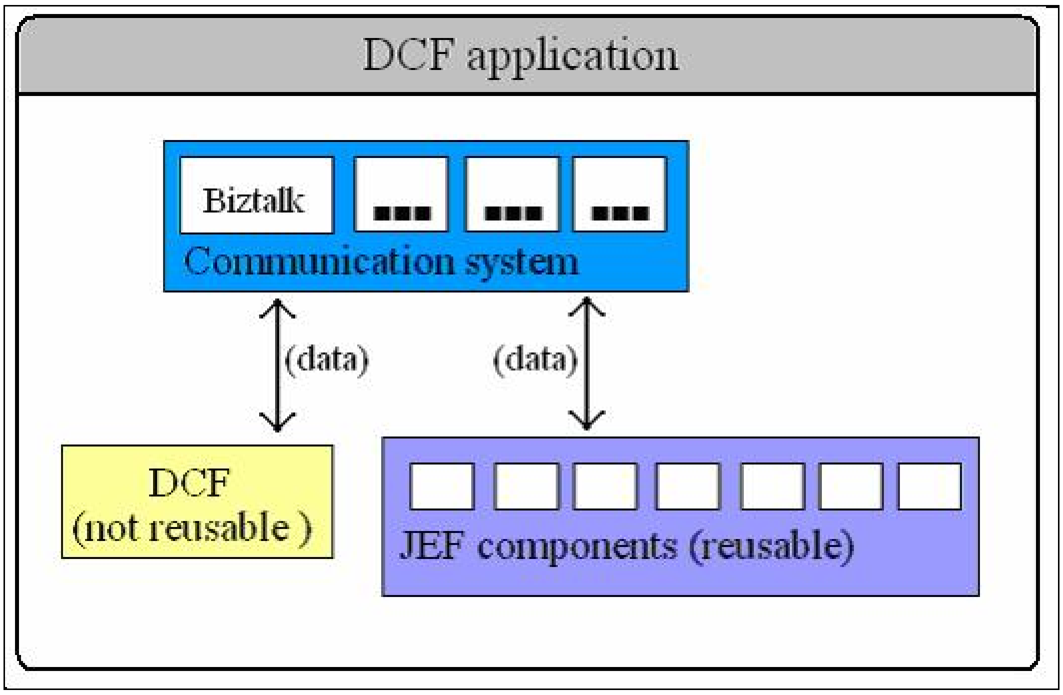


Figure 3-1: Overview of DCF application

3.2 DCF (Digital Cargo Files)

The project DCF was established in year 2004 and it is planned to be finished in September year 2006. The DCF version 2 will be based on the same pre-study that was done for the first part of the project, and most project documents will be updated to the existing documents from the first DCF project. The focus of the project is the same as the DCF1 project: Specification and implementation of a revised filing process and a new system solution. The project will take the deliverables from the DCF1 project as a starting point, and complete the remaining technical and organizational scope not covered by the DCF1 project.

The project will have two deliverables:

Date	Description
Within 2005	DCFv1.1 will be delivered, and will extend the DCF1 system for use in PRO. The deliverable will also cover functionality to support paper deals
July 2006	DCFv2.0 will be delivered, and will cover the remaining technical and organizational scope. Experiences from the acceptance test of DCF1 can be incorporated in the DCF2 project, but will then be treated as changes to the DCF2 scope.

Table 3-1: Delivery table for DCF

We like to remind the goals of DCF, and the goals refer to the expected achieved effects after the project is completed.

Goal ID	Description
G1	Efficient and common work process for filing of deal and cargo documents extended to all O&S business units in addition to NGL which is already covered by DCFv1.0.
G2	Efficient and common work process for filing of deal and cargo documents extended to all O&S business units in addition to NGL which is already covered by DCFv1.0.
G3	Complete cargo files.
G4	Reduced filing time by 50% for all new business units covered by the project.
G5	Reduced retrieval time by 50% for all new business units covered by the project.
G6	User satisfaction to be rated as min. 4.5 for all new business units covered by the project.
G7	O&S Management ownership of the filing solution for all new business units covered by the project: - Supported by management - Mandatory work process for all O&S organization units - Determination among management when implementing - Required used by all relevant users

Table 3-2: The goals of DCF

3.2.1 System overview of DCF

[01] The Wet Supply Chain (WSC) is the part of the Statoil business that is the responsibility of the Oil Trading and Supply unit (O&S) within the Refining and Marketing division of Statoil ASA. The units focus is marketing and trading the crude oil, liquid natural gas and refined products either directly from the field or from refineries and processing plants to customers worldwide.

A number of supporting systems are in place to support the unit, and the main bulk of them have been developed in-house in the period 1992 – 2001. In addition there are a number of externally developed systems that are integrated with these home-grown systems. Central in the WSC system portfolio is RATS (Reports and Trades System), and SPORT (Statoil Planning and operational system, for Offshore and Refinery Terminals), which are used for handling the specialized trades on 21 Day BFO cargos and the similar Dubai trades in the middle-eastern / Asian markets.

SPORT is a planning and scheduling system for offshore and refinery terminals, scheduling the cargoes that are to be delivered from the offshore field or terminal. It is also used to manage supply operations.

In addition a number of auxiliary systems have been integrated with these two central systems to support the work processes. SAP is used for invoicing and is also the master for all counterparty and material (products and qualities) static data. OCD (O&S Common Data) is used as a broker between SAP and the business applications for transforming and extending the master data into the forms used by the business applications. Message Manager is used for communication, in particular sending telexes, which is the format used for sending and receiving contracts. CRBA (Common Role Based Access) is used to define user access to functions in the systems through the use of roles.

The Digital Cargo Files (DCF) system is part of the TOPS system portfolio, which is built to replace the existing systems supporting the O&S WSC organization. The TOPS portfolio today consists of web-based systems for generating contracts and verifying deals, and a "rich client" system for easier entry of physical deals and deal maintenance. The DCF system will be delivered as an integrated part of the TOPS portfolio, specifically the "rich client" part.

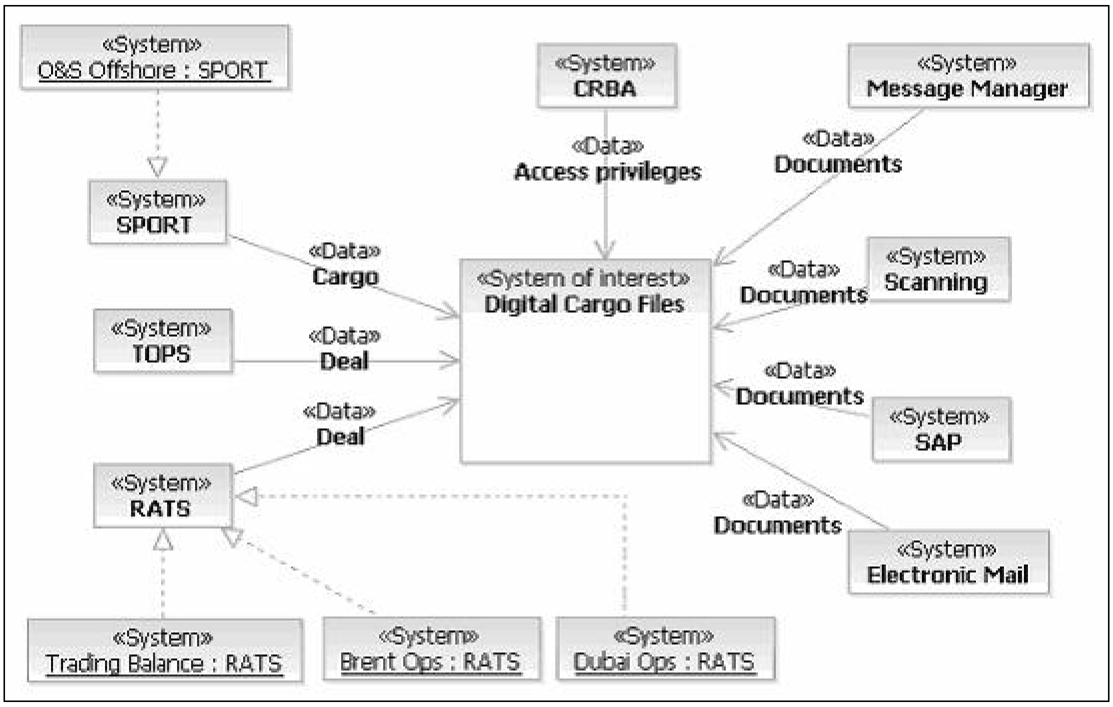


Figure 3-2: Overview of communication channels in DCF

3.3 S&A (Shipment and Allocation)

[11] Statoil's allocation, developed in 2003 – 2004, is an allocation and tariff system for fields, refinery and terminals. The main functionality of this system is to allocate produced quantities, do stock accounting and calculate tariff. The first version of S&A is released in Mars 2006.

The figure below illustrates the to-be context map of S&A:

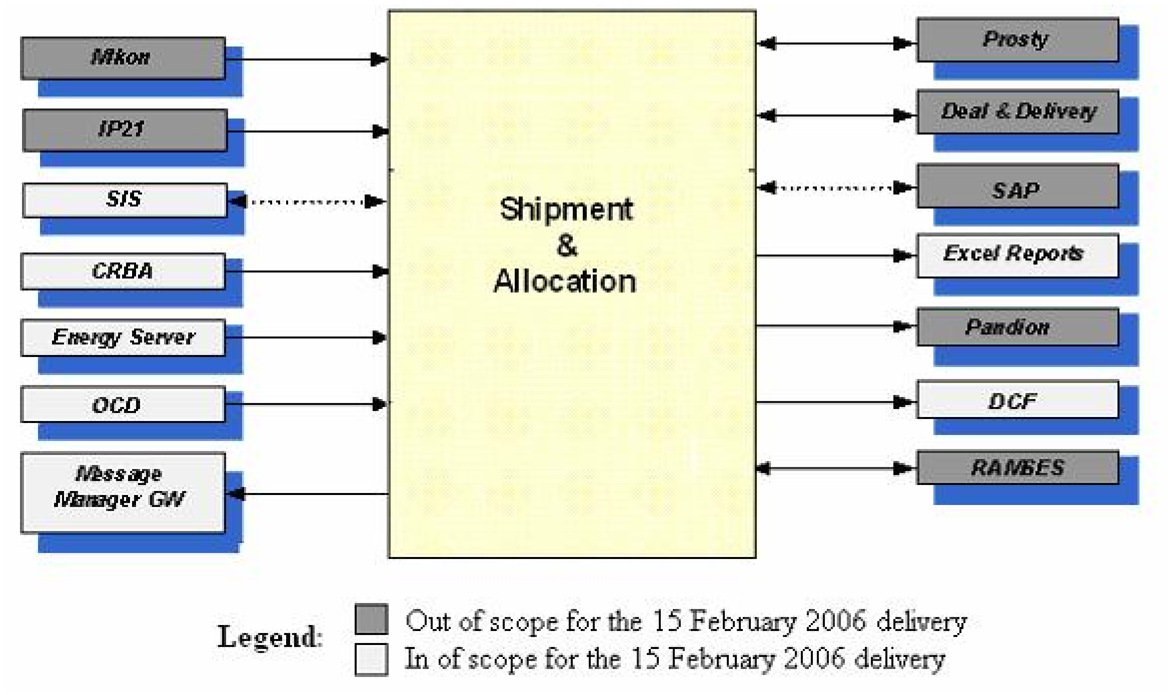


Figure 3-3: The context map of S&A

3.3.1 S&A project goals

The project goals listed below are related to the total project, i.e. improving business processes, gathering systems requirements and implementing new IT-solutions. These project goals give the rationale and constraints for the system architecture. The project goals are:

Goal	Description
S&A's G1	More efficient and controllable business processes through common business principles within lift planning and cargo planning.
S&A's G2	More efficient and controllable business processes through common business principles within lift planning and cargo planning.
S&A's G3	More efficient and controllable business processes by establishing common ways to interact with processes outside WSC.
S&A's G4	Less time and effort spent on input and transfer of data with IT-solutions supporting input nearest possible the source of the data, and most of data transfer done electronically.
S&A's G5	Greater profit by earlier incorporating products into value, more sales approach into lifting principle.
S&A's G6	More beneficial business processes with new processes and IT-solutions according to the terminal principle regarding lift scheduling and cargo planning, possible reallocation of recourses to more important areas like problem solving and control activities.
S&A's G7	Reduced cost/reduced demurrage on transportation by optimising the logistic procedure for loading/unloading and incorporating this into the new IT-solutions.
S&A's G8	Less time spent on creating reports.

Table 3-3: The goals of S&A

3.3.2 System overview of S&A

The roles that are used in the specification that are specializations of the general roles are:

Role	Description
SOL operator	An operator that works on Statoil Operated Licenses SOL.
Port operator:	An operator that works with functions related to port operations.
Lab operator	An operator that works on or related to a laboratory.

Table 3-4: Description of roles in S&A

Here is a summarized list of which roles are using the system:

Role\Installation	Offshore	VP	Troll/Heidrun	Snøhvit
Operator	X	X	X	X
External				
SOL Operator	X	X	X	X
Port Operator	X			X
Laboratory Operator	X			X

Table 3-5: The roles in S&A

3.4 The available data resources

This sub-chapter will give a detailed explanation of the available data resources from Statoil. We have gathered this information from the documentation from Statoil, and interpreted it with help from Anita.

3.4.1 Lines-of-Code

We will be using the physical number of LOC (lines-of-code). All the LOC's were given to us by Anita.

3.4.2 Cost

The cost of a project will be the cost of developing it. These values have been gathered from the handbooks of respective projects and directly from project supervisors. The costs will be using the Norwegian currency, NOK.

3.4.3 Defects

We have been given defects on the projects DCF, S&A and JEF. These reports have been using the term *incident*. An incident can be an error or some other groups (this will be mentioned later). For this thesis, we will be using the term *incident* instead of *defect*, but the sub-groups will remain the same. We wish to make a note on how we have counted the incidents. Originally, one incident could have multiple instances of the same ID because of the progressions of its state. We have chosen to use only unique IDs because we will not use the attribute "state", and it will be a lot fewer incidents to work from. For example, DCF version 2.0 has 11705 lines of incidents, but only 880 of them have unique IDs. The following table is an overview of the attributes of incidents:

Attribute	Description
ID	A unique identifier for the incident. JEF incidents have JEFXXXX, S&A incidents have SAXXXX and DCF incidents have DCFXXXX.
Headline	A textual description of the incident.
Owner	The owner of the incident, or the one responsible.
State	The state of the incidents shows where incident is in a workflow cycle as a specific time (see figures below).
Priority	The level of the priority the incident is given. 0 is critical, 1 is high, 2 is medium, 3 is low and 4 is not prioritized.
Severity	The level of severity the incident is given. 1 is critical, 2 is high, 3 is medium and 4 is low.
Classification	The classification of incidents indicates whether the incident is an error, enhancement or duplicate.

Table 3-6: Overview of incident attributes

The following sections will give a detailed description of each of the most important attributes.

Severity of incidents
The severity of incidents tells us how much impact the incident has on the system, and how important it is to have it fixed. The following explanation for each degree of severity is taken from the test plan:

Severity	Detail
Critical	The defect causes system crash, loss or corruption of data. The defect stops the testing and the defect must be corrected or the change request must be implemented before the system can work properly. A defect with this grade could also mean that the system do not fulfill critical business functionality or will disrupt other systems.
High	The defect means loss of a part of required functionality or quality. Alternatively, the defect stops the current test or has impact on tests concerning interface units.
Medium	The defect could mean loss of part of required functionality or quality, but there exist ways to work around the problems. The defects have only impact on this test, and in limited degree on the test progress. Correcting the defect will add value to the business like in time and/or costs.
Low	The defect may only be cosmetic and has no impact of importance on the functionality and quality. The defect does not stop the test and does not block the test progress.

Table 3-7: The incident attribute 'severity'

Priority of incidents
The priority of incidents indicates which incidents are the most important to fix. Critical (or 1) is the most important, while low (or 4) are the least important.

Classification of incidents

The JEF development project has defined five types of incidents.

Classification	Detail
Error	The same as defect or a fault. The system does not follow the specification. Error in other system the error will be corrected in relation to that system.
Duplicate	The incident is a duplicate of another, and will be corrected as part of another incident.
Rejected	The incident is not approved to be worked on, either because it was not verified or it was not classified as en incident.
Postponed	The incident is set on hold, as it needs more analyze or will be continued on later.
Enhancement	Applies as a change. Enhancements could be moved to changes, but most of them are corrected as errors.
Not reproducible	The incident was not reproducible. It will stay in this classification until it is rejected or a new occurrence appears.

Table 3-8: The incident attribute 'classification'

State of incidents

The state of the incident tells us how far the incident has come in its life cycle. There are two life cycles; one for an enhancement or change request, and one for an error [07]. The following figures show the life cycles of an enhancement or change request.

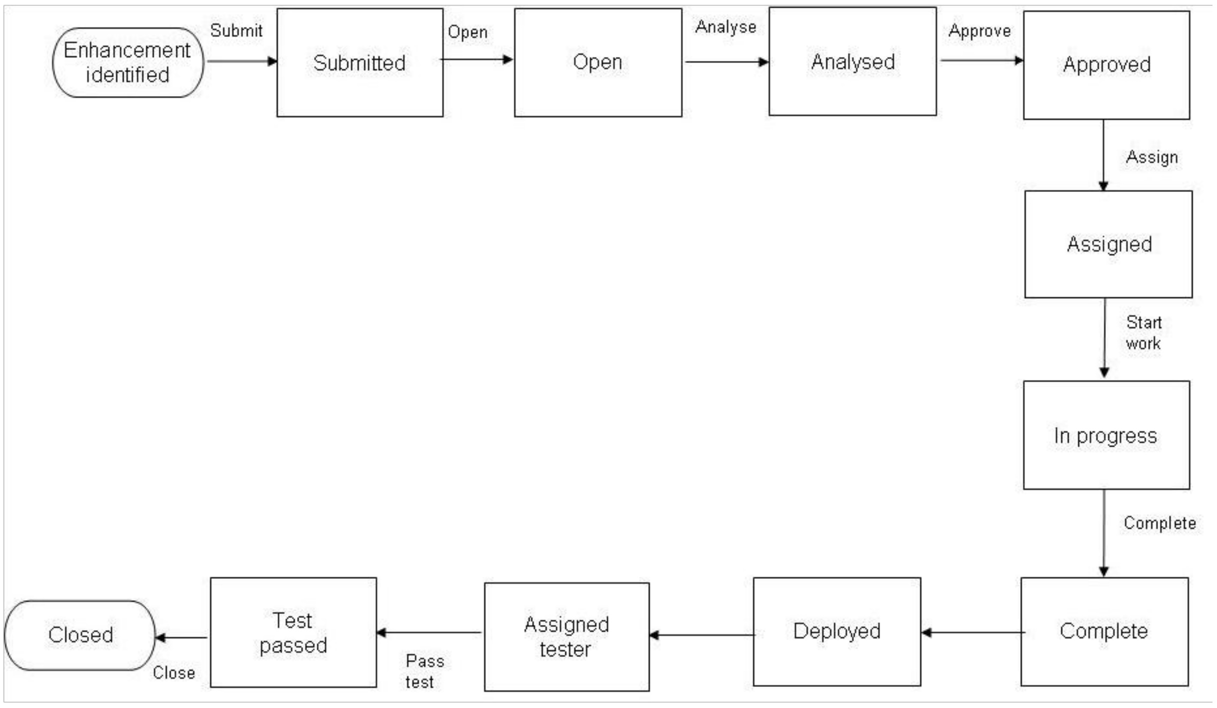


Figure 3-4: State of an enhancement or change request

The following figures show the life cycles of an error.

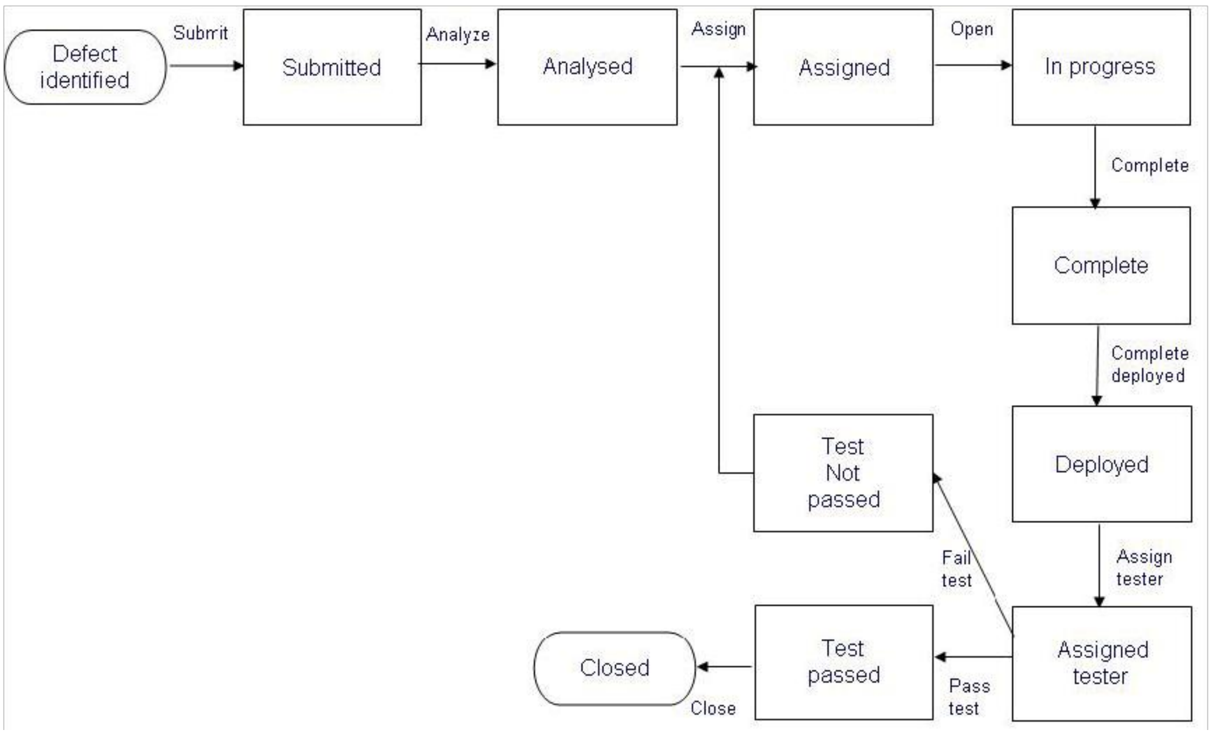


Figure 3-5: State of an error

3.4.4 Change request

When a developer identifies an error or a need enhancement, they file it as a change request (CR). These requests are stored in structured files for further inspection before the next release. CR has a number of attributes:

Attributes	Detail
CQ ID	The unique ID of each CR.
Scope	The scope of the CR. The scope is version of the project in which the CR is based on.
Headline	A textual description of the CR.
Priority	The priority given to the CR.
State	The state of the incidents shows where incident is in a workflow cycle as a specific time (see figures above).
Owner	The owner or the one having the responsibility for the CR.

4 Research focus and problem elaboration

The research questions in our study are the impact of reuse component based on architecture JEF. To address those research questions, we have to choose some specification to aim. Based on the literature search, previous studies and a pre-study of the available data, we chose to focus on effort, defect density and quality attribute of stability in the case study.

Subchapter 4.1 gives an overview of the issues, which has to be considerated. This will narrow our focus and enable us to specify hypotheses.

Subchapter 4.2 will present two hypotheses based on the considerations in last subchapter.

Subchapter 4.3 will elaborate on which methods, which will be used to investigate the hypotheses.

4.1 Consideration

Software reuse has become a topic of much interest in the software community due to its potential benefits, which include increased product quality and decreased product cost and schedule. The most substantial benefits derive from a product line approach, where a common set of reusable software assets act as a base for subsequent similar products in a given functional domain. The upfront investments required for software reuse are considerable, and need to be duly considered prior to attempting a software reuse initiative.

From a business' point of view, the purpose of reuse is primary shorter time to market and cheaper than defining from scratch. A good software reuse process facilitates the increase of productivity, quality, interoperability and reliability, and the decrease of costs and implementation time. An initial investment is required to start a software reuse process, but that investment pays for itself in a few reuses. In short, the development of a reuse process and repository produces a base of knowledge that improves in quality after every reuse, minimizing the amount of development work required for future projects, and ultimately reducing the risk of new projects that are based on repository knowledge.

Any organization of a development processes needs to understand its place in the life cycle. Each release version in a project has to be improved, upgraded and maintained from the previous version. The needed improvement comes from the desires of the users or customers. After the improvement, it releases a new version and it will be evaluated again from the users or customers. The circle circulates again and again, until the product is perfect or the dead line is expired. This circle is calling Evolution Delivery Life Cycle [06].

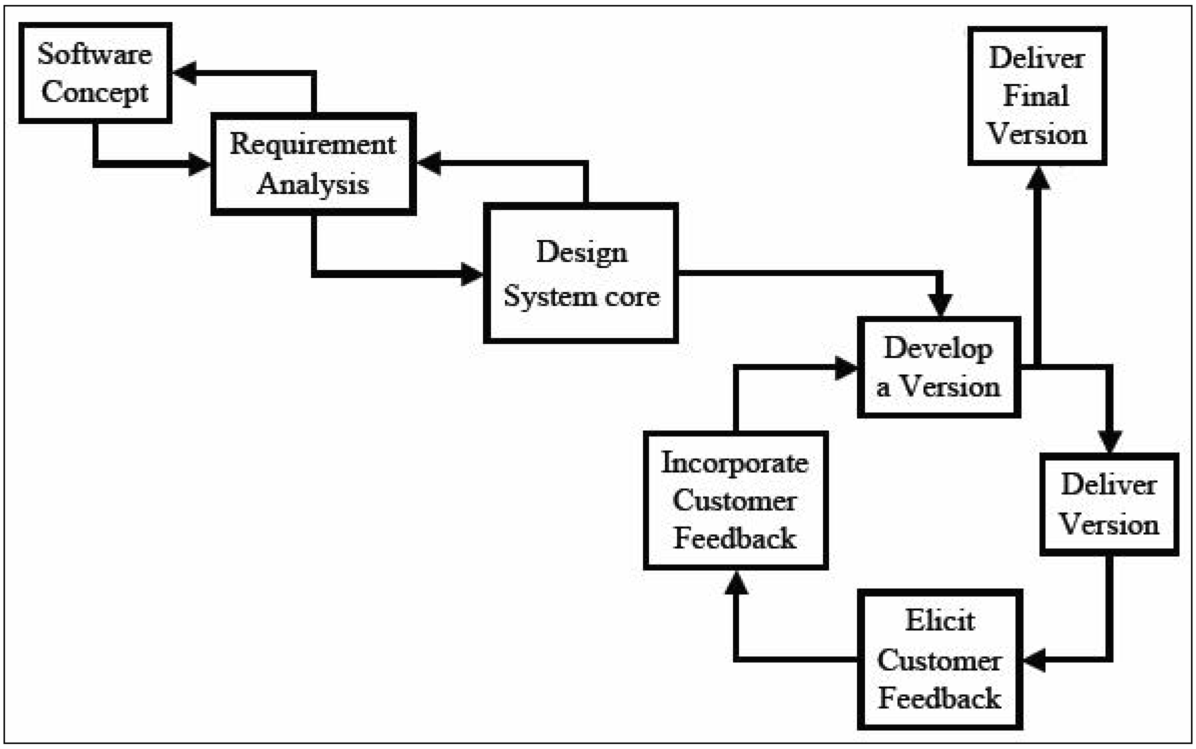


Figure 4-1: Project evolution delivery life cycle

The consideration of the thesis is based on Statoil's use JEF, and it raises some very intriguing issues. We will make a research to see whether or not JEF has made a considerable positive impact on the overall business process. If this is the case, we want to investigate if this impact can be justified with the cost of deploying, maintaining and future developing of JEF. The impact of JEF can also be measured in qualitative metrics, such as effort, defect density and quality of stability.

The resources has been used to explore these issues is the existing and available reports of variant projects, which are using the architecture JEF. The exploration occurs to compare those reports, such as test document, project document and project handbook. The comparison is made from different projects and different versions in the same project, and it will consider measurements such as time, cost, incident and quality. Figures below illustrate the attention in our exploration.

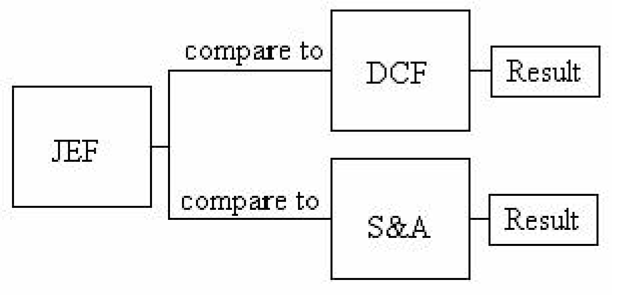
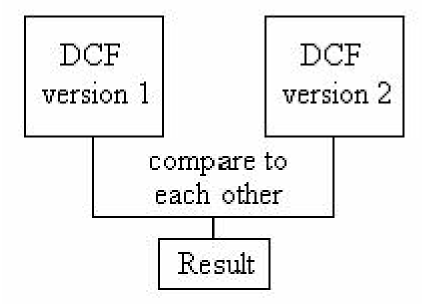
Variant projects	Variant version of same project
 JEF compare to DCF Result compare to S&A Result	 DCF version 1 DCF version 2 compare to each other Result

Table 4-1: The area of attention

4.2 Problem definition and research questions

Here we are going to define the hypotheses and sub-questions of each hypothesis. The answers to the sub-questions are supported or rejected the hypothesis.

4.2.1 Effort definition to hypothesis

The purpose to reuse existing components instead to implement the new one is obviously to make lower costs and a higher effort.

Hypothesis of effort	
H_E: The effort in DCF and S&A is lower than the effort in JEF.	
Question ID	Question
QE-1	What is the effort in DCF?
QE-2	What is the effort in S&A?
QE-3	What is the effort in JEF?

Table 4-2: Hypothesis of effort

4.2.2 Defect density definition to hypothesis

The defect density is an important measure of software quality, which is widely used in industry. It is often one of the measures used to ascertain release readiness. There are two factors that control the defect density at release. One of them is the extent and effectiveness of the testing and debugging effort [05]. The other is the initial defect density present at the beginning of testing [03]. Defect usually occurs by implementation of new function and this make the product more attractive to users, but may not at the expense of an increase in residual defect density. Defects can be introduced in several activities of the software development life cycle and are classified based on their origin. The type of defect artifacts of interest for this thesis is coding defects. Coding defect includes primary codes, functions and unexpected defect such as exception error. The classification of the defect considers how the defect is to be repair, and in chapter 2.2 is clearly defined the various classification.

Hypothesis of defect density	
H_D: The defect density in DCF and S&A is lower than the defect density in JEF.	
Question ID	Question
QD-1	What is the defect density in DCF?
QD-2	What is the defect density in S&A?
QD-3	What is the defect density in JEF?
QD-4	If the defect density in DCF and S&A is higher than the defect density in JEF, does priority (critical, high, medium, low) to correct the incident make any difference?

Table 4-3: Hypothesis of defect density

4.2.3 Stability definition to hypothesis

The world around software systems is constantly changing, and the software system has to upgrade, to be suitable to the environment. For this reason most software systems need to be modified many times after their first release. Another reason for modifying software is when software products are updated and improved to keep the competitive advantage against other products on the market. The result is software products that evolve from release 1, to release 2, to release 3, and so on. The software life cycle of a product is continuous development that only ends when the product is obsolete.

The changes that are implemented from release to release include anticipated changes, such as following the market plan, and other unanticipated ones. Additionally, the bug fixes for the previous release are usually incorporated into the code baseline. The goal is to increase the productivity in the subsequent release by choosing the most appropriate design solution from the start. Consequently, stakeholders are generally interested in a system designed such that future changes will be relatively easy to implement, and thus increase the maintenance productivity for implementing these changes. What to be changed in an application tells what is the quality of the different quality attributes. Here we are just focusing on the stability. The stability of an application is quite important, because stability gives the trust and safety to the users when they use the application.

Hypothesis of stability	
H_S: The stability in project that has been developed from scratch (JEF) is lower than the stability in the projects that base on reusable components (DCF).	
Question ID	Question
QS-1	What is the number of change request in DCF?

Table 4-4: Hypothesis of stability

4.3 Method

Here we are going to define our research strategy variables that are going to solve the problems and determine if the hypotheses are true or false. There are three major variant types of empirical investigation, such as survey, case study and experiment. In our situation we do not have direct contact to developers or employees in Statoil and that is why it is not possible to carry out the investigations. We received data reports of Statoil's projects from scholarship recipients who have contact to Statoil's technical department. In this way we have devised our own methods and used existing metrics to carry out the evaluations. The Following sub-chapter will illustrate our own methods as well as existing metrics.

4.3.1 The effort metric

The formula below presents the effort. It introduces the total number of cost divides by the total number source line of code in the same project.

$$\boxed{\text{Effort} = \frac{\sum \text{Cost}}{\sum \text{SLOC}}}$$

Equation 4-1: Metric for effort

4.3.2 The defect density metric

The most commonly used means of measuring quality of a piece of software code is the defect density metric. That metric is defined by:

$$\boxed{\text{Defect density} = \frac{\sum \text{Defect}}{\sum \text{SLOC}}}$$

Equation 4-2: Metric for defect density

The size of code is normally defined as LOC (Lines-of-code) or KLOC (thousand lines-of-code).

4.3.3 The stability metric

We are going to use the same method to declare the stability that has been used by scholarship recipients. By our understanding of the method, we can use the number of change request from a project to compare to the number of change request from another project. When project A gets lower change request than project B ($A_{cr} < B_{cr}$) it tells us that project A has higher stability than project B. The same method can also use to check the stability between the various versions in the same project ($A_{1dd} < A_{2dd}$).

5 Data resources

This chapter will introduce the data resources we will be using in our empirical studies. Most of the data has been gathered from Statoil through Anita. Microsoft Excel has been used to visualize the data.

Subchapter 5.1 will present the data resources from project DCF in detail.

Subchapter 5.2 will present the data resources from project S&A in detail.

Subchapter 5.3 will present the data resources from project JEF in detail.

5.1 JEF

We will use only the newest versions of JEF releases. This means we will use the following versions: JEF 2.9, JEF 3.0 and JEF 3.1. Version 3.2 is under development.

5.1.1 Lines-of-Code

The following table shows the size of each component in each of the releases. The size is measured in LOC (lines-of-code).

Component	Release 2.9	Release 3.0	Release 3.1	Release 3.2
JEFClient	7 871	8 400	8 885	8 885
JEFWorkbench	4 187	4 515	4 748	4 748
JEFSecurity	1 588	1 593	2 374	2 374
JEFUtil	1 312	1 359	1 647	1 647
JEFIntegration	958	0 958	958	958
JEFDataaccess	181	0 181	268	268
JEFSessionMgmt	778	1 593	1 468	1 468
Sum	16 875	18 599	20 348	20 348

Table 5-1: Overview of KLOC of components in JEF

5.1.2 Cost

This table shows the budgeted costs of JEF through the different versions. The costs are in NOK.

JEF version	Budgeted cost
2.9	25 400 000
3.0	27 100 000
3.1	28 400 000

Table 5-2: Budgeted costs of JEF

5.1.3 Incidents

The following table shows the number of defects in each of the component in each of the releases.

Component	Release 2.9	Release 3.0	Release 3.1	Release 3.2	Sum
JEFClient	136	13	2	0	151
JEFWorkbench	21	4	1	2	28
JEFSecurity	9	1	0	1	11
JEFIntegration	7	0	0	0	7
JEFSessionMgmt	2	0	1	1	4
JEFDataAccess	2	0	0	0	2
JEFUtil	2	0	0	0	2
General	18	4	0	0	22
Sum	197	22	4	4	227

Table 5-3: Overview of incidents of components in JEF

The following sections show the frequency of different attributes of the incidents through all the versions.

Severity

Severity was not documented for the versions of JEF in question.

Priority

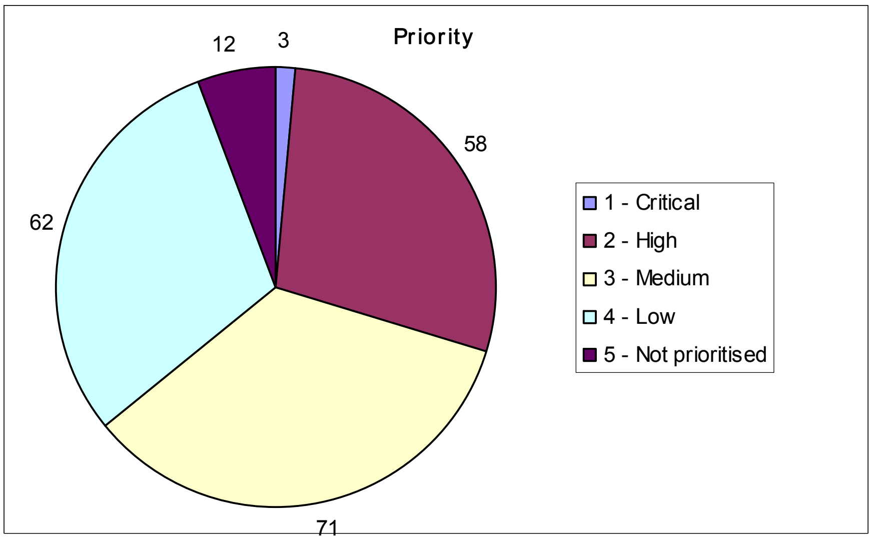
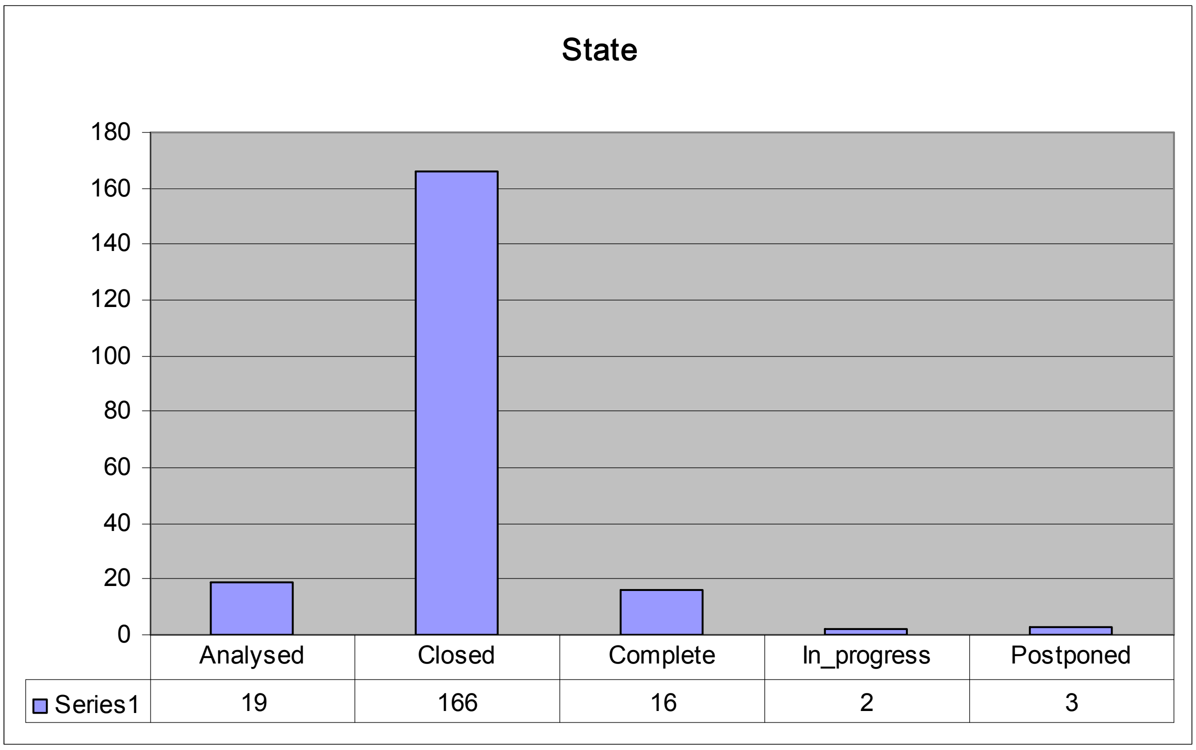


Figure 5-1: Frequency of priorities – JEF

A large percent of the incidents were given a "Not Prioritized" description. Since they were not blanked, we assume that they were deemed not important enough to place a priority on them.

State



	Analysed	Closed	Complete	In_progress	Postponed
Series1	19	166	16	2	3

Figure 5-2: Frequency of states – JEF

Classification

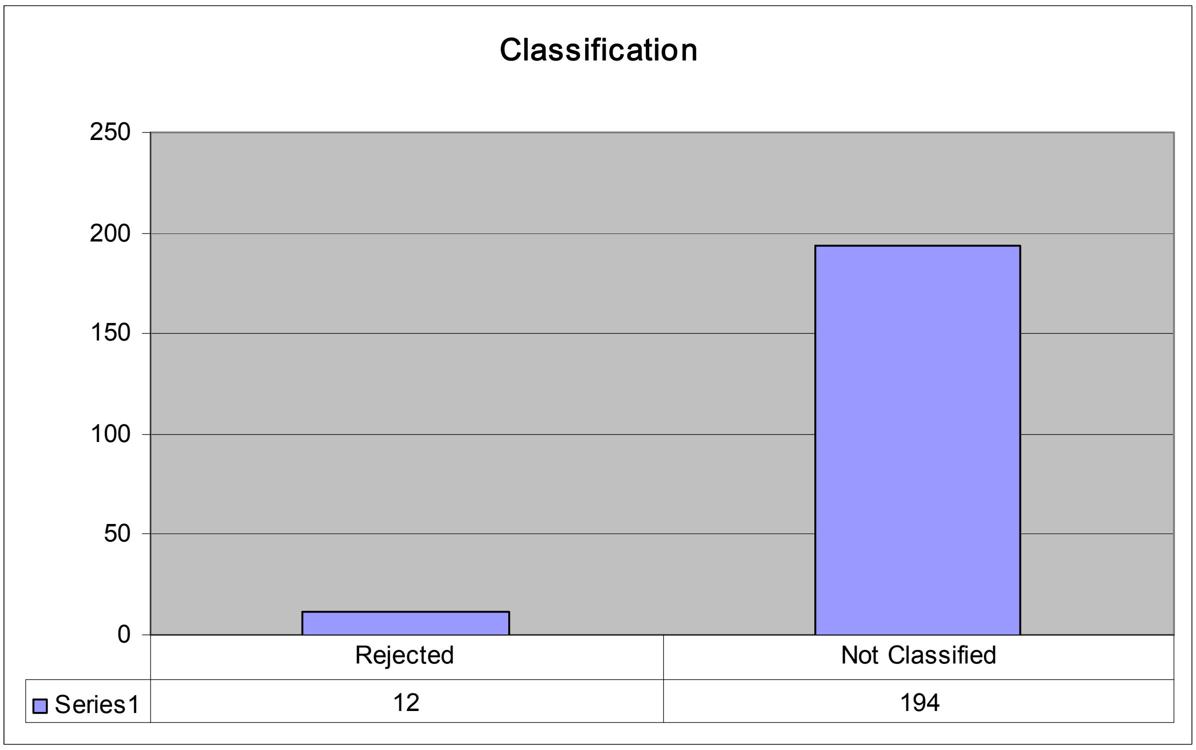


Figure 5-3: Frequency of classifications – JEF

As we can see, almost none of the incidents were given a classification. This attribute is not an important aspect of our study, so we did not investigate this matter further.

5.1.4 Change requests

This data was is not available.

5.2 DCF

We will use the following versions of DCF: Version 1.0, Version 1.1 and version 2.0

5.2.1 Lines-of-Code

The following table shows the lines-of-code for each of the versions.

Version	LOC
1.0	20 702
1.1	21 459
2.0	25 072

Table 5-4: Overview of sizes – DCF

5.2.2 Cost

The following table shows the budgeted cost of developing DCF. These numbers are taken from the handbooks, and they are in NOK.

Version	Budgeted Cost
1.0	11 400 000
1.1	11 400 000
2.0	15 000 000

Table 5-5: Overview of budgeted cost – DCF

5.2.3 Incidents

The total number of incidents for DCF version 2.0 is 880. The following sections show the frequency of different attributes through all the versions of DCF.

Severity

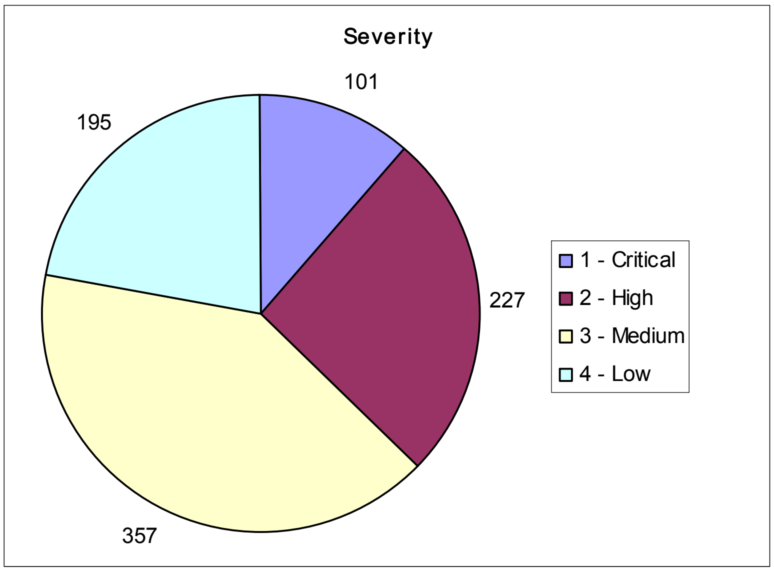


Figure 5-4: Frequency of severities – DCF

Priority

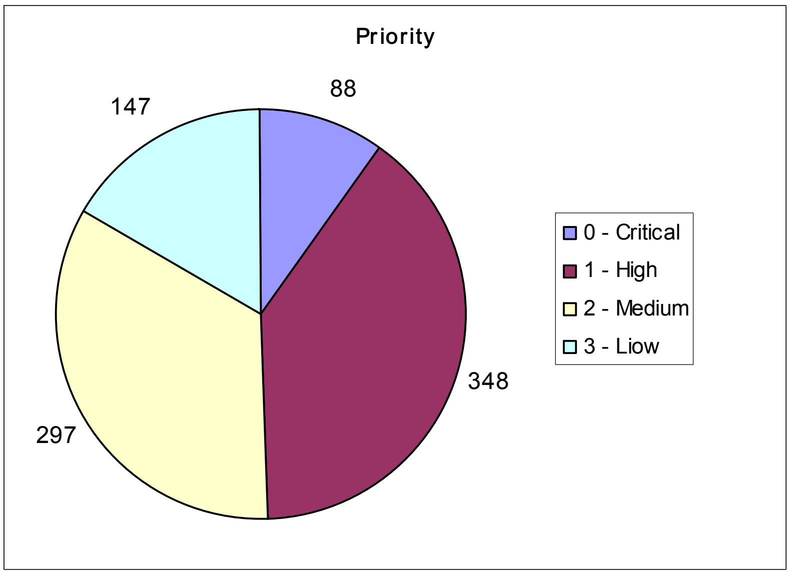


Figure 5-5: Frequency of priorities – DCF

State

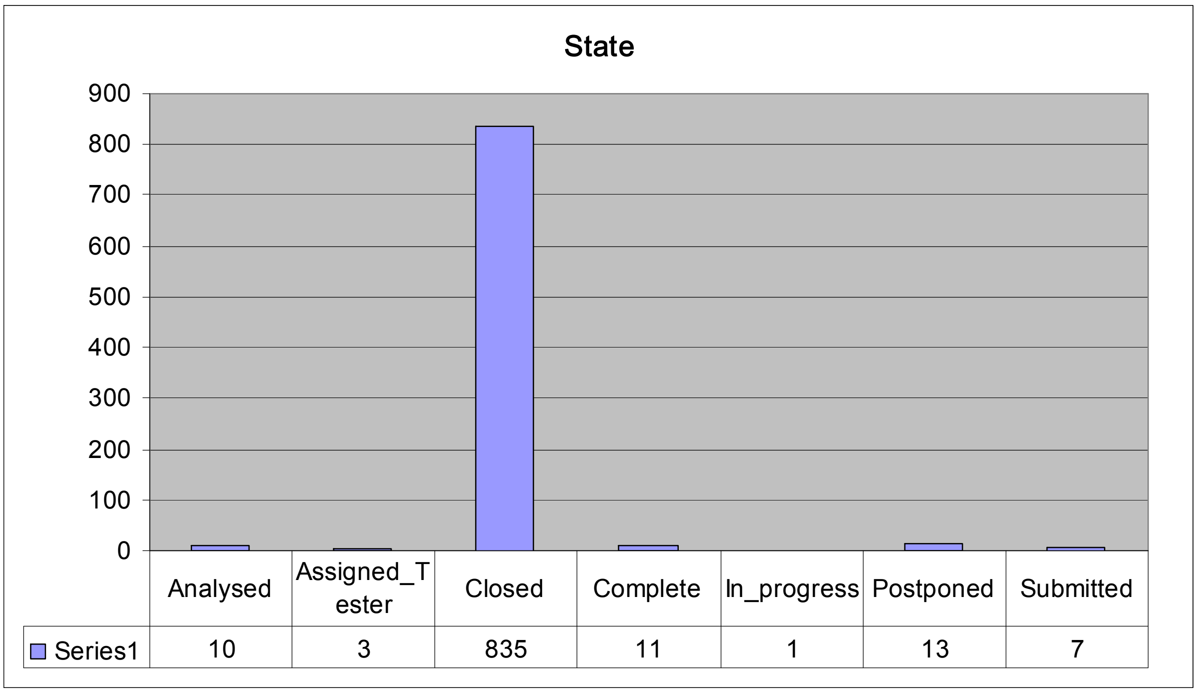


Figure 5-6: Frequency of states – DCF

Classification

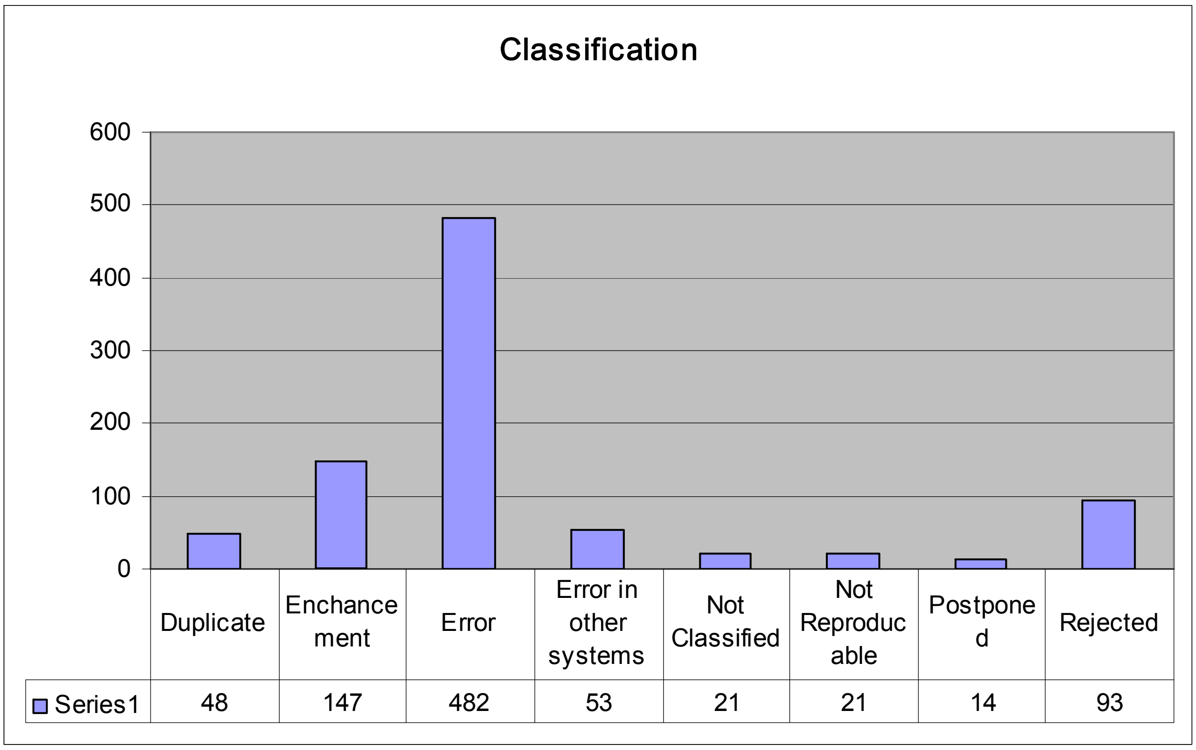


Figure 5-7: Frequency of classifications – DCF

5.2.4 Change requests

There are a total of 96 change requests for the next version of DCF. This is a sample of the change request document for DCF.

CQ id	DCF1.1 scope	DCF2.0 scope	Headline	Priority	State
DCF00000509	X		Change in rules for FilingPlan generation	1. High	Closed
DCF00000982		X	DCF Server to create DELETE operation in FilingPlan	1. High	In_progress
DCF00000994			Common search fields in advanced search	2. Medium	Closed
DCF00001461			Not possible to correct user mistake		Submitted
DCF00001462			Refresh issues on folders when using front end	2. Medium	Closed
DCF00001463		X	Document list: Unable to email documents with BU's common email adress as sender	2. Medium	Closed
DCF00001464			DCF - Use of TD in naming of deal-folders	2. Medium	Closed
DCF00001467			FRONTPAGE DEALFOLDER: Check if SPOT or TERM deal	2. Medium	Closed
DCF00001469		X	ADVANCED SEARCH: Clear Screen button	1. High	Complete

Table 5-6: Sample of change request – DCF

5.3 S&A

Currently, version 1.0 is the only version of S&A. This version is not finished yet, so we will be using the LOC and cost of the current build of S&A instead of using the estimated end-results.

5.3.1 Lines-of-Code

Version 1.0 of S&A has a total of 64 319 LOC.

5.3.2 Cost

The cost of S&A has been collected from the budget document. The following table shows the estimated budget and the actual cost.

ID	Name	Estimate	Used	Remaining
020	Operational/Business spec	1 735 000	1 798 445	0
030	System context	50 000	36 290	0
040	Sourcing and recommended solution	50 000	135 014	0
050	ICT architecture	300 000	276 590	0
060	Functional specification & planning	1 865 000	1 629 280	0
070, 080	Design, build, test	9 000 000	7 810 564	400 000
090	Implementation, test & training	2 640 000	482 066	1 100 000
100	Hand-over	160 000	0	160 000
110	Shut-down	100 000	0	100 000
	Project management	3 000 000	3 321 580	20 000
	Tools and enviroment	500 000	1 021 495	0
	Other	600 000	502 737	20 000
	Sum	20 000 000	17 014 061	1 800 000

Table 5-7: Overview of cost - S&A

We will be using value in the “Used” column to calculate the effort.

5.3.3 Incidents

There are a total of 239 defects in S&A version 1.0. The following sections show the frequency of different attributes through all the versions.

Severity

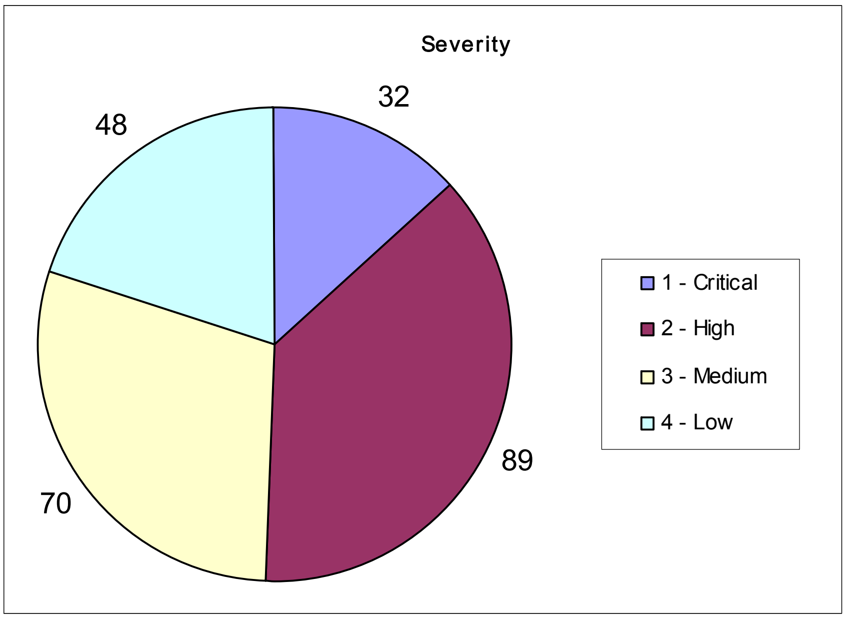


Figure 5-8: Frequency of severities - S&A

Priority

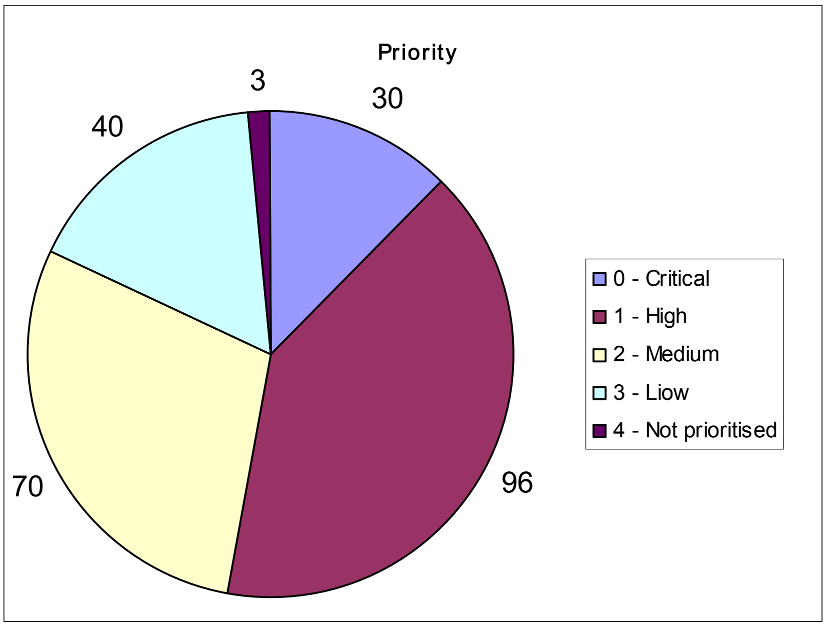
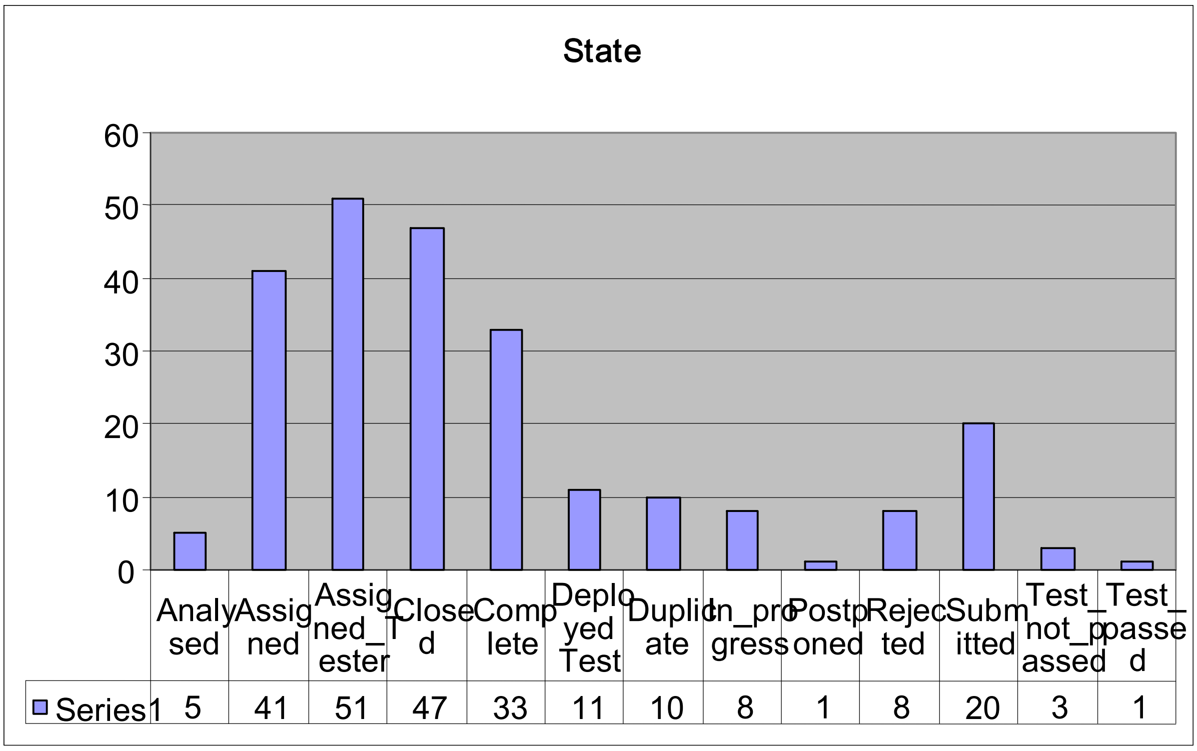


Figure 5-9: Frequency of priorities - S&A

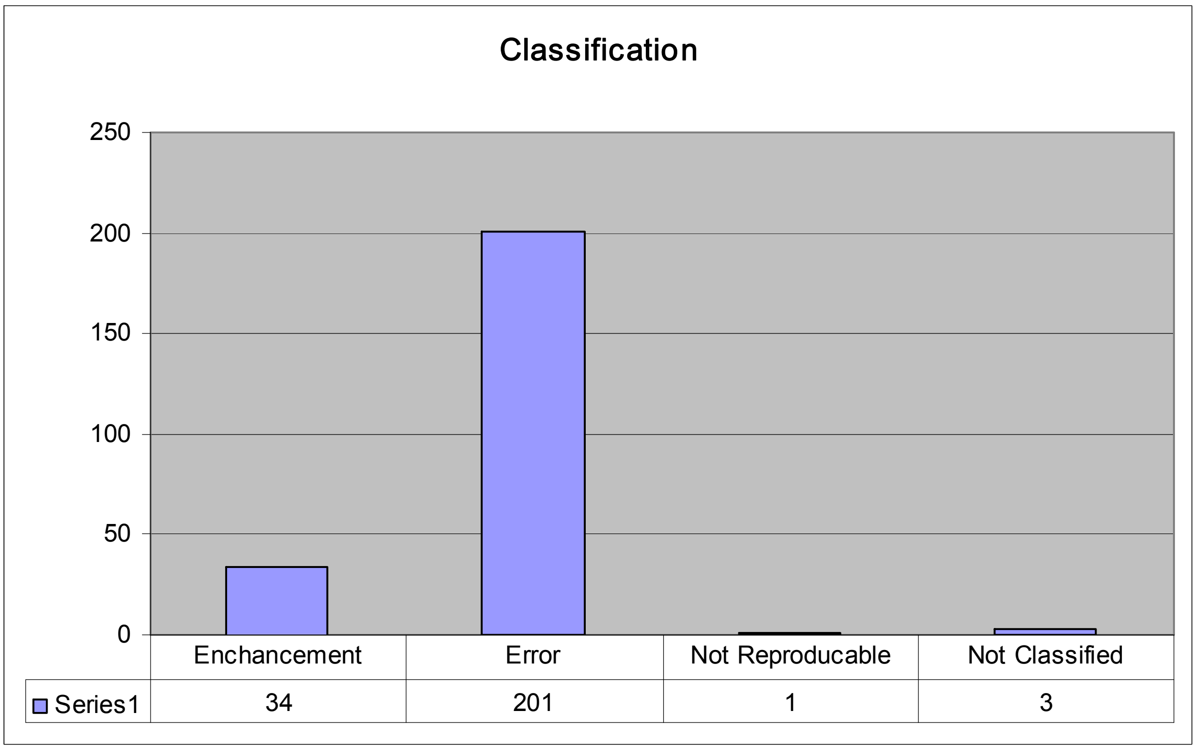
State



	Analysed	Assigned	Assigned_Tester	Closed	Complete	Deployed Test	Duplicate	In_progress	Postponed	Rejected	Submitted	Test_not_passed	Test_passed
□ Series1	5	41	51	47	33	11	10	8	1	8	20	3	1

Figure 5-10: Frequency of states - S&A

Classificaton



	Enchancement	Error	Not Reproducable	Not Classified
□ Series1	34	201	1	3

Figure 5-11: Frequency of classifications - S&A

5.3.4 Change requests

This data was is not available.

6 Results

After collecting data in the operation phase, we want to be able to draw conclusion based on those data. To be able to draw valid conclusions, we must interpret the data. During each subchapter we are going to analyze each hypothesis, which are given in chapter 4, based on the related data resources in chapter 5. We will answer each of the questions during each hypothesis we will answer separately, and in the end of each subchapter we will conclude the hypothesis. During the comparing between the projects and following the different number of releases in the projects, we like to compare three releases from JEF to three releases from DCF, and only one release from S&A to the last release from JEF and DCF.

Subchapter 6.1 presents effort evaluation.

Subchapter 6.2 presents defect density evaluation.

Subchapter 6.2 presents stability evaluation.

6.1 Effort evaluation

To remind the hypothesis of effort is:

"H_E: The effort in DCF and S&A is lower than the effort in JEF."

6.1.1 Question QE-1

"What is the effort in DCF?"

The answer to the question is using data information from table 5-4 and 5-5. The table below presents the effort of DCF according to different versions:

DCF version	LOC	Budgeted Cost	Effort
1.0	20 702	11 400 000	550,7
1.1	21 459	11 400 000	531,2
2.0	25 072	15 000 000	598,3
Average	8 357	5 000 000	199,4

Table 6-1: Effort in DCF

Graphs are a great way to visualize this kind of information. For this reason, graphs are often used in newspapers, magazines and businesses around the world. The following line graph in two dimensions shows the effort in DCF. The data point in the graph is balanced with version of the project version on x-axis and effort on y-axis. There is a line with markers displayed at each data value. This line facilitates to represent the positive, the negative or the neutral movement from the one to another data value.

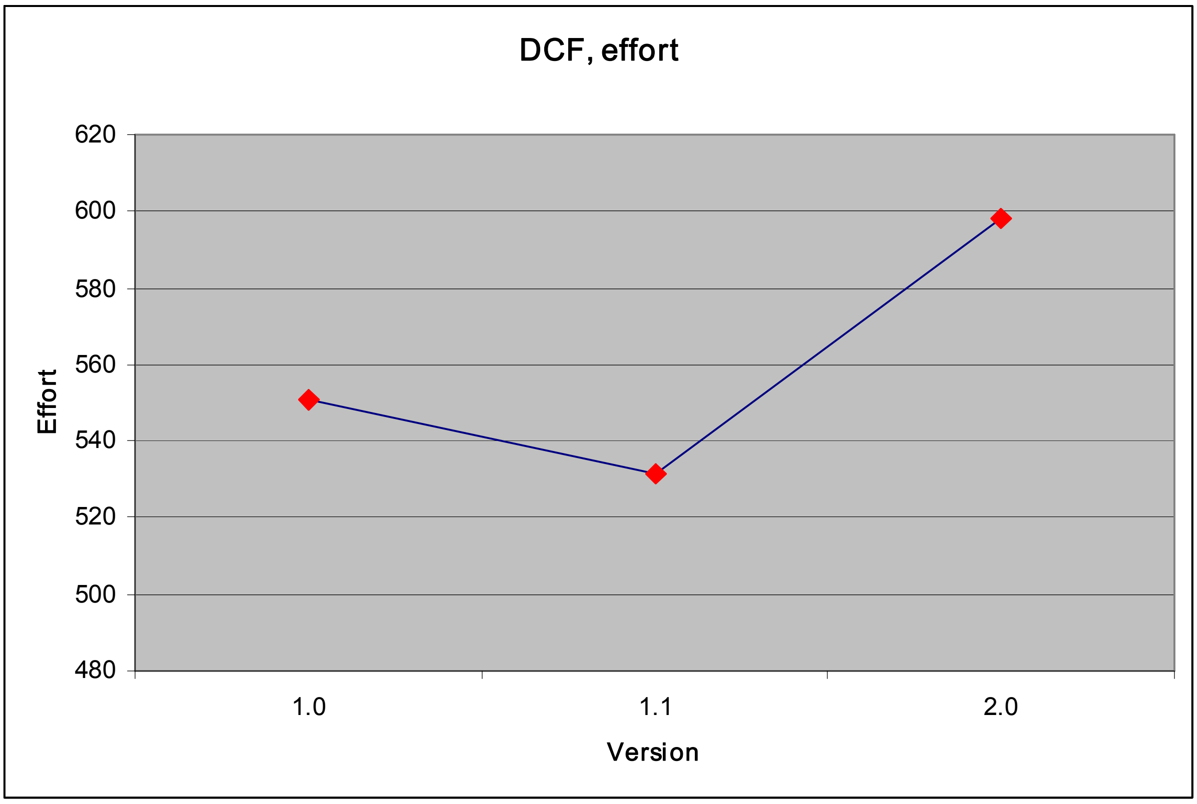


Figure 6-1: Effort in DCF

What we can read from this graph is that from version 1.0 to version 1.1 in DCF, the effort has decreased. This is maybe because of the releases of the version are not far away from each other. The line between those versions is moving in the right direction. As we know when the effort is low, it corresponds to a low cost or a high LOC. From version 1.1 to version 2.0, the effort is increased. The releases of the version are farther way from each other, and according to the table's representing the version 2.0 is added more 3613 LOC. The effort has increased be approximate 12.6% from 1.1 to 2.0. It is hard to determine what the causes are and whether it is a trend or a abnormality since we only have three data points

6.1.2 Question QE-2

"What is the effort in S&A?"

The answer to the question will be using data information from sub-chapter 5.3.1 and table 5.7. The table below presents the effort of S&A. S&A has only been released in one version, thus there is no point to calculate the average and draw a graph. The comparison about this project to another projects can not be done at this moment, and is filed under "Further Work".

S&A version	LOC	Budgeted Cost	Effort
1.0	64 319	17 014 061	264,5

Table 6-2: Effort in S&A

6.1.3 Question QE-3

"What is the effort in JEF?"

The answer to the question will be using data information from table 5-1 and 5-2. The table below is presents the effort of JEF according to different versions:

JEF version	LOC	Budgeted Cost	Effort
2.9	16 875	25 400 000	1505,2
3.0	18 599	27 100 000	1457,1
3.1	20 348	28 400 000	1395,7
Average	6 783	9466667	465,2

Table 6-3: Effort in JEF

The following line graph shows the effort in JEF.

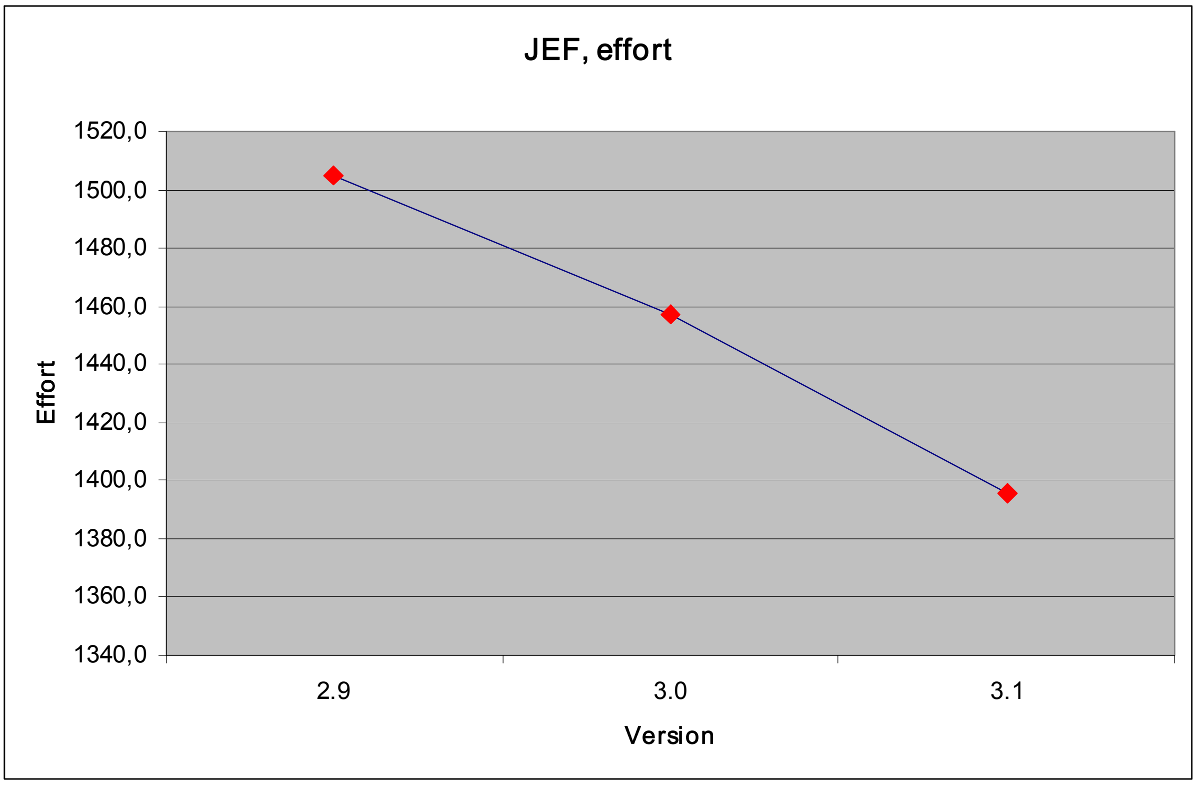


Figure 6-2: Effort in JEF

The graph shows that the line of effort between the various versions is decreasing and moving in the right direction. The line is closely like linear.

6.1.4 Analysis of hypothesis – H_E

Like mentioned in previous chapters we would like to constitute separate comparisons, such as three deliveries from JEF compared to three deliveries form DCF, and only one release from S&A compares to the last release from JEF and DCF. In the end we will draw a conclusion.

JEF VS DCF

The following table and graph presents a overview to facilitate a comparison between the effort in JEF and DCF.

Release	LOC JEF	Effort JEF	LOC DCF	Effort DCF
1	16 875	1505,2	20 702	550,7
2	18 599	1457,1	21 459	531,2
3	20 348	1395,7	25 072	598,3
Average	6 783	465,2	8 357	199,4

Table 6-4: Effort (JEF VS DCF)

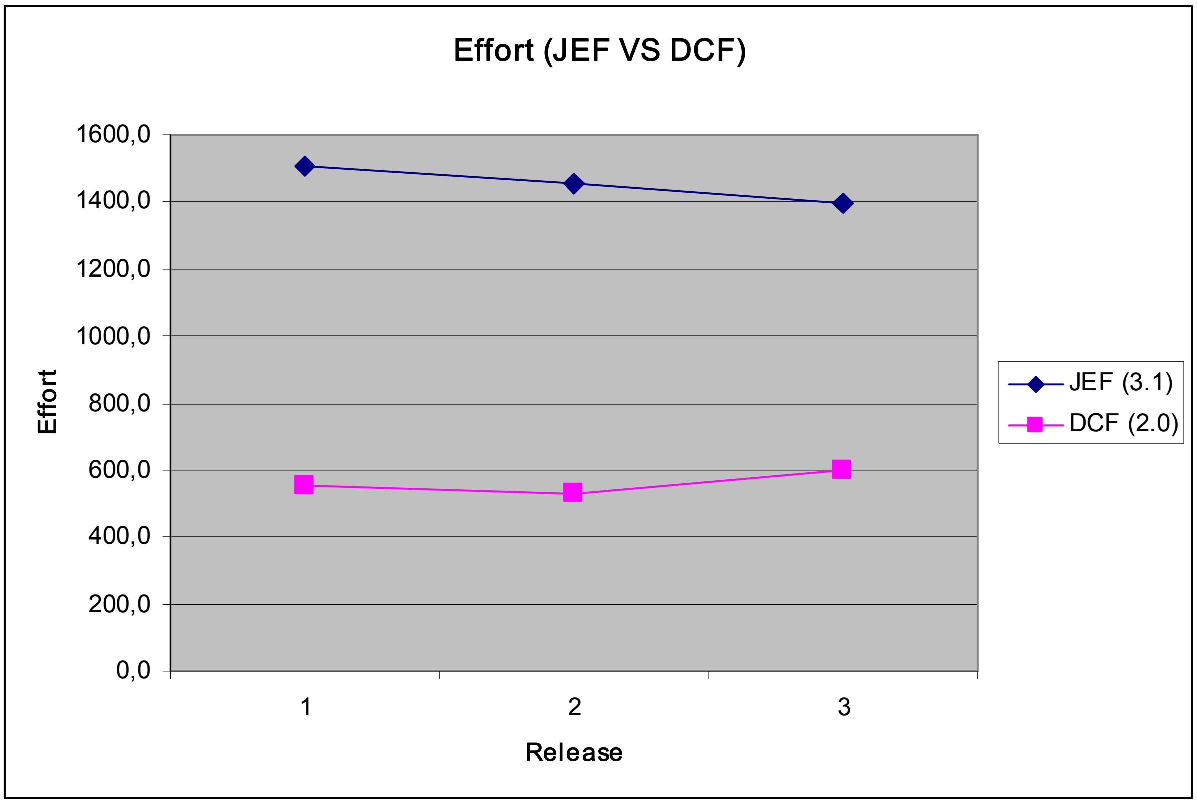


Figure 6-3: Effort (JEF VS DCF)

The graph and the values in the table show that the effort in JEF is considerable higher than DCF. However the trend of the effort in JEF is in the right direction, while DCF moves in the wrong direction. But the difference of the effort between JEF and DCF is so far away from each other, it will require much more releases before those efforts will meet, and only if they continue to keep this development.

DCF and S&A VS JEF

The following table and line graph shows the effort of the last version in each project.

LOC DCF	Effort DCF	LOC S&A	Effort S&A	LOC JEF	Effort JEF
25 072	598,3	64 319	264,5	20 348	1395,7

Table 6-5: Effort (DCF and S&A VS JEF)

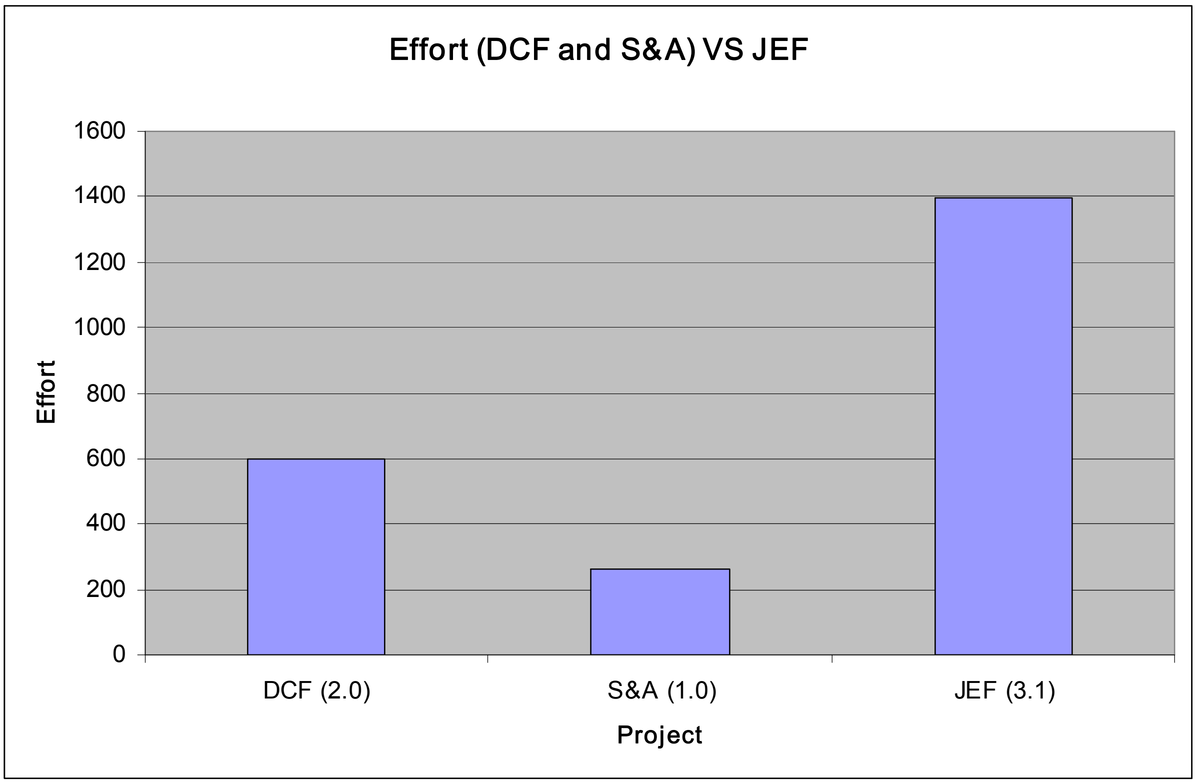


Figure 6-4: Effort (DCF and S&A VS JEF)

The graph is clearly presents that the effort in JEF is much higher than the two other projects. For example, the effort in JEF is over 5 times higher than the effort in S&A. At this moment we will determine to accept the hypothesis.

Is there any reason to tell why the effort in DCF and S&A is much lower than JEF? A reason is clearly shown by the number of LOC in DCF and S&A. They are much higher than the LOC in JEF. Both DCF and S&A are based on a part of JEF's reused components. Those reused components consume almost zero for the cost while contributing to a higher LOC. This will yield a much lower effort. This is one of the largest advantages of reusing software components, and it is clearly shown in Statoil's numbers.

6.2 Defect density evaluation

The hypothesis is as follows:

> *"H_D: The defect density in DCF and S&A is lower than the defect density in JEF."*

6.2.1 Question QD-1

> *"What is the defect density in DCF?"*

The answer to the question is using data information from sub-chapter 5.2.3 and table 5-4. The table below presents the defect density in DCF according to different versions:

DCF version	LOC	Defect number	Defect density
1.0	20 702	880	0,0425
1.1	21 459	880	0,0410
2.0	25 072	880	0,0351
Average	8 357	293	0,0117

Table 6-6: Defect density in DCF

The following line graph shows the defect density in DCF.

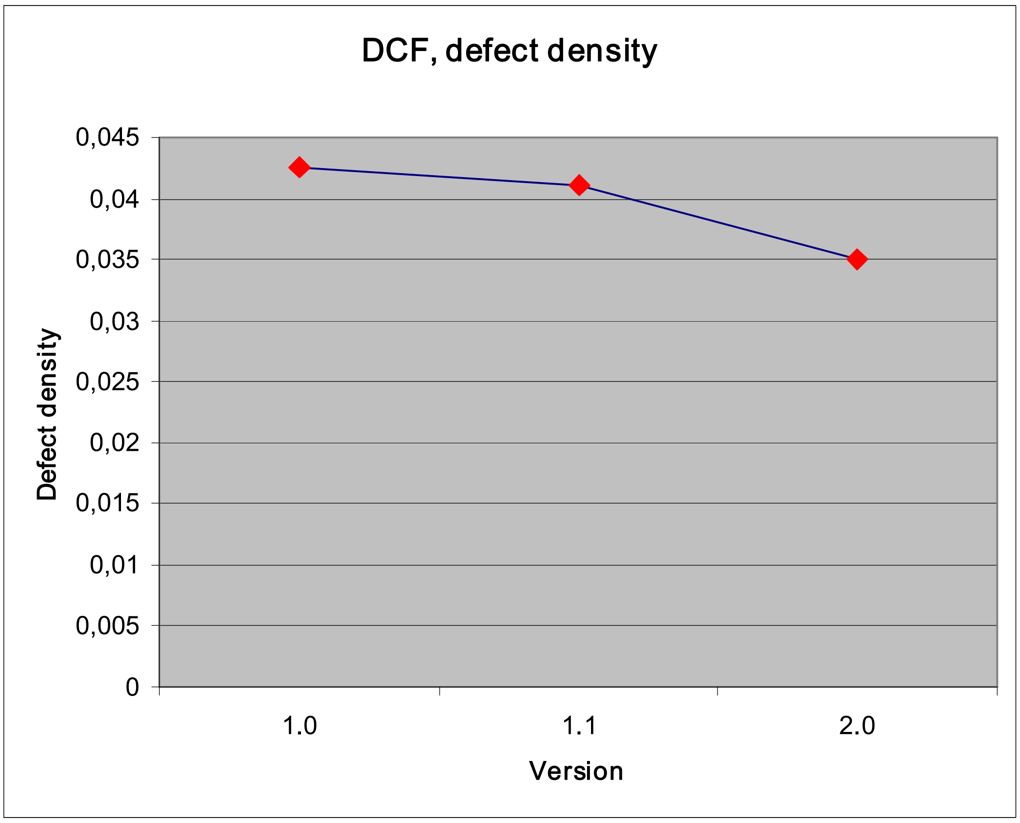


Figure 6-5: Defect density in DCF

The graph shows us that the defect density decreases from version 1.0 to version 1.1, and from version 1.1 to version 2.0 the defect density is decreased even steeper. The defect density in those deliveries is moving in right direction.

6.2.2 Question QD-2

"What is the defect density in S&A?"

The answer to the question is using data information from sub-chapter 5.3.1 and sub-chapter 5.3.3. The table below presents the defect density in S&A.

S&A version	LOC	Defect number	Defect density
1.0	64 319	239	0,0037

Table 6-7: Defect density in S&A

6.2.3 Question QD-3

"What is the defect density in JEF?"

The answer to the question is using data information from table 5-1 and 5-3. The table below presents the defect density in JEF according to different versions:

JEF version	LOC	Defect number	Defect density
2.9	16 875	197	0,0117
3.0	18 599	219	0,0118
3.1	20 348	223	0,0110
Average	6 783	74	0,0037

Table 6-8: Defect density in JEF

The following line graph shows the defect density in JEF.

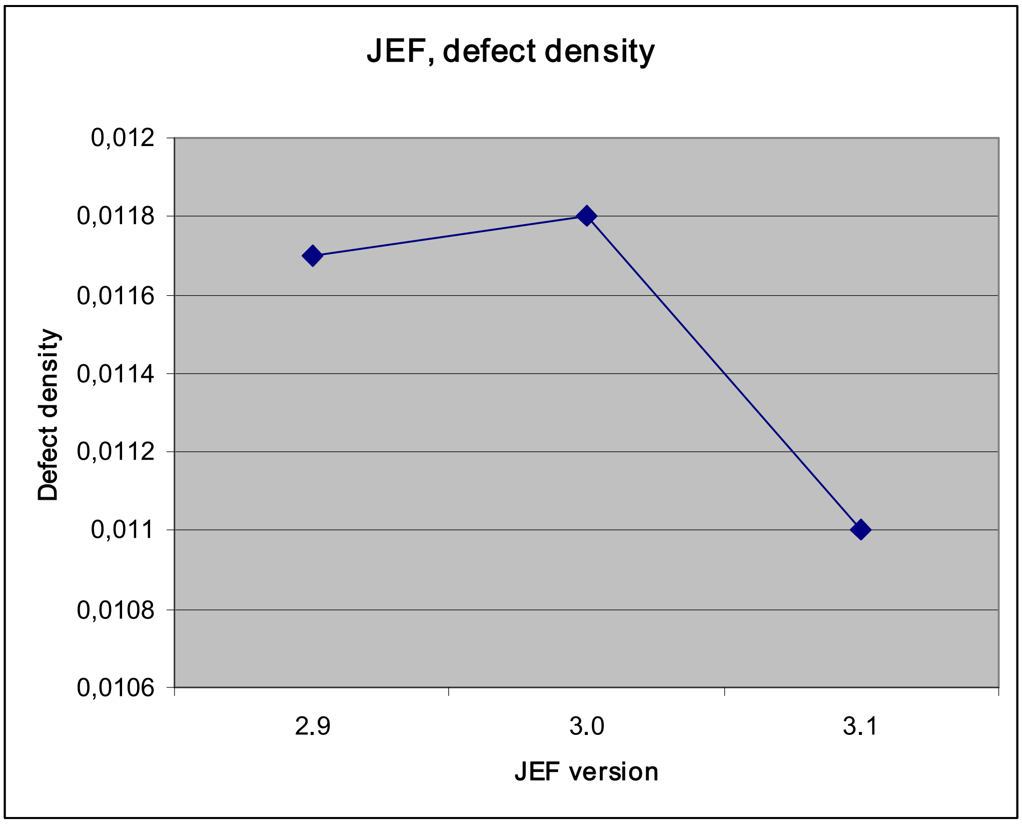


Figure 6-6: Defect density in JEF

The graph shows that from version 2.9 to version 3.0 the defect density increased. Thus the line between those versions is moved in the wrong direction. From version 3.0 to version 3.1, the defect density decreased. The line falls steeply between the deliveries.

6.2.4 Question QD-4

"If the defect density in DCF- and S&A- application is higher than the defect density in JEF, does priority (critical, high, medium, low) to correct the defect make any difference?"

JEF VS DCF

The following is presenting table and graph, where they facilitate to compare when those values are putted beside each other and in the same table and graph.

Release	LOC JEF	DD JEF	LOC DCF	DD DCF
1	16 875	0,0117	20 702	0,0425
2	18 599	0,0118	21 459	0,0410
3	20 348	0,0110	25 072	0,0351
Average	1 158	-0,0002	1 457	-0,0025

Table 6-9: Defect density (JEF VS DCF)

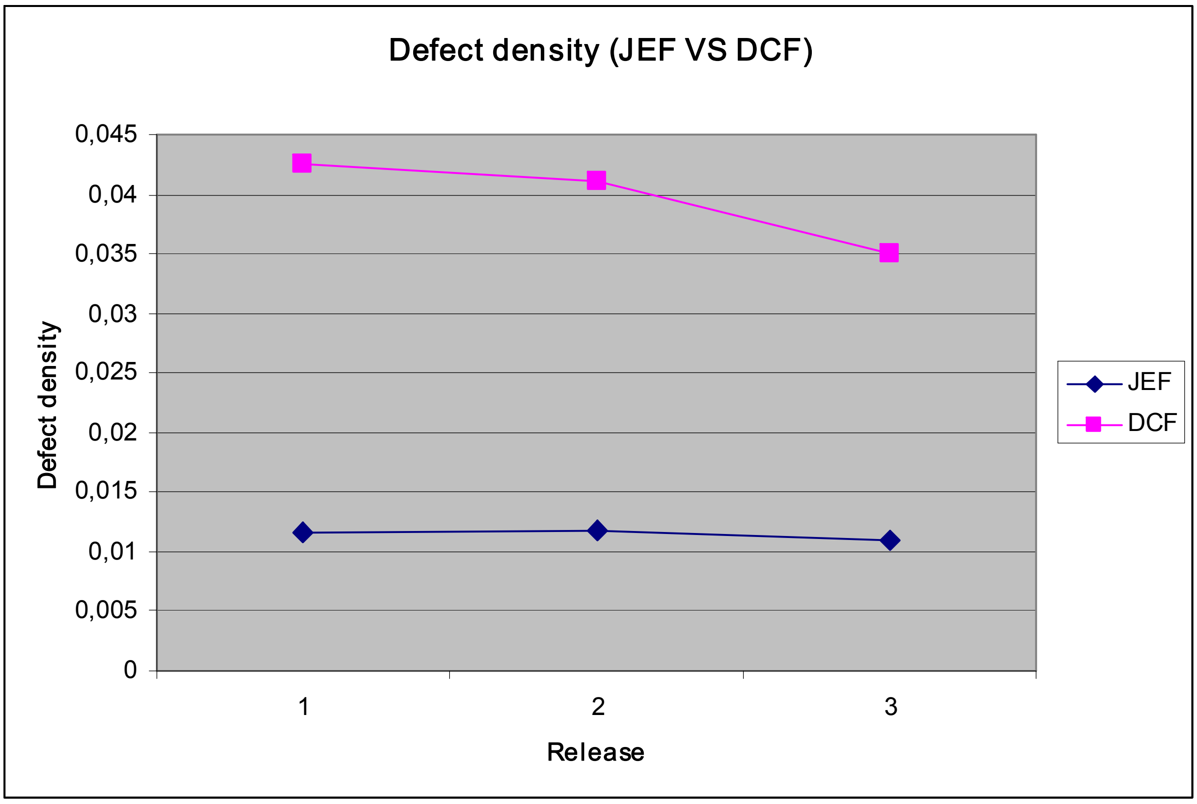


Figure 6-7: Defect density (JEF VS DCF)

The graph and the values in the table show that the defect density in JEF is quit lower than DCF. The movement of defect density in DCF is moving in right direction. Especially from the second release to the third release in DCF, it has decreased steeply. All the while the defect density in JEF is also moving in right direction, but it does not decrease as steeply as DCF does. The difference of the defect density between JEF and DCF is quite far away from each other.

DCF and S&A VS JEF

The following table and line graph show the defect density of the last version in each project.

LOC DCF	DD DCF	LOC S&A	DD S&A	LOC JEF	DD JEF
25 072	0,0351	64 319	0,0037	20 348	0,0110

Table 6-10: Defect density (DCF and S&A VS JEF)

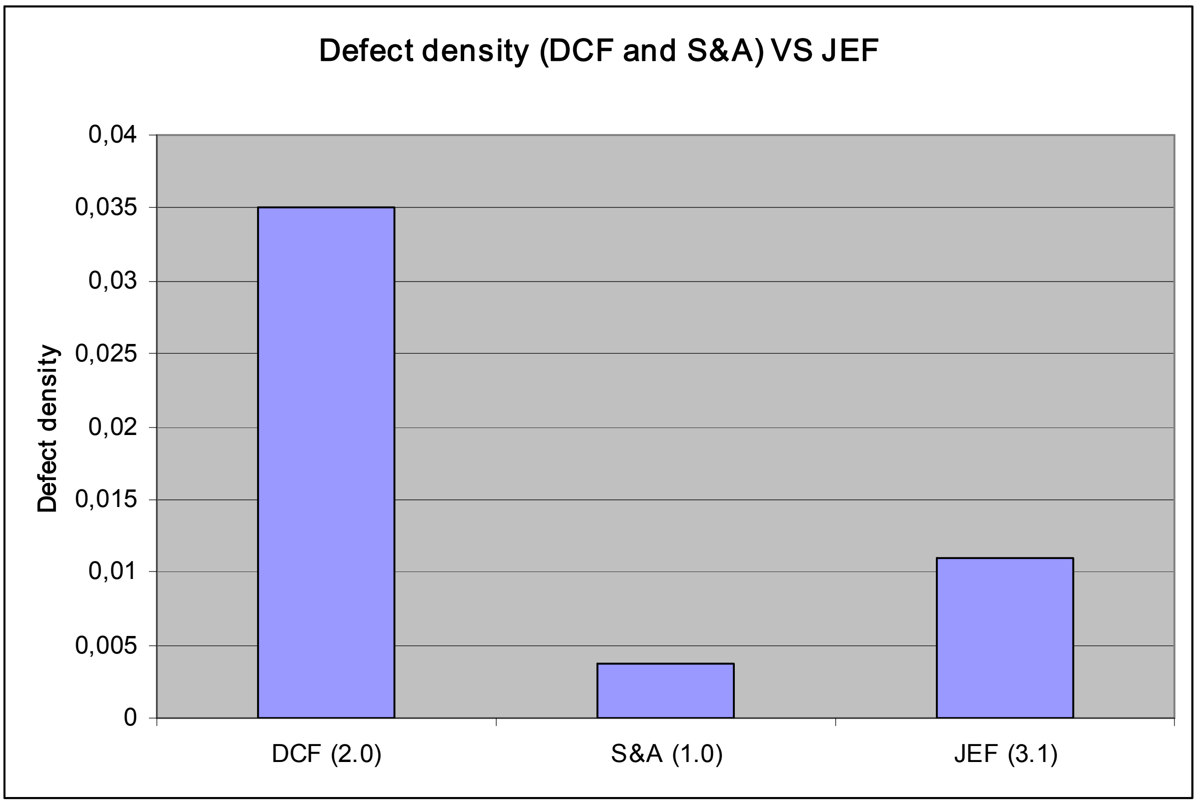


Figure 6-8: Defect density (DCF and S&A VS JEF)

The defect density in DCF is highest, JEF is in the middle and S&A is lowest. But the graph above is not enough foundation to answer the question, thus we need to look more detail in the attributes of defects. Reused components have more defects with highest severity than the total distribution, but less defects after delivery, which shows that these are given higher priority to fix [9]. We will look about the priority to determine if this does matter for our case. We will be using the data information in figure 5-1, 5-5 and 5-10 to describe a surveyable set of priorities in the table below. The numbers in the table are presented in percentages.

Priority / Project	JEF	DCF	S&A
Critical	0,015	0,100	0,126
High	0,282	0,395	0,402
Medium	0,345	0,338	0,293
Low	0,301	0,167	0,167
Not prioritised	0,058	0,000	0,013

Table 6-11: Priority in JEF, DCF and S&A

High priority to correct the defect correspondences the defect may made big impact in the system, such as component's communication, system crash, configuration and so on, and it may claimed longer time to fix.

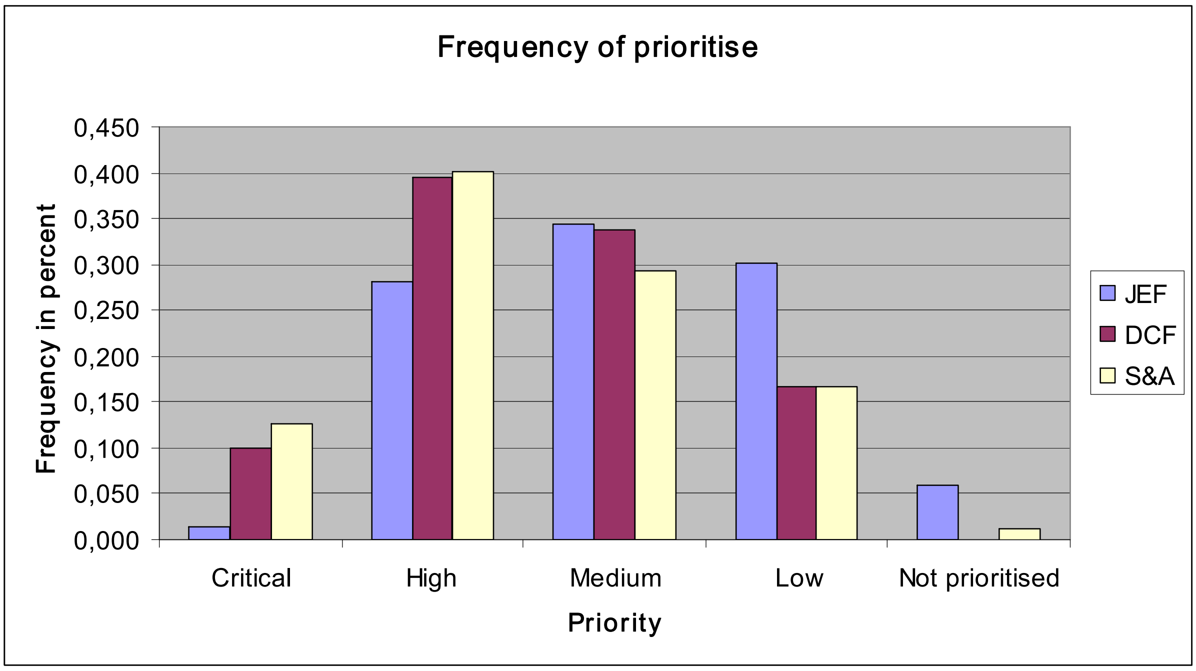


Figure 6-9: Priority in JEF, DCF and S&A

The described result shows that JEF gets middle defect density and it gets in average lowest priority to correct the defect of the three projects. DCF gets highest defect density and it gets in average middle priority to correct the defect of the three projects. With this we like to determine the priority to correct the defect doesn't make any difference.

6.2.5 Analysis of hypothesis; H_D

Since the defect density in JEF gets middle between DCF – and S&A – application, it makes difficult to conclude the hypothesis. We can not accept this hypothesis because of the high defect density in DCF. At this moment we will conclude that will not reject this hypothesis because of S&A.

6.3 Stability evaluation

It is often said:

> "*To improve is to change, to be perfect is to change often.*"

The option to change is an important factor in all kind of systems. Quick and easy are crucial requirements for changes. The successful companies are able to rapidly implement changes. The figure of Evolution Delivery Life Cycle in sub-chapter 4.1 is a good illustration to describe the processes of demand to change in a project

To remind the hypothesis of effort is:

> *"H_S: The stability in project that has been developed from scratch (JEF) is lower than the stability in the projects that base on reusable components (DCF)."*

6.3.1 Question QS-1

"What is the number of change request in the last version of DCF?"

A request is submitted to the service owner. This request should clearly define the nature of the requested change, any deficiencies that the change is expected to correct, problems or symptoms to be addressed, the conditions for satisfaction of the request. Change request which fails to meet the minimum criteria for submission may be honored by the service owner or rejected at their discretion.

6.3.2 Analysis of hypothesis; H_S

At this moment we cannot draw any conclusion about the hypothesis, because there is lack of data information in JEF and S&A to make comparison.

6.3.3 Summing up

Here is a table of summing up above these hypotheses.

HypID	Hypothesis text	Conclusion
H_E	The effort in DCF and S&A is lower than the effort in JEF.	Accepted
H_D	The defect density in DCF and S&A is lower than the defect density in JEF.	Not rejected
H_S	The stability in project that has been developed from scratch (JEF) is lower than the stability in the projects that base on reusable components (DCF).	Unable to draw conclusion

Table 6-12: Summary of hypotheses conclusion

7 Evaluation and discussion of results

This chapter will present evaluation of the results with regards to validity and a discussion about the study.

Subchapter 7.1 will discuss the different threats to validity.

Subchapter 7.2 will discuss the study, the restraints and limitation of the study, and how it could have been done differently.

7.1 Threats to validity

As written in chapter 2, there are four different types of validity. We will now discuss each of them in relation with our study, and try to point out threats that might undermine our conclusion.

Threats to conclusion validity:
Is there a relationship between the introduction of JEF and the change in effort and defect density in DCF and S&A in the context of reuse?

One minor threat to this validity is that we were missing some key data for our study. We only got change request documents for DCF, and it was therefore not possible to make a comparison to see if reuse has affected the stability. There were also some missing attributes from the incident reports, such as severity in JEF.

Another threat would be that JEF has a different functionality and composure than DCF and S&A. JEF has been developed for reuse, while DCF and S&A have been developed by reusing.

In addition, there could have been different development team working on the different projects. We can not determine if this is a major or minor threat because we do not have the necessary data to confirm this.

Threats to internal validity:
If there is a relationship between the introduction of JEF and the change in effort and defect density in DCF and S&A, is the relationship causal?

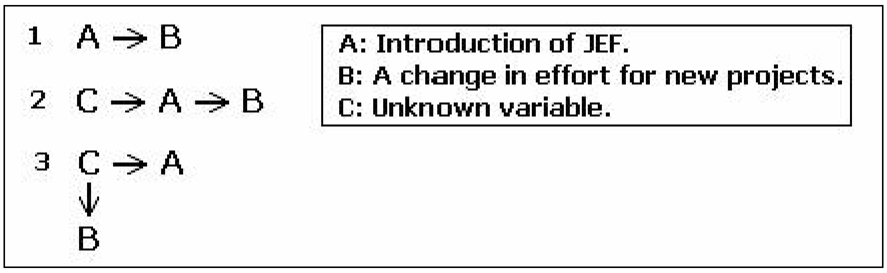


Figure 7-1: Different types of relationships

What we want to answer is if the relationship between A is directly related to B (1), and not simply a side effect of another variable (2 or 3).

There are three factors, which may have been an influence on the change in cost for new projects.

1) An introduction to a new chief architecture will also introduce a massive focus on documentation and observation. Employees under this focus will be pressed to work more efficiently because they want to prove their knowledge and skill.
2) An introduction to a new architecture might also thrill some employees. They might be excited over new features in the new architecture and work harder than normal to test the new tools and concepts.
3) The productivity and efficiency may decrease because more defects and misunderstanding will occur because of the tools and concepts.

However, we believe that Statoil is a matured and experienced organization and proper preparations has been made internally to meet these threats. Even if these threats are highly likely, we think they will make an insignificant influence on our study.

Threats to construct validity:
This validity is mostly used in conducting experiments, and is therefore not relevant for our study. We have chosen to exclude this validity from our discussion.

Threats to external validity:
This study has been conducted specifically to JEF, DCF, S&A and Statoil. It is therefore not possible to generalize this study to other organizations or domains. It could, however, be feasible to generalize this study for future projects in JEF. One threat to validity is that this study has been conducted in relatively early time period in JEF.

We think that our study will yield a valid conclusion and that the threats to validity have not undermined the conclusion.

7.2 Discussion

After conducting our study and made the analysis of the available information, we have identified one major issue that has restrained our work throughout the thesis. There has simply not been enough data from JEF and its projects to draw a firm and sound conclusion.

First, the development of JEF and related project as DCF and S&A are still young. Theirs information data is not complete, thorough and surveyable; such as at this moment S&A is released only one version and this version is not finished yet. During the work in this thesis, we discovered that we had to carry out a lot of considerations and make a lot of change to define the commissions or hypotheses as regard the lack of data information in those projects. It claims from us primary a lot of work and time.

Second, the data information in those projects is not accordance to each other, and it makes difficult to define the hypothesis; such as there is missing data information about change request and severity in JEF, while DCF and S&A contain those information. This is one of the reasons why we cannot draw the conclusion of the hypothesis about stability.

Third, we have not had a direct communication with Statoil. All our questions have been answered through either Anita or Odd Petter. We think, that a direct communication method with Statoil would have significant advantages, such as clarity in what we wanted, more in-depth information that data alone can not give us and less time spent waiting for answers. However, we appreciate all the help we were given by Anita and Odd Petter, and it has become clear that this method of approach has been valuable to us. Not only does these two have major domain knowledge, but they have also been helping us with other various issues. In addition, we did not have the opportunity to conduct all of our study. We had planned interviews, surveys and case studies to support our thesis. But it was revealed that Statoil has its own system for performing such activities, and that the system is not made available for people from the outside (this includes Anita and Odd Petter). This reason discouraged us from conducting the empirical work we had planned.

Fourth, delay of information. Since the reason above, we got late information. In the beginning we thought that we could get the data information about change request in JEF, thus we kept the hypothesis about stability. We got a late message that it might not possible to get data information about change request in JEF, and this is another reason we cannot be complete the hypothesis.

These mentioned points had a major influence on our study. The most noticeable part, which is not shown in this thesis, is the continually changes of the hypotheses. We wished that we could establish well-formed hypotheses of what we really wanted to explore, and then get the needed data resources to support our journey. We had, however, not this privilege. Because of the fourth points mentioned above, we had to continually change our hypotheses to correspond with the data resources we could get from Statoil. This made us revise our hypotheses several times, and we had to make major changes in them (not counting small changes on research questions)

8 Conclusion and further work

Subchapter 8.1 will conclude this thesis.

Subchapter 8.2 will give aspect of thesis that can be investigated further that we did not have time or resources to do.

8.1 Conclusion

The purpose of this thesis has been to conduct a study of component reuse and whether it is advantageous to Statoil or not. The continued cooperation between us and Statoil is still to study the component reuse with SJEF. In our previous report we have made a pre-study to understand the general attitude toward component-based development, in this report we have made deeper insight into component-based development and made greater hypothesis researches.

We decided to use empirical strategy in our quantitative study. All of our data resource has been gathered by our mentor and two PhD students who acted as contact person for Statoil. As for the hypotheses presented in chapter 4 – Research focus and problem elaboration, all hypotheses were evaluated through the conducted empirical strategy.

The results show that there are major advantages that could be made from reusing software components. The most obvious advantage from this study is shown in the analysis in effort. We think that these results can be used to further hone the art of reuse in Statoil, as well as a base for further studies.

8.2 Further work

In this chapter, suggestions for further work on this subject of this thesis will be discussed. The hypotheses could be used to make a prediction model for future systems in the same environment or for maintaining the current system. Following sub-chapters are specified in four sub-chapters, so descendants can make assessment about these factors are relevant or not. It may be interesting to look if further releases of JEF will make any changes to the conclusions in the followed hypotheses. We are investigating if the effort in JEF will be improved and the defect density in JEF will be sustained. The defect density can also be specified in attribute of and the stability in JEF will also make sense.

8.2.1 Will the effort in JEF be improved?

The effort in JEF is much higher than DCF and S&A, and this is groundwork to accept the first hypothesis. The compared graph between JEF and DCF tells that the effort in JEF is moving in right direction and the effort in DCF is moving in wrong direction. It may be interesting to know if there will be any time those efforts will meet. If the movements of effort in those projects are keeping moving in this way, there is one day those lines will be met. Here is an illustrated table and graph, which is presenting the meeting point according to the number of releases in those projects.

Release	LOC DCF	Effort DCF	LOC JEF	Effort JEF
3	25 072	598,3	20 348	1395,7
4	26 529	614,2	21 506	1359,2
5	27 985	630,0	22 663	1322,7
6	29 442	645,9	23 821	1286,2
7	30 899	661,8	24 979	1249,8
8	32 355	677,6	26 136	1213,3
9	33 812	693,5	27 294	1176,8
10	35 269	709,4	28 452	1140,3
11	36 725	725,2	29 609	1103,8
12	38 182	741,1	30 767	1067,3
13	39 639	757,0	31 925	1030,8
14	41 095	772,8	33 082	994,3
15	42 552	788,7	34 240	957,8
16	44 009	804,6	35 398	921,3
17	45 465	820,4	36 555	884,9
18	46 922	836,3	37 713	848,4
19	48 379	852,2	38 871	811,9

Table 8-1: The effort in JEF will be improved?

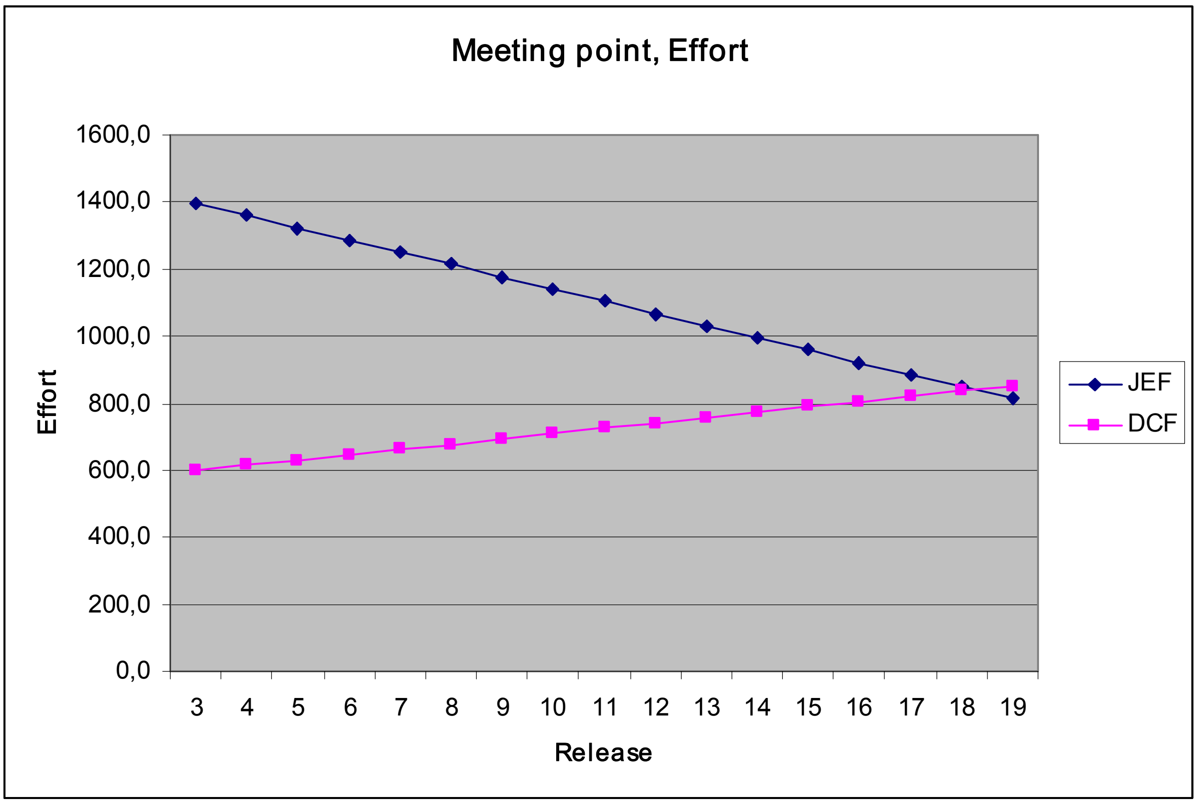


Figure 8-1: Meeting point for defect density for JEF and DCF

The calculation in the illustrated table about LOC from each release is based on to add the average increased LOC in the last release in each project. The average increased LOC is presented in a previous table from chapter 6. The effort in JEF is added the average decreased effort in the last release in JEF, and the effort in DCF is added the average increased effort in the last release in DCF. This is the basis to illustrate the linear graph above.

$$\mathrm{LOC}_{\text{Project X}}^{\,n+1} = \mathrm{LOC}_{\text{Project X}}^{\,n} + \overline{\mathrm{LOC}}_{\text{Project X}}$$

$$\mathrm{DD}_{\text{Project X}}^{\,n+1} = \mathrm{DD}_{\text{Project X}}^{\,n} + \overline{\mathrm{DD}}_{\text{Project X}}$$

Figure 8-2: Formula to draw linear graph

8.2.2 Will the defect density in JEF be sustained?

The compared graph about defect density between JEF and DCF tell us that the defect density in JEF is sustained after three releases and defect density in DCF is moving in right direction. In the same way as above, is there any time the lines of defect density in DCF and JEF will meet? Here is an illustrated table and graph, which presents the meeting point according to the number of releases in those projects.

Release	LOC DCF	DD DCF	LOC JEF	DD JEF
3	25 072	0,0351	20 348	0,0110
4	26 529	0,0326	21 506	0,0108
5	27 985	0,0302	22 663	0,0105
6	29 442	0,0277	23 821	0,0103
7	30 899	0,0252	24 979	0,0101
8	32 355	0,0228	26 136	0,0098
9	33 812	0,0203	27 294	0,0096
10	35 269	0,0178	28 452	0,0094
11	36 725	0,0154	29 609	0,0091
12	38 182	0,0129	30 767	0,0089
13	39 639	0,0104	31 925	0,0087
14	41 095	0,0080	33 082	0,0084

Table 8-2 The defect density in JEf will be sustained?

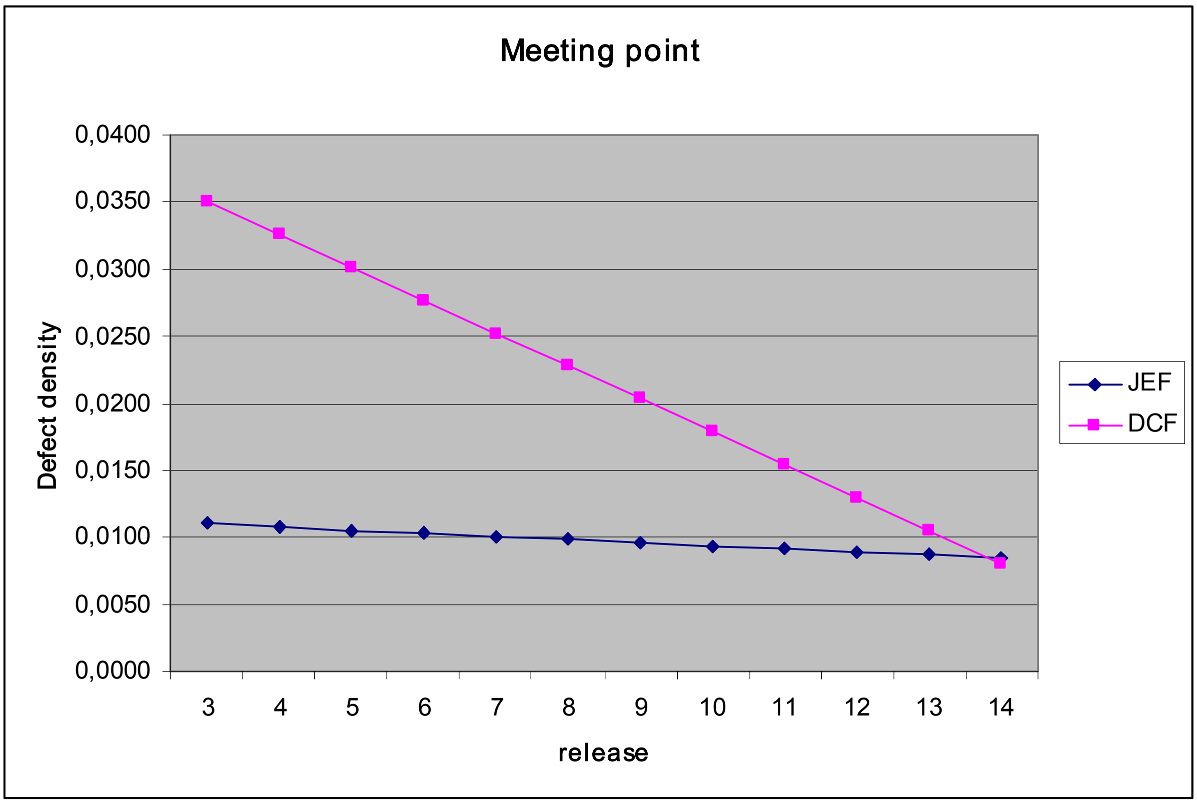


Figure 8-3: Meeting point for defect density for JEF and DCF

The calculation in the illustrated table is presenting in the same way as shown above sub-chapter.

This can be done for project S&A at a later period as well. But since we only have one version of S&A, we can not make any assumptions of how S&A's defect, costs and size be evolved over time.

8.2.3 Defect density and attributes

We have only used the number of defects as the metric when comparing defect density between projects. At an early stage of our thesis we wanted to look beyond these numbers and look into the attributes of the incidents. Priority, severity, length of time spent on an incident and such are good indicators of the size of an incident. By using these attributes, we could give each incident a value, and thus make a more in-depth comparison between the projects. An example would to give the different ratings in severity a value:

Severity rating	Value
Critical	4
High	3
Medium	2
Low	1

Table 8-3: Example for values on severity

Then we could sum up the "*true*" value of severity (the numbers below are fictional):

	DCF		JEF	
Severity rating	Incidents	Point	Incidents	Value
Critical	1	4	6	24
High	3	9	5	15
Medium	8	16	15	30
Low	20	20	6	6
Sum	32	49	32	75

Table 8-4: Example of points for severity

By looking only at the number of defects, we could say that JEF and DCF have the equal defect density (assuming they had the same size in LOC). But by giving each incident a value, we see that JEF has a higher number of larger incidents, thus giving it a larger total value. This could affect our conclusions.

8.2.4 Stability in JEF

It was not possible to gather the data about change request in JEF from Statoil. Thus we cannot answer the question, and the analysis is suggested to further work. The metric to define stability is defined in previous chapter; it uses the number of change request from a project to compare to the number of change request from another project.

Hypothesis of stability	
H_x: The stability in project that has been developed from scratch (JEF) is lower than the stability in the projects that base on reusable components (DCF).	
Question ID	Question
QS-X	What is the number of change request in JEF?

Table 8-5: Stability in JEF

Since this document has responded the question about number of change request in the DCF, then it can be one of the reference points in further work by comparing to the change request in JEF and draw the conclusion of the hypothesis.

In addition, the stability in a project can also be measured by using its released version to define how well stability was going during the development from the one to another version. Like the same method, by comparing to the number of change request between two released versions; $A_{1CH} < A_{2CH}$.

8.2.5 Modifiability in a project

In the beginning we wished to emphasize on modifiability regarding quality attributes. But because lack of time we had to put off this focus to further work. We can investigate the relationship in defects between the limeware and the actual upgrade of new versions of projects. If a project has a high amount of defect density in it's limeware, or still has a large percent of its errors from past version compared to the defect density in the new upgrade, we can say that the project has a low modifiability.

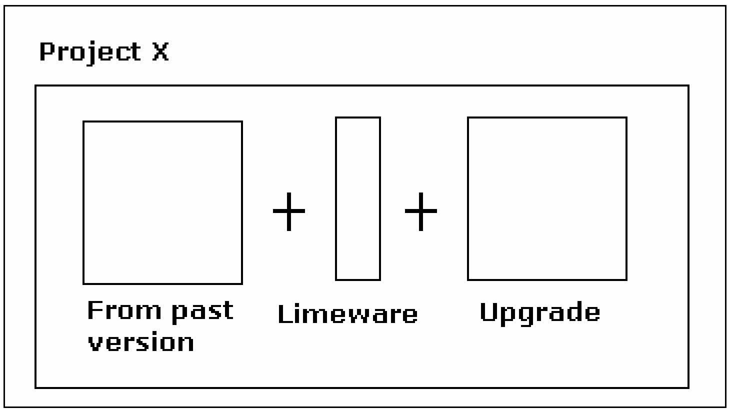


Figure 8-4: Defect density for modifiability

It can be specified at DCF, S&A and JEF through several versions and look at the where the defects are located. If almost all the defects are located in the upgrade part this may tell that the last version has a high degree of modifiability.

9 Appendices

This chapter will present the references that are made and will provide an explanation of each abbreviation used in our thesis.

Subchapter 9.1 will provide all references.

Subchapter 9.2 will explain all the abbreviations.

9.1 References

Number	Reference
[01]	DCF document from Statoil: “DCF v2.0 – Project Handbook.doc” “DCF ICT Architecture.doc” “DCF 2 Requirement Specification.doc”
[02]	ISO/IEC. 2000, “Information technology – software product quality – Part 1: Quality model”, ISO/IEC FDIS 9126-1: 2000(E)
[03]	J. C. Munson and T. M. Khoshgoftar, “Software metrics in reliability assessment”, in Handbook of Software Reliability Engineering, Ed. M. R. Lyu, IEEE-CS Press/McGraw-Hill, 1996.
[04]	JEF Document from Statoil: “SJEF Concepts and Definations Version 2. Date of issue: 04.04.2005.” “JEF 3.0 release info.doc” “JEFroadmap.doc”
[05]	J. Musa, “Software Reliability Engineering”, McGraw-Hill, 1999
[06]	Len Bass, Paul Clements, Rick Kazman: “Software Architecture in Practice”. Second Edition, Chapter 7.
[07]	Mari Torgersrud Haug and Thea Christine Steen: ”*An Empirical Study of Software Quality and Evolution in the Context of Software Reuse (at Statoil)*”. 20 Dec. 2005. http://www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2005/steen-haug-fordyp05.pdf
[08]	McCall J.A, Richards P.K, and Walters G.F, 1977. “Factors in software quality”. RADC-TR-77-369, US Dept. of commerce
[09]	Parastoo Mohagheghi, Reidar Conradi, Ole M. Killi, Henrik Schwarz: “An Empirical Study of Software Reuse vs. Defect-Density and Stability”. http://www.idi.ntnu.no/grupper/su/publ/pdf/ericsson-qa-icse04-final.pdf
[10]	PerOlof Bengtsson, “Architecture-Level Modifiability Analysis”, Bleking Institute

	of Technology, 2002.
[11]	S&A Document from Statoil: "S+A_Requirement_Specification.doc" "SA ICT architecture.doc"
[12]	Victor R. Basili, Lionel C. Briand, and Walcelio L. Melo: "*How reuse influences productivity in object-oriented systems*". Communication of the ACM, 39(10), October 1996. http://delivery.acm.org/10.1145/240000/236184/p104-basili.pdf?key1=236184&key2=7388656411&coll=portal&dl=ACM&CFID=70552384&CFTOKEN=43329339
[13]	Walcelio L. Melo, Lionel Briand, and Victor R. Basili: "*Measuring the impact of reuse on quality and productivity in object-oriented systems*". Technical Report CS-TR-3395, 1995. http://www.cs.umd.edu/~basili/publications/technical/T95.pdf
[14]	William M. K. Trochim: "*Introduction to validity*". Last revised 16th January 2005. http://www.socialresearchmethods.net/kb/introval.htm

9.2 Abbreviations

Abbrevation	Description
API	Application programming interface. The interface that a computer system or application provides in order to allow requests for service to be made of it by other computer programs
C&V	A planned project in JEF which has not started being developed yet.
CBA	Cost-benefit analysis is the process of weighing the total expected costs vs. the total expected benefits of one or more actions in order to choose the best or most profitable option.
CBD	Component Based Development
CBSE	Component Based Software Engineering
CORBA	Common Object Request Broker Architecture is a standard for software componentry, created and controlled by the Object Management Group (OMG).
COTS	Commercial off-the-Shelf, referring to software components from third parties.
CR	Change request, a request for change to the next version of project.
DCF	Digital Cargo Files, a project under development in Statoil.
DD	Defect Density, a measure the frequency of defects based on software size. DD is calculated from total number of defects divides on SLOC (either logical or physical).
GOTS	Government off-the-Shelf, referring to software components from the government.
GQM	Goal-Question-Metric is a method of experiment design. The goals define the abstract level of the experiment. The questions identify the operational level (what we want to learn). The metrics define the quantitative level (what we will measure).
INCO	INcremental and COmponent-base development is a project that has the primary goal to advance the state-of-the-art and -practice for incremental and component-based software development.
J2EE	Java 2 Enterprise Edition platform
JEF	Java Enterprise Framework for Statoil is a J2EE technical framework for Enterprise Applications.

O&S Masterplan	A project template which every new project in JEF should follow.
NTNU	Norwegian University of Science and Technology, an university in Trondheim, Norway.
S&A	Shipment and Allocation, a project that is under development in JEF. Version 1 of this project has not been finished yet, so there is no data resource available from this project.
SJEF	Statoil JEF.
SLOC	Source Line-of-Code is a measurement for sizes of software. It can be either logical SLOC or physical SLOC.
Statoil	Statoil ASA is one of the largest operators on the Norwegian continental shelf.