NTNU
Innovation and Creativity

WebSys- Robustness Assessment and Testing

Thuy Hue Thi Pham

Master of Science in Computer Science
Submission date: June 2006
Supervisor: Tor Stålhane, IDI

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

Problem Description

Oppgaven består av en teoridel og en praktisk del.

- Teori: Basert på arbeide som er gjort i WebSys- hvordan kan vi (1) vurdere hvor robust et system er? (2) Teste hvor robust et system er?
- Praktisk arbeid: Basert på teoridelen og ved hjelp av et verktøy fra en tidligere diplomoppgave(AutAT) skal man (1) Definere et sett av tester for robusthet. (2) Evaluere testresultatene og sammenlikne de med resultatene fra teorien

Assignment given: 20. January 2006
Supervisor: Tor Stålhane, IDI

Abstract

In recent years, the World Wide Web (WWW) has become a popular platform for system development. There are several factors that make Web-development special. There is a large number of quality requirements in the Web-based system. Web projects involve people with a diverse background, such as technical people with background in programming and non-technical people with background in graphical design. In addition, the Web-based system are often not developed separately, but is integrating existing subsystems. The time-to-marked requirement is strong. Web-based system must tolerate errors and abnormal situations caused by internal component failure or user mistakes. Therefore, robustness is considered to be a critical factor for Web-based systems. Building a robust Web-based system is never an easy task.

Furthermore, the end users of Web-based systems have different backgounds. Many have knowledge of the Web, others have little or no knowledge of the Web. Since Web-systems are used by people with a rather diverse background, it is important that the Web-based systems must have error tolerance and ability to survive due to user mistake.

The main focus of this project is analyzing robustness of Web-based system. In order to analyze robustness of Web-based system, it is necessary to carry out a robustness assessment. Assessment methods are used to evaluate the robustness and give an estimating of the system's robustness. Further, robustness testing of a Web-based system has to be performed to get an idea of the system's current robustness. The result of estimating and test result will also be discussed, compared and evaluated.

An Automatic Acceptance Testing of Web Applications (AutAT) will be used to test the robustness of a Web-based system. DAIM (Norwegian: Digtal Arkivering og Innlevering av Masteroppgaver) is the target system that will be tested the robustness of.

Keywords: Robustness, testing, robustness assessment, robustness estimating, Web-based system, AutAT, DAIM.

Preface

This master thesis is written as part of my master degree at the Department of Computer(IDI) and Information Science at the Norwegian University of Technology and Science (NTNU) in Trondheim in the spring of 2006.

I would like to thank my supervisor, Tor Stålhane for valuable feedback, support, ideas, and comments during my work. In addition, I will also thank Kai Torgeir Dragland and Jan Grønsberg for helping me understanding the DAIM system.

Trondheim 16th of June 2006

Hue Pham Thi Thuy

Contents

I Introduction 1

1 Motivation 2

2 Project Context 3
- 2.1 Background . . . 3
- 2.2 Problem Definition . . . 3
- 2.3 Limitation of Scope . . . 4

3 Outline of the Report 5

II Prestudy 7

4 Definitions and Basic Concepts 8
- 4.1 Robustness . . . 8
- 4.2 Robustness and other quality requirements . . . 9

5 State of the Art - Existing Robustness Assessment 11
- 5.1 Jacobson's analysis method and FMEA . . . 11
  - 5.1.1 Object-Oriented Software Engineering (OOSE) . . . 11
  - 5.1.2 Failure Mode and Effect Analysis (FMEA) . . . 14
  - 5.1.3 The proposed method . . . 15
  - 5.1.4 Robustness Assessment: An Example . . . 18
  - 5.1.5 Conclusion . . . 19
- 5.2 Information flow analysis and hazard analysis . . . 20
  - 5.2.1 Information Flow Analysis . . . 20
  - 5.2.2 Hazard analysis . . . 21
  - 5.2.3 The Process of Quality Modelling . . . 21
  - 5.2.4 Conclusion . . . 22
- 5.3 Application of Hazard Analysis to Software Quality Modelling . . . 22
  - 5.3.1 Representation of quality models . . . 23
  - 5.3.2 Adaption of Hazard Analysis Method . . . 24
  - 5.3.3 The proposed method . . . 24
  - 5.3.4 Robustness Assessment: An Example . . . 25
  - 5.3.5 Conclusion . . . 26
- 5.4 Summary and the Choice . . . 26

5.4.1 Summary 26
5.4.2 The Choice of Assessment method 27

6 Testing 29
6.1 V-model 30
6.2 Types of testing 31
6.2.1 Black Box Testing 31
6.2.2 White Box Testing 31
6.3 Testing of Web Applications 32
6.4 Tools for Automatic Acceptance Test 33
6.4.1 AutAT 34
6.4.2 Watir 34
6.5 Summary 35

7 Goal/Question/Metric method 37
7.1 Goal 38
7.2 Questions 38
7.3 Metrics 39

III Robustness Assessment of DAIM 43

8 DAIM's requirements 44
8.1 Nonfunctional requirements 44
8.1.1 Database 44
8.1.2 Robustness requirements of DAIM 45
8.2 Functional Requirements 45

9 DAIM and Robustness Assessment 49
9.1 Jacobson's analysis method and FMEA 49
9.1.1 UC01: Log in 49
9.1.2 UC02: Fill in contract 51
9.1.3 UC03: Create cooperation group 53
9.1.4 UC04: Generate contract/schema 55
9.1.5 UC05: Deliver master thesis 57
9.1.6 UC06: Search for master thesis 60
9.1.7 UC07: Handle payments 62
9.1.8 UC08: Display master thesis 64
9.1.9 UC09: Choose a Censor 65
9.1.10 UC10: Validate information 66
9.1.11 UC11: Create/update account 68
9.1.12 UC12: Get information 71
9.1.13 UC13: Import data 72
9.1.14 UC14: Change information 74
9.1.15 Conclusion 77
9.2 Application of Hazard analysis to Software Quality Modelling 77

9.3 Estimate DAIM's robustness 78
9.4 Conclusion 80

IV Testing DAIM 83

10 User Testing Session 84

11 Test Results 87
11.1 Metrics 87
11.1.1 M1: Error tolerance 87
11.1.2 M2: Error messages 89
11.1.3 M3: Feedback 90
11.1.4 M4: DAIM precision 91
11.1.5 M5: Syntactic Input Validation 92
11.2 Questions 94
11.2.1 Quesion 1: Input Tolerance 94
11.2.2 Question 2: DAIM interface 94
11.3 Goal Attainment 95
11.4 Threats to Validity 96

V Evaluation 99

12 Discussion and Evaluation 100
12.1 Conclusion and Discussion 100
12.1.1 Early robustness assessment methods 100
12.1.2 Testing 101
12.2 Evaluation 102

13 Further Work 105

VI Appendix 107

A Abbreviations and Glossary 108
A.1 Abbreviations 108
A.2 Glossary 109

B Quality Models and SFMEA 110
B.1 Choose sensor 110
B.2 Validate information 112
B.3 Get information 113
B.4 Import data 115

C Qualitative Robustness Assessment 117

D Tests Data 131

E Original DAIM Document **154**

List of Figures

4.1 A partition over all operational conditions 9

5.1 Analysis phase of the OOSE life cycle 12
5.2 The use case model of the OOSE approach 12
5.3 Stereotype symbols 13
5.4 Interaction rules 14
5.5 Analysis diagram for Search by Author 19
5.6 An example of information flow diagram 21
5.7 Quality Modelling Process 22
5.8 Notation for representaion of quality models 23
5.9 The model for user cannot find required information 26

6.1 V-model of testing 30
6.2 An example of a test in AutAT 35

7.1 An example of GQM-tree 38

8.1 Use cases diagram for DAIM 46

9.1 Analysis diagram for Log In use case 50
9.2 Analysis diagram for Fill in contract 51
9.3 Analysis diagram for Create cooperation group 53
9.4 Analysis diagram for Generate contract/schema 56
9.5 Analysis diagram for Delivery master thesis 58
9.6 Analysis diagram for Search master thesis 60
9.7 Analysis diagram for Handle payments 62
9.8 Analysis diagram for Display master thesis 65
9.9 Analysis diagram for Choose censor 65
9.10 Analysis diagram for Validate information 67
9.11 Analysis diagram for Create/update account 69
9.12 Analysis diagram for Get information 71
9.13 Analysis diagram for Import data 73
9.14 Analysis diagram for Change information 75

11.1 Kiviat diagram for questions and goal attainment 96

12.1 An example of Jacobson analysis diagram at high level 103
12.2 An example of Jacobson analysis diagram at low level 103

12.3 Difference between the estimated and true value of robustness 104

B.1 The quality model for Choose censor 111
B.2 The quality model for Validate information 113
B.3 The quality model for Get information 114
B.4 The quality model for Import data 116

List of Tables

5.1 FMEA with class as starting point 15
5.2 FMEA with Control Object as starting point 17
5.3 A part of FMEA for Search by Author 19
5.4 FMEA format for request 21
5.5 The format of Software FMEA chart 24
5.6 The SFMEA for "User cannot find required information" 25

7.1 Metric 1 - Error tolerance 39
7.2 Metric 2 - Error meassages 40
7.3 Metric 3 - Feedback 40
7.4 Metric 4 - DAIM precision 41
7.5 Metric 5: Syntactic input validationn 41
7.6 The relationships between the metrics and the questions 42

9.1 FMEA for Log in 50
9.2 FMEA for Fill in contract 52
9.3 FMEA for Create cooperation group 54
9.4 FMEA for Generate contract/schema 56
9.5 FMEA for Delivery master thesis 58
9.6 FMEA for Search master thesis 61
9.7 FMEA for Handle payment 62
9.8 FMEA for Choose censor 66
9.9 FMEA for Validate information 67
9.10 FMEA for Create/update account 69
9.11 FMEA for Get information 72
9.12 FMEA for Import data 73
9.13 FMEA for Change information 75
9.14 FMEA with class as starting point 79
9.15 Summary Robustness Estimating 79

10.1 Test data for <Use case ID: use case name> 85

11.1 A summary of test results for each use case 88
11.2 Collected data for M2: error message 89
11.3 Collected data for M3: Feedback 91
11.4 Collected data for M4: DAIM precision 91
11.5 Collected data for M5: Syntactic input validation 93

B.1 SFMEA for Choose censor 110
B.2 SFMEA for Validate information 112
B.3 SFMEA for Get information 113
B.4 SFMEA for Import data 115

C.1 The FMEA for Log in 117
C.2 The FMEA for Fill in contract 118
C.3 The FMEA for Create cooperation group 119
C.4 The FMEA for Generate contract/schema 120
C.5 The FMEA for Deliver master thesis 121
C.6 The FMEA for Search for master thesis 122
C.7 The FMEA for Handle payments 122
C.8 The FMEA for Display master thesis 124
C.9 The FMEA for Choose censor 125
C.10 The FMEA for Validate information 125
C.11 The FMEA for Create/update account 127
C.12 The FMEA for Get information 128
C.13 The FMEA for Import data 129
C.14 The FMEA for Change information 130

D.1 Test data for UC01 - Log in 131
D.2 Test data for UC02: Students 132
D.3 Test data for UC02: Startdate and deadline 133
D.4 Test data for UC02: Bibliographic data 133
D.5 Test data for UC02: Supervisor and Secondary Supervisor 134
D.6 Test data for UC03 - Create cooperation group 134
D.7 Test data for UC04 - Generate contract/schema 135
D.8 Test data for UC05-01 - Upload a picture of front-page 136
D.9 Test data for UC05-02 - Upload an attachment as a zip file 137
D.10 Test data for UC05-03 - Upload the report in pdf format 137
D.11 Test data for UC05-04 - Order extra copy 139
D.12 Test data for UC06 - Basic search 140
D.13 Test data for UC06 - Advanced search 140
D.14 Test data for UC07 - Handle payments 141
D.15 Test data for UC08-01 and UC08-02 Display and sort 142
D.16 Test data for UC08-03 - Display master thesis by searching 143
D.17 Test data for UC09 - Choose a censor 144
D.18 Test data for UC10 - Validate information 145
D.19 Test data for UC11-01 - Add a student Category - Bibliographic data . . 147
D.20 Test data for UC11-02 - Change status/information for supervisor 148
D.21 Test data for UC11-03 - Change status/information for a censor 148
D.22 Test data for UC11-04 - Find a supervisor/censor by searching 149
D.23 Test data for UC12 - Get information 150
D.24 Test data for UC13 - Import data 150
D.25 Test data for UC14-01 - Change institute information 152
D.26 Test data for UC14-02 - Add an administrator user 153

Part I

Introduction

Chapter 1

Motivation

The WWW is becoming a popular form of software applications. Web-based systems are widely applied in many fields, such as education, entertainment, business, etc. Most researchers agree that Web-based systems are different from most other types of software systems. Web development is different because the people who build Web sites are different. Web projects involve people with a diverse background. Furthermore, the time-to-marked requirement is strong and development process should be done evolutionary, with multiple deliveries throughout the lifecycle.

Web-based systems users are always looking for systems that serve them in a reliabe way, providing quick and useful service. In addition, it is important that the Web-based systems must have error tolerance and ability to survive due to user mistake and internal component failure.

Robustness is considered to be a critical factor for Web-based systems for following reasons:

- It's difficult to control the input profile of end users because the Web-based systems are available for almost everyone.
- Web-based systems are often integrated with existing system and built by developers from different backgrounds.
- A early assessment of robustness will prevent robustness failures or reduce chances for such failures. Robustness assessment will also be used to analyze trade-offs when there are contradicting other quality requirements.

Robustness is a common goal that designer of systems strive for. Therefore, it is important to have an early robustness assessment of a Web-based systems. The robustness must also be tested to make sure that the system is robust enough.

Chapter 2

Project Context

This chapter presents background information, problem definition and limitations of the project.

2.1 Background

My master thesis is related to the WebSys (WEB-based SYStem) [15] research project at NTNU. The main objective of WebSys is to develop high-quality research competence and guidelines for industrial development of timely and reliable Web-based systems.

WebSys aims to examine, propose, try out and improve novel methods and techniques to balance the classic counterparts, time-to-market and reliability in the development of the web-based systems. The WebSys has also three sub-goals:

- Goal 1: Better understanding of the software process and related technologies that concerns how to make trade-offs between the Time To Market (TTM) and reliability in web-based systems.
- Goal 2: Contribute to continuous improvement of the software development process in companies where reliability and TTM need to be considered together.
- Goal 3: Dissemination and exchange of knowledge gained. The aim is to disseminate PhD reports, research papers, technical reports, educational and presentation material etc. through the web, in own education and in appropriate national and international.

2.2 Problem Definition

The purpose of this work is to explore the robustness of a Web-based system. Robustness has been defined in several ways. Thus, it's also important to have a clear definition of Web-based systems's robustness. In addition, it is important to get a rough assessment and estimation of robustness.

The main idea of the project is to answer the questions:

- How could we get an early assessment of a Web-based systems' robustness
- How could we test a Web-based systems's current robustness

The project will use AutAT [27] as a testing tool and DAIM [14] as the target system. DAIM (Norwegian: Digital Arkivering og Innlevering av Masteroppgave) system is a Web-based system which used to archive and deliver student's master thesis at NTNU. AutAT will be used to test the robustness of DAIM. Results from this work can be used to increase our understanding of DAIM's robustness.

2.3 Limitation of Scope

The WebSys [25] defines robustness as the degree to which a system or component can function correctly.

- In the presence of invalid input or stressful environmental conditions
- On a wide range of browsers

Based on the definition above, J. Zhou et al. [25] mean it's useful to split the definition into three terms:

- Error tolerance
- Stress tolerance
- Platform tolerance

Since my focus is web-based systems, I will be focusing on the error tolerance. The second and third types of robustness will not be explored. More detail about these types robustness will be described in Section 4.1.

Chapter 3

Outline of the Report

This chapter contains a brief overview of the report, giving a short description of the issues discussed in each part.

Part I - Introduction
The first part sets the focus for the project. It contains motivation, background, problem definition, limitation of scope and ends with outline of the project.

Part II - Prestudy
The second part describes the background information of this project. First, the definitions and basic concepts of robustness are described. Further, some of existing robustness assessment are presented, compared and summarized. Based on studied robustness assessment, a summary and choices of assessment methods are done. The traditional V-model of testing, an overview over different types of testing, some problems with testing of Web application and a short description of the tools that will be used to test will also be included in this part. It ends with a Goal/Question/Metric method that will be used for evaluating and analyses the robustness testing of the DAIM system.

Part III - Robustness Assessment of DAIM
This part first takes a look at the DAIM requirements. Both nonfunctional and functional requirements are captured. Chosen robustness assessments are used to assess and estimate the robustness of DAIM. A short judgement of these assessment methods will also be included here.

Part IV - Testing DAIM
This part focuses on testing of the DAIM system. It starts with description of the user testing session, which captured the process of testing. Test planning and test results are described. The purpose of this part is to analyse and evaluate the robustness of the DAIM system with respect to GQM-method.

Part V - Evaluating
The second to last part contains evaluation and discusssion of the results for this project. It is especially important to evaluate and validate the results received from the chosen assessment methods and the results of testing the DAIM system. These two re-

sults will also be compared and concluded. It ends with and discussion and further work.

Part VI - Appendix
This part contains abbreviations and glossary. A set of quality models, SFMEA and tests specification will also be included. This part ends with a summary of original document of DAIM system.

Attachment
Attachment contains test cases in AutAT, Watir. In addition, the files that I've used to run the tests, such as invalid file, or file contains invalid data will also included here. A username and password that has been used to log in DAIM system as administrator roles has been saved as a file and included here. An entire document of the DAIM system that describe requirements, ER-diagram and a description of classes will also be found here.

Part II

Prestudy

Chapter 4

Definitions and Basic Concepts

Before moving on, we need some terms from the domain of robustness. The main objective of this chapter is to define some terms which are relevant for the robustness of a Web-based system, robustness assessment, and robustness estimation.

4.1 Robustness

Robustness is defined in several ways. I have searched the Internet for definitions of robustness in order to see if they could contribute to my understanding of the concept of robustness.

IEEE [10] defines robustness as:

> The degree to which a system or component can function correctly in the presence of invalid input or stressful environment conditions.
>
> — *IEEE Std 610.12-1990*

According to [25], the WebSys consider three types of robustness:

- Input robustness: The ability to survive incorrect input from a user or from another system with which our system is cooperating.
- Load robustness: The ability to behave correctly under loads near or above the load that the system was designed for.
- Platform robustness: The ability to operate correctly on a wide variety of new platforms.

"Input robustness" has the same meaning as "error tolerance". IEEE [10] defines error tolerance as:

> The ability of a system or component to continue normal operation despite the presence erroneous input.
>
> — *IEEE Std 610.12-1990*

For WebSys, erroneous input is all input that can not be used by a system or component to perform its intended function(s).

Steven D. Gribble [8] defines robustness as follows:

> Robustness is the ability of a system to continue to operate correctly across a wide range of operational conditions, and to fail gracefully outside of the range.
>
> — *Steven D. Gribble*

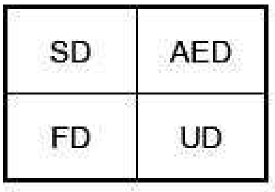
In my work, I will be focusing on the input robustness, which is defined in [25]. This type of robustness will be explored and analyzed for future conclusion.

4.2 Robustness and other quality requirements

Two important elements of robustness are specification completeness and correctness (100% reliability). A robust system operates correctly across a wide range of operational conditions. J. Zhou et al. [24] consider that a system or component which is totally correct with a complete specification is robust, in that its behavior is predictable for all possible operational envirements.

The authors in [24] introduce an operational environment partition model from [4] to formalize the difference between robustness and reliability. The total operational environment of a system or component can be divided into four parts and illustrated in Figure 4.1:

Figure 4.1: A partition over all operational conditions



- SD: The standard domain refers to the set of all operational conditions for which a system satisfies its specification
- AED: The anticipated exceptional domain denotes the set of all operational conditions for which correct exception results are produced.
- FD: The failure domain refer to all operational conditions for which the behavior of the system contradicts the specification or exceptional specification.
- UD: The unanticipated domain contains the set of all operational conditions which are not included in the specification.

Reliability is related to the failure domain. The smaller the failure domain is, the more reliable is the system. When FD= {}, the system is said to be correct, regardless of whether UD is empty or not.

A robust system requires that both FD={} and UD={} are satisfied. It's difficult to achieve this for any real system or component. Steven D. Gribble [8] argues against a seemingly common design paradigm that attempts to achieve robustness by predicting the conditions in which a system will operate, and then carefully architecting the system to operate well in those conditions. Gribble claims that this design technique is akin to precognition: attempting to gain knowledge of something in advance of its actual occurrence.

It is exceedingly difficulty to completely understand all of the interactions in a complex system a priori. It's aslo effectively impossible to predict all of the perturbations that a system will experience as a result of changes in environmental conditions, such as hardware failures, load bursts, invalid user input, or the introduction of misbehaving software. Given this, any system that attempts to gain robustness solely through precognition is prone to fragility.

Even if the system behaved correctly when operating within its design assumptions, small perturbations sometimes led to the violation of these assumptions, which in turn lead to system wide failure.

The concept of robustness in this report is similar to the one used in [8]. The authors in [8] exclude the internal faults of the system or component, and only deal with the operational, interaction-related faults.

Chapter 5

State of the Art - Existing Robustness Assessment

This chapter presents several frameworks for conducting robustness assessment for web-based systems.

5.1 Jacobson's analysis method and FMEA

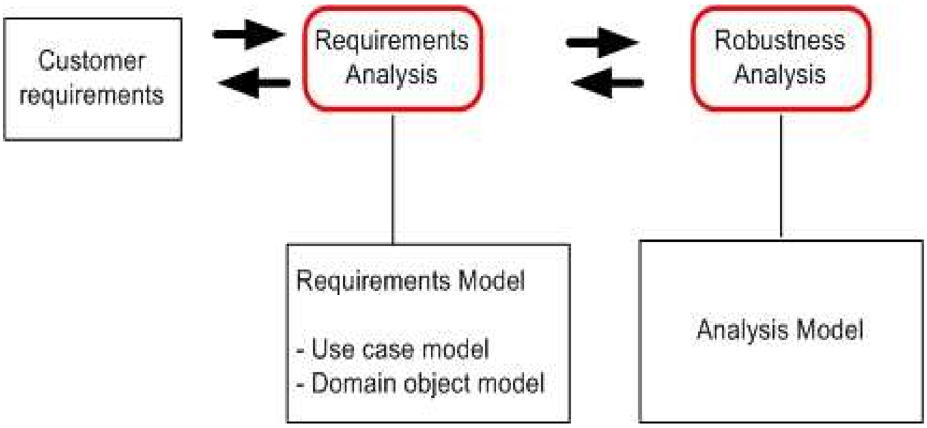
J. Zhou et al. [24] propose a framework for robustness assessment based on Jacobson's robustness analysis method and Failure Mode and Effect Analysis (FMEA) [7]. This section presents the frameworks from J. Jhou et al.[24] and T. Stålhane [19]. Both frameworks are previous works in WebSys [15] project.

5.1.1 Object-Oriented Software Engineering (OOSE)

Overview Structur of OOSE

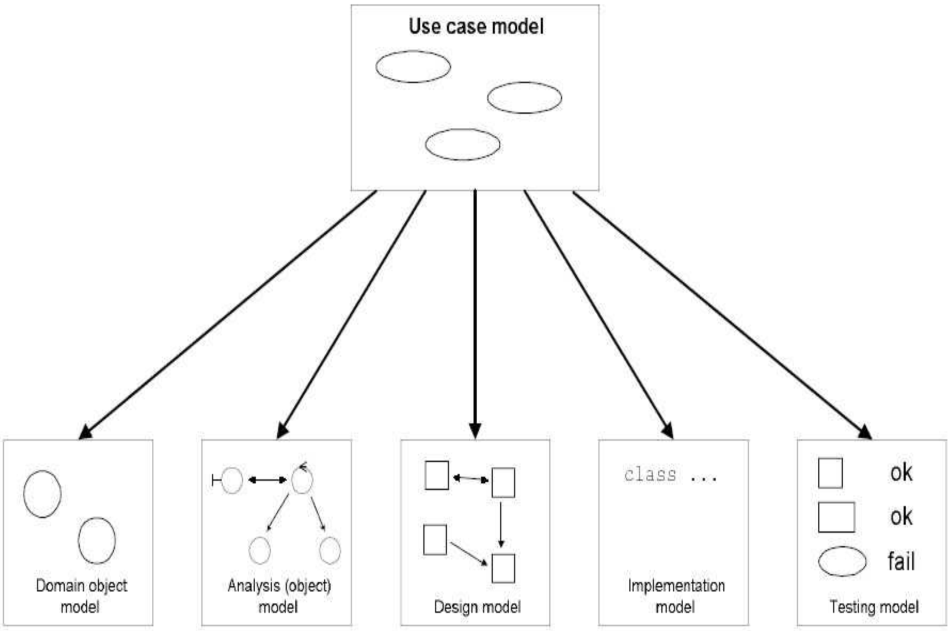
Jacobson et al. [11] call their method Object-Oriented Software Engineering (OOSE). The OOSE method is divided into three major consecutive processes: analysis, construction and testing. The analyse phase is further divided into two steps, called requirements analysis and robustness analysis. Figure 5.1 shows the analysis phase of the OOSE approach. The first step derives the requirements model from the informal customer requirements. This model is expressed as a use case model, and may be augmented by a domain object model. The second step, robustness analysis structures the use case model into the analysis model.

Figure 5.1: Analysis phase of the OOSE life cycle



The OOSE method defines a process to transform formalized requirements into a sequence of models. The steps include the requirements, analysis, design, implementation and testing models. The use case model is the basis on which all other models are developed, as illustrate in Figure 5.2. Together with the domain object model it forms the requirements model.

Figure 5.2: The use case model of the OOSE approach



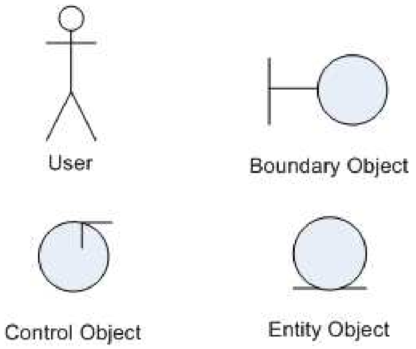
The domain object model consists of the objects found in the problem domain. These objects can be structured with the inheritance and aggregation relationships.

Robustness analysis is an intermediate level of design, between use cases and the software design level. By analyzing each use case, robustness analysis identifies a set of objects that will participate in the use case. The analysis model is based on typed objects. The three object types are entity, control and boundary. The purpose of the typing is to support the creation of a structure that is adaptable to change. Thus, for example, changes to the interface requirements can be limited to interface objects.

- Entity objects model information that exists in the system for a longer time, typically surviving a use case. Domain objects often become entity objects, but this is not necessarily true. Entity objects can be structured with inheritance and aggregation relationships as described above for domain objects.
- Boundary objects model behavior and information related to the presentation of the system to the outside world. The actors use boundary objects when communicating with the system.
- Control objects model functionality that is not naturally tied to the other object types. They server as the "glue" between boundary objects and entity objects. The control object could, for example, operate on several entity objects, perform a computation and return the result to an interface object that would represent it to the user.

Figure 5.3 shows how to represent these three types of objects in a robustness diagram.

Figure 5.3: Stereotype symbols

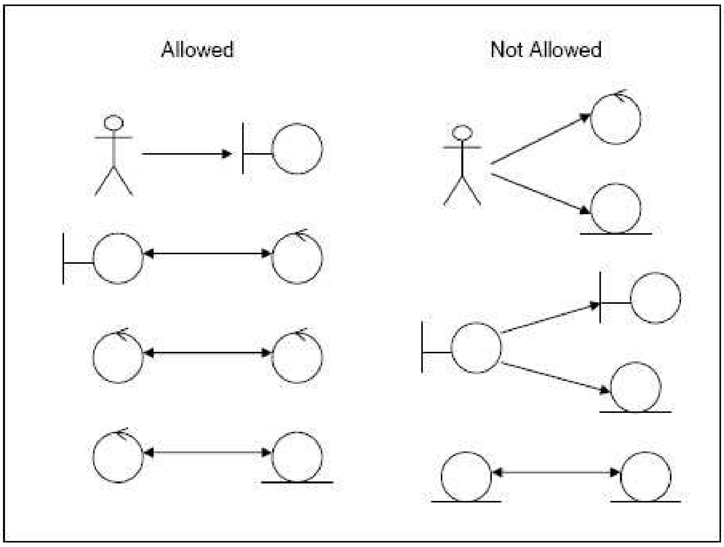


Rules for interaction among these objects are illustrated in Figure 5.4.

- Actors can only talk to boundary objects
- Boundary objects can only talk to Control objects and Actors
- Entity objects can only talk to the Control objects
- Control objects can talk to boundary objects, other Control objects and Entity objects, but not to Actors.

A closer look at the three types of objects when they applied for web-based system are shown below:

Figure 5.4: Interaction rules



- Boundary objects are the objects that the users will use to interact with the system. These are elements that compose a web page, such as hypertext, forms, menus, buttons, etc.
- Entity objects often map to the database tables and elements in legacy systems. They represent resources required by a use case execution.
- Control objects embody mostly application logic. They serve as mediators between the users and the stored data. This is where one captures the frequently changing business rules and policies.

According to J. Zhou [24], the Jacobson's analysis method provides a systematic method for decomposing the system into objects. In addition, the control objects capture application logic and manage all interactions between boundary object and entity objects, they serve as natural placeholders for robustness assessment using the FMEA.

5.1.2 Failure Mode and Effect Analysis (FMEA)

The FMEA discipline was originally developed in the United States Military. FMEA has been widely adobted and has become standard practice in Japanese, American, and European. Goddard [7] stated that FMEA is a traditional reliability and safety analysis technique that has enjoyed extensive application to diverse products over several decades.

A systematic thinking promoted by FMEA is relevant when a new product or system is developed. J. Zhou et al. [24] use FMEA to perform robustness assessment early in the web-based system development process. For each component or subsystem we can analyse the failure modes, their causes and effects on the rest of the system. FMEA seeks answer for questions like:

- What could go wrong with this component or subsystem?

- How badly might it go wrong?
- What needs to be done to prevent failures?

The Software FMEA can generally be classified as either product FMEA or a process FMEA depending upon the application. The product FMEA analyses the design of a product by examining the way each item's failure modes affect the operation of the product. The process FMEA analyses the process involved in design, building, using and maintaining a product by examining the way that failures in manufacturing or service processes affect the operation of the product. Both FMEA types focus on design; either design of the product or design of the process.

5.1.3 The proposed method

Classes as Starting Point for FMEA

T. Stålhane in [19] proposes to use classes as starting point for FMEA for robustness analysis and assessment. The analysis process will result in one FMEA table for each class, organised according to the methods in the class. The FMEA table for robustness is illustrated in Table 5.1.

Table 5.1: FMEA with class as starting point

Class id:					
Method id.	**Failure mode description**	**Seriousness**	**Avoidandce cost or feasibility**	**Robustness consequences**	**Indicators**
(1)	(2)	(3)	(4)	(5)	(6)

The table headings from Table 5.1 have the following interpretations:

- Method id: An id of a method in the class identified in column (1).
- Failure mode description: A short description of the failure mode under consideration is described in column (2).
- Seriousness: An estimate of how serious this failure mode is. The main purpose of the column (3) is to help us to assign a priority to the construction of barriers against each failure mode. The entry can be for instance high (H), medium (M), or low (L) or a numeric score, e.g. from 1 to 10.
- The column (4) can be one of two:
  - Avoidance cost: The cost of removing the failure mode by constructing a barrier.
  - Avoidance feasibility: The feasibility of creating a barrier to avoid this failure mode. The entry can be as for seriousness in column (3).

- Robustness consequences: How will this failure mode contribute to lack of robustness for the system? This entry is recorded in column (5).
- Indicators: Events or relationships that can be observed if the faiure occurs. This entry is described in column (6).

Before moving on, we need to define the terms failure mode and barrier.

- Failure mode: A way that a component can fail. A failure mode is not an absolute entity. Both "wrong value" and "value too low" are acceptable failure mode but they are on diffent level.
- Barrier: An activity, a process or a component that is inserted in order to stop or reduce the effect of a local failure.

We can start to identify and construct the barriers to the failure mode when we have filled in the FMEA tables for all classes. This way of defining robustness was originally proposed by T. Stålhane [19]. T. Stålhane assess robustness as follows:

$$Robustness = \frac{FMwB}{TFM} \tag{5.1}$$

FMwB: Number of Failure Mode with a Barrier
TFM: Total number of Failure Modes

This is the simplest way to assess robustness, but this way assumes that these barriers have 100% success when they are stopping/reducing the effect of local failure. The effectiveness of a barrier or the feasibility of constructing a barrier is used to get a better robustness estimate. Barrier effectiveness can be interpreted as a probability that the barrier will handle an input or system state without impairing robustness.

$$P(barriersucceeds) = \frac{REH}{TRE} \tag{5.2}$$

REH: Number of Risky Events Handled
TRE: Total number of Risky Events

Definition 5.2 has a precise statistical meaning since it gives the probability that the barrier succeeds. The barrier effectiveness (BE) can also assess in a more informal, quantitative way by expressing as follows:

$$BE = \frac{CQoB}{MQoB} \tag{5.3}$$

CQoB: Current Quality of the Barrier
MQoB: Maximum Quality of the Barrier

The current quality of the barrier will be a function of the resources used to construct it. The barrier's effectiveness against a certain failure mode (FM) will be denoted as BE(FM). T. Stålhane [19] discusses also how BE can express as a mathematical expression. Further, the author concludes with a more refined expression for robustness assessment. Let the index i denotes the failure mode number. The robustness assessment can then be expressed as follows:

$$Robustness = \frac{\sum(FM(Seriousness)_i * BE(FM_i))}{\sum FM(Seriousness)_i} \tag{5.4}$$

This expression can be used in several ways, for instance to assess the robustness of a class, a system, a function or a scenario. In each case, it's just a question of which methods or classes are included in the sum. The expression for robustness can be used to:

- Compare alternative ways to use a certain number of resource or the effect of using different amounts of resources.
- Rank robustness problems according to feasibility and effect of avoidance techniques.

Control Object as Starting Point for FMEA

The FMEA worksheet proposed by J. Zhou et al. [24] is a revised version of one described in [1]. The authors from [24] focus on identifying means to eliminate or reduce the chance of robustness-related failures. The entries in the FMEA worksheet are illustrated in Table 5.2.

Table 5.2: FMEA with Control Object as starting point

System: Use case:			Performed by: Date:		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
(1)	(2)	(3)	(4)	(5)	(6)

The headings from Table 5.2 can be described as follows:

- Control Object: The name of the Control object is given in column (1).
- Robustness failure mode: All robustness-related failure modes of this control object are identified in column (2). A robustness failure is defined as non-fulfillment of a robustness requirement.
- Possible cause: Possible cause of the failure mode is described in column (3). Since I have focus on robustness, only those stemming from outside the boundary of the objects are rated as causes of robustness failure.
- Local effect: The main effect of the identified failure mode on the subsystem (the function of the use case) is recorded in column (4).

- System effect: The main effect of the identified failure mode on the primary function of the system is recorded in column (5).
- Preventive means: Possible ways to prevent or reduce the effect of identified robustness failure are described in column (6).

The authors in [24] have choosen the FMEA method to do robustness assessment at the analysis and architecture design stages. The difficulty is that little information is available at the early stage. To do an FMEA, it's necessary to decompose the system into well-defined components, and then do an FMEA on each part. Jacobson's analysis method could be used to support such a decomposition and analysis in a practical and systematic way.

A five-step method is developed by J. Zhou et al. [24], using Jacobson's analysis model and FMEA for Web-based system robustness assessment as follows:

- Step 1: Define the robustness requirements of the system.
- Step 2: Divide the system into subsystems by focusing on the important use cases. For each use case, perform Jacobson's analysis and identify Boundary objects, Control objects, and Entity objects. Complex logic can be partitioned into several Control objects.
- Step 3: Prepare a complete list of Control objects for each use case.
- Step 4: For each Control Object, fill in the FMEA worksheet, which is showed in Figure 5.2.
- Step 5: Review failure modes in the FMEA worksheet and prioritize those items that are pertaining to a particular robustness goal.

5.1.4 Robustness Assessment: An Example

An example from [24] is described in this section to illustrate the proposed method. A minimum robustness level is defined by that a system should always responds to the user's action either by correct results or appropriate prompt. In order to keep it short, I've present the robustness assessment for one use case only. Figure 5.5 shows the result of using Jacobson's analysis diagram for the selected use case "Search by author".

User types the name of an author on the "Search Page" and then presses the Search button. The system searches the Catalog and retrieves all the Books with which that author is associated. The system then retrieves the important details about each book and displays the list of Books on the "Search Results Page".

The application logic of this use case is captured by "Search on Author", "Retrieve Details" and "Display" Control objects. The FMEA worksheet for "Search on Author" Control object is shown in Table 5.3. The "Retrieve Details" and "Display" Control objects are not presented in the Table 5.3, see [24] for an entire FMEA worksheet of the use case "Search by Author".

Figure 5.5: Analysis diagram for Search by Author

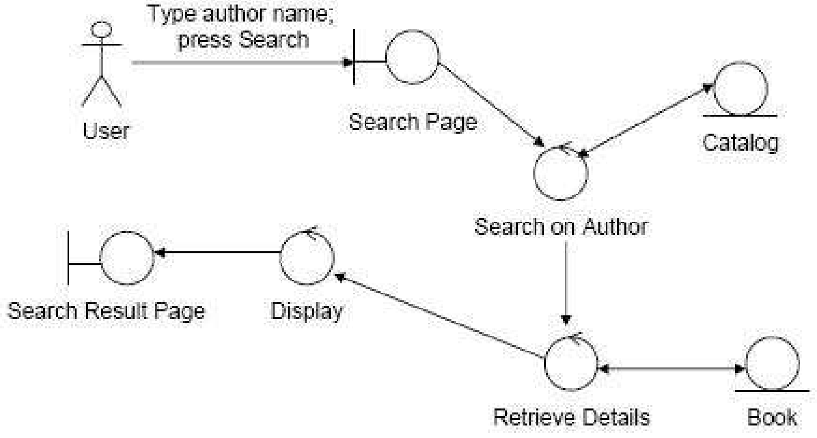


Table 5.3: A part of FMEA for Search by Author

System: Use case:Search by Author			Performed by: Date:		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Search on Author	No response is produced at all	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. "Search Page" detects the failure of the object and interacts with "Display" to prompt the user appropriately
	Information found is incorrect	Error user input or incorrect content in "Catalog"	Incorrect data is presented to the user	Users move to other systems if they suspect the quality of the system	Control user input and prevent serious errors from entering the object; manage data in "Catalog" and ensure its correctness

5.1.5 Conclusion

A report from T. Stålhane [19] in the WebSYS project uses classes as a starting point for FMEA. In addition, the robustness assessment can also be estimated by using a mathematical expression. The expression for robustness can be used to compare alternative ways for resources or to rank robustness problems. This method will be used in combination with other framework to give a quantitative robustness assessment.

J. Zhou et al. [24] have presented an approach to early assessment of robustness for web-based software system. The main goal of this assessment is to identify measures that will increase the system's robustness. They believe that the reliability and robustness are distinguished only by different types of faults. The Jacobson's analysis describes the robustness assessment for each use case and then goes into details to show the application logic of this use case. A weak point of the Jacobson's analysis model and the FMEA is that both of them do not enable explicit references the component whose the quality-carrying properties affect the system quality attribute.

5.2 Information flow analysis and hazard analysis

Y. Zhang el al. [23] have also suggested a framework for modelling the quality of web-based information system. The authors in [23] describe an attempt to develope a quality modelling method based on a combination of information flow analysis and a hazard analysis. This method can also use for assessment of robustness.

5.2.1 Information Flow Analysis

Information flow analysis examines the use of information in specific business processes that are within the scope of a systems investigation. It views the activities of a system from the viewpoint of the information, such as where the information orginitated, how it is used and processed, and so on.

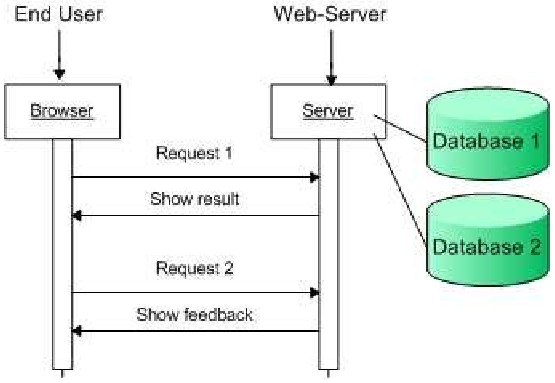
The information flow diagram is used to analyze the information flow. The diagram will show the movement of information through a system. Information flow diagram also permits the system analysts to focus their interests on certain points, for example, the system's current robustness.

Information drives the information system activities. It can trigger events and can be processed to provide information for use. An information flow can be seen as a "packed" of information passing between system component and/or across the system boundary between the system and its environment.

The information flow diagram can have several levels, some providing overviews of major processes and others going into detail to show each step of information movements. This enables analysis of an information system at several levels of abstraction and let us consider quality model at several levels of details.

Different information systems should use different information flow diagrams, for example sequence diagram from Unified Modeling Language (UML) [6] and data flow diagram from A. Sølberg et al. [20]. Sequence diagram is used as an information flow diagram and illustrated in Figure 5.6

Figure 5.6: An example of information flow diagram



5.2.2 Hazard analysis

Software hazard analysis is a software quality assurance activity that focuses on the identification and assessment of penitential hazard, which may cause a system to fail. Y. Zhang et al. [23] use FMEA to consider failure modes of each component within a system, see Table 5.4.

Table 5.4: FMEA format for request

Request:					
Ref No.	**Failure modes**	**Possible Cause**	**Local Effects**	**System Effects**	**Relevant Quality Attributes**
(1)	(2)	(3)	(4)	(5)	(6)

The FMEA version is somewhat similar to the FMEA from [24]. They differ by column (1) and (6). See interpretations of the Table 5.2 for description of columns (2-5). The columns (1) and (6) are described as follows:

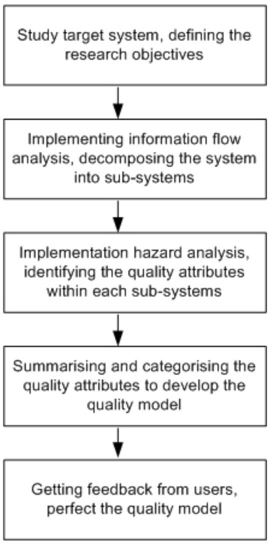
- Ref No.: Reference number for request is identified in column (1).
- Relevant Quality Attribute for failure modes are described in column (6).

The process first selects the individual components or functions within a system. It then makes assumption about the failure modes of each component. Next, it considers the causes of the failure, and determine its ultimate consequences.

5.2.3 The Process of Quality Modelling

The purpose of quality modelling is to identify the quality attributes relevant to the target systems and the relationships between the quality attributtes and systems components. Figure 5.7 outlines the modelling process. The quality modelling process is split into three steps as follows:

Figure 5.7: Quality Modelling Process


Study target system, defining the research objectives
Implementing information flow analysis, decomposing the system into sub-systems
Implementation hazard analysis, identifying the quality attributes within each sub-systems
Summarising and categorising the quality attributes to develop the quality model
Getting feedback from users, perfect the quality model

1. System activity modelling: Study the current systems, in order to express the information processing activities and processes that occur.

2. System structure modelling: Decompose the whole system processes into several sub-systems using information flow diagram.

3. Hazard analysis: Identify the quality attributes, using hazard analysis, within each process. These quality attributes can be further decomposed, employing lower-level information flow diagrams.

5.2.4 Conclusion

Y. Zhang [23] proposed a method to systematically derive quality models from architectural designs of information systems. The authors in [23] leave it to the practioners and researchers to develop their own quality models for individual systems. They adapted hazard analysis methods to enable software engineers systematically identify certain types of quality attributes and the quality carrying properties of each component and connector, and to establish the links between them.

5.3 Application of Hazard Analysis to Software Quality Modelling

H. Zhu et al. [26] propose a systematic method for constructing quality models of information systems. A diagrammatic notation is devised to represent quality models that enclose application specific features. A Software FMEA is adapted to deriving quality models.

5.3.1 Representation of quality models

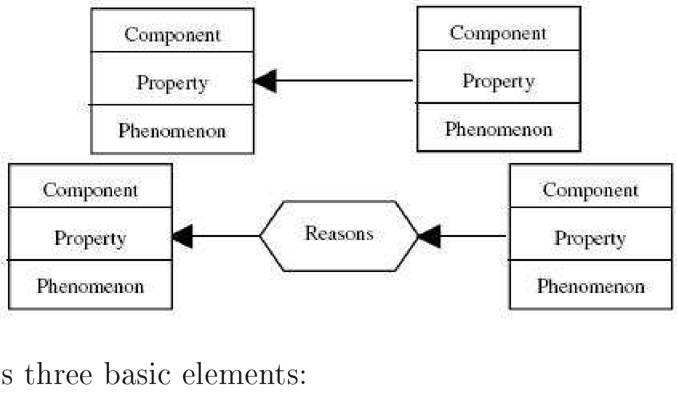
The quality model in [26] is built based on Dromey's principle [5]. Dromey's generic quality model consists of three principal elements: product properties that influence quality, a set of high-level quality attributes, and a mean of linking them. H. Zhu et al. [26] argue that how a quality-carrying property of a component is related to a quality attribute of the system is important because it provides insight that can significantly improve the usability of the quality models.

The link between quality attributes/quality-carrying properties, such as robustness and reliability, cannot be easily established or validated. However, abstract properties usually demonstrate themselves through concrete events and observable phenomena, which are tangible and observable. The relationships between observable phenomena are often self-evidence in the context of the system, and also easier to establish and validate than abstract properties. This provides the crucial information for software testers to develop test cases to check if the system implements as designed. Thus, authors in [26] has the following requirements for the representation of quality models:

- Requirement 1: Explicitly associate quality attributes/quality-carrying properties to the components of the system.
- Requirement 2: Associate abstract properties with observable and verifiable phenomena of the components/system
- Requirement 3: Be able present the rationale of the relationships between the properties. Such rationales can be system specific and should be able to be verified and validated in the context of the system.

Figure 5.8 illustrates the diagrammatic representation of quality model from [26]. The model is a directed graph.

Figure 5.8: Notation for representaion of quality models



Each node contains three basic elements:

- The component of the system
- The quality-carrying properties of the component
- The observable phenomena of the property

5.3.2 Adaption of Hazard Analysis Method

H. Zhu et al. [26] use FMEA to identify a system's potential failure modes, their possible causes and the consequences. Authors in [26] claim that the original FMEA table is ambiguous about which component causes the failure. Therefore, the FMEA is modified so that the component that causes a failure becomes clear as illustrated in Table 5.5.

Table 5.5: The format of Software FMEA chart

Software FMEA for Web-based Information System					
No.	**Failure modes**		**Possible Cause**		**Explanation**
No.	**Component**	**Phenomena**	**Component**	**Fault/Failure mode**	**Explanation**
(1)	(2)	(3)	(4)	(5)	(6)

The table headings of Table 5.5 have interpretations as follows:

- No: The id of the component which can result in failure mode. The id is identified in column (1)
- Failure mode: Each failure mode in the table forms a node containing a component and a phenomena, which are recorded in column (2) and (3).
- Possible cause: Each row in the chart forms a link from the node that represents the cause to the node that represents the failure mode. Possible cause consists of component and fault/failure mode, which are described in column (4) and (5).
- Explanations: The explanations in column (6) gives the reason of the link.

5.3.3 The proposed method

In [26], the failure mode of a use case will first be illustrated by a SFMEA. In the application of SFMEA, each of the causes and consequences of a failure mode become a new entry to the chart. The causes and consequences are further investigated until the cause is primitive and consequences are terminal. A failure mode is primitive if it is caused by a fault of a component and its causes cannot be further identified without additional knowledge about the system. A failure mode is terminal if it does not effect any other component of the system or does not cause any other failures.

Then, the construction of a quality model takes the information charted in the SFMEA as input. Each failure mode in the chart forms a node with the component and phenomenon as specified in the SFMEA chart. Each row in the chart forms a link from the node that represents the cause to the node that represents the failure mode. The explanation column of the row gives the reason of the link.

5.3.4 Robustness Assessment: An Example

Let's consider an example to illustrate the use of the proposed framework. This example is taken from [26]. Table 5.6 shows the failure modes for the case that cannot find the required information on the Web page.

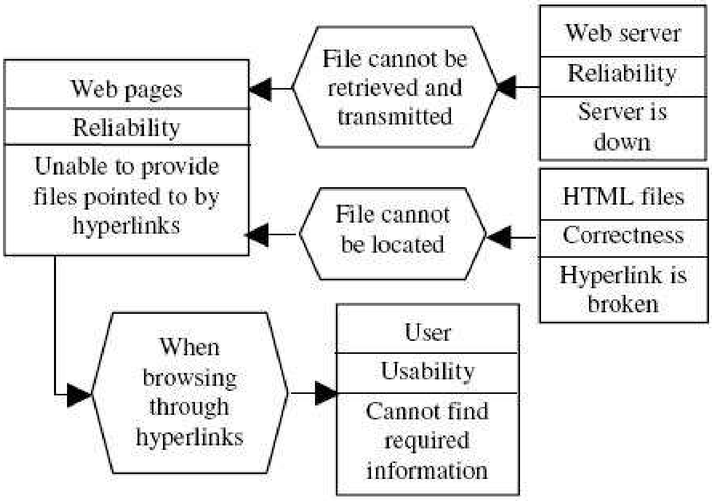
Table 5.6: The SFMEA for "User cannot find required information"

Software FMEA for Web-based Information System					
No.	**Failure modes**		**Possible Cause**		**Explanation**
No.	**Component**	**Phenomena**	**Component**	**Fault/Failure mode**	**Explanation**
1	The user	Cannot find required information	Web page	Unable to obtain a file through a hyperlink	When the user searches for information by browing through hyperlinks
2	Web page	Unable to obtain a file through a hyperlink	HTML files	This link is broken	The file cannot be found due to the broken of the link
3			Web server	Server is down	The file cannot be retrieved and transmitted

Figure 5.9 shows the quality model for the case that user cannot find the required information on the Web page.

For each node in the diagram, the observable phenomenon is compared with the definitions of a set of quality attributes and quality-carrying properties of the components. For example, "a hyperlink is broken" demonstrates the quality attribute correctness. "Server is down" is related to the reliability of the system.

Figure 5.9: The model for user cannot find required information


Web server
Reliability
Server is down
File cannot be retrieved and transmitted
Web pages
Reliability
Unable to provide files pointed to by hyperlinks
HTML files
Correctness
Hyperlink is broken
File cannot be located
When browsing through hyperlinks
User
Usability
Cannot find required information

5.3.5 Conclusion

To conclude, H. Zhu et al. [26] introduce a quality model that enables explicit references between quality attribute and quality-carrying properties. It also enables the explicit annotation of the reasons why two properties or attributes are related. Moreover, the FMEA is adapted so that the component that causes a failure becomes clear.

Causes and failure modes are further investigated until they are primitive. This require details information. Furthermore, the relationships between system's component and subcomponent must be available, since it will be used in the quality model and FMEA table. This framework, however, is not directly related to the robustness, but it can be used for assessment the robustness and other quality attributes that have influence on the robustness.

5.4 Summary and the Choice

Evaluating the frameworks assessment is not a major task in this project. Thus, the studied frameworks will be compared at a high level. This section presents the summary and the choice of assessment methods that will be used to assess the DAIM system.

5.4.1 Summary

I have looked at some of todays frameworks for early assessment of software quality. There are two basic categories of frameworks: those that suggest quantitative measurement as in T. Stålhane [19] and those that generate qualitative assessments, such as [24, 23, 26] have done. Further, these frameworks are separated by the way they decompose the system, the quality model and the version of the FMEA they use. The described frameworks are in many ways similar. They all used FMEA to find failure modes, possible causes and their affects.

The author in [19] proposed several ways to assess robustness. T. Stålhane has introduced the definitions of failure mode, barriers, and effectiveness. There are several ways to assess the robustness, which are dependent on the implemented barriers and effectiveness.

The method from J. Zhou [24] uses Control Object as a startpoint for FMEA, and does not give a quantitative estimating of the robustness. This approach can be used for an early assessment of robustness for web-based software systems. J. Zhou et al. believe that the reliability and robustness are distinguished only by different types of faults. Thus, the reliability techniques can be applied to improve the robustness of the system. The proposed framework integrates Jacobson's analysis model [11] and the FMEA. As a result, a five-step methodology is developed by J. Zhou [24] and will identify preventive measures against robustness failures, which in turn will contribute to specification enhancements.

The method from Y. Zhang et al. [23] is somewhat similar to the method from J. Zhou [24]. They are separated by the way they decompose the system. [23] uses information flow diagrams that enables analysis of an system at several levels of abstraction and consideration of quality models to various levels of detail. The authors in [23] let practitioners and researcher develop their own quality models for individual systems. The adapted FMEA enables to identify the relevant quality attributes.

The framework from Hong Zhu et al. [26] is distinguished from the other frameworks by the way they identify the quality model. The quality model enables explicit references to the components whose quality-carrying properties affect the system quality attribute. It also enables the explicit annotation of the reasons why two quality attributes are related. Moreover, the Software FMEA (SFMEA) is adapted and can be directly used to construct quality models of information systems. It provides the logic that bridges the gap between abstract system quality attributes and the tangible quality-carrying properties, the observable behavior of the system and their components.

In conclusion, all the frameworks described in this chapter can be used for assessment for software quality in Web-based informations system. The frameworks from [19, 24] are described for robustness assessment. The frameworks from [23, 26] are described for software quality modelling, and can be used for a specific quality requirement, such as robustness. Moreover, these frameworks can be used for considering other quality attributes that related to the robustness. Both quality model and adapted SFMEA from [26] require detail information, which is not suitable for early assessment method, where it's still lack much information.

5.4.2 The Choice of Assessment method

In order to get an assessment of a Web-based systems' early robustness, we have to use both quantitative and qualitatative techniques. There are three frameworks which we should try out in practice. Each framework is suitable for one or several specific phases in development process. These frameworks will not work equally well in every phases. Thus, to obtain the best effeciency, each framework must be used in the phase when it is most suitable.

First, J. Zhou et al. [24] used the Jacobson's analysis method to model the use cases, sub-use cases or user stories, which is suitable for both levels in detail and high level. Moreover, the framework from [24] can use as an early assessment method at system requirements, analysis and architecture design stages. It also provides a practical way to help one address all the necessary scenarios in the use case. Moreover, the identified objects and the essential relationship between the three stereotypes enable us to conduct robustness assessment.

Then, for each module, a simplied FMEA version is applied to find robustness-related failure modes, possible causes, their effects, and possible ways to prevent or reduce robustness failures. This framework will be used to assess the robustness of DAIM system.

Second, the proposed quality model and adapted SFMEA that H. Zhu et al. [26] have described. There are several advantages of the method proposed in [26]. First, it enables software engineers to derive quality models at an earlier stage of software development. This is important since it creates the awareness of the required quality attribute. Second, the quality models include the abstract properties, attributes, and observable phenomena of the components of the system, the rationale of the links between the phenomena. This makes the software quality models testable and verifiable.

Because of time pressure, I'll illustrate the framework from H. Zhu [26] by consider some use cases in DAIM. This will be done to get an idea of how this framework works and to see if it can be used as an assessment method for robustness assessment. The framework will be evaluated and concluded, but not used to assess the robustness of DAIM. The robustness of DAIM will be estimated, based on the work that used framework J. Zhou et al. [24] have proposed.

Both of frameworks in [24, 26] give a qualitative way to assess the robustness, but this is not enough to assess current robustness of a Web-based system. The robustness could be estimated quantitative by the mathematical expression as described [19]. T. Stålhane et al. [19] proposed a simple robustness expression to assess the robustness of a class, a system, a function or a scenario. Thus, framework from [24] is used to get a qualitative robustness assessment. Then, based on the Jacobson's analysis method and FMEA from this framework, the changes will be done and described in section 9.3 is used to give a quantitative way for estimating the robustness. I believe that a combination of Jacobson's analysis, FMEA and a mathematical expression should be the methods which the robustness of a Web application should be estimated.

Chapter 6

Testing

Testing the input robustness of the DAIM system is important part of this project. The results from assessment methods will be evaluated and validated by comparing with the results from testing.

In general, the software testing techniques that are applied to Web-based applications are the same as those that are applied to other applications [9]. Both cases require basic types such as functionality tests, forced-error tests, acceptance test, and so forth. Some differences between the two are that the technology variables in the Web environment are multiple, and the additional focus on security- and performance-related tests, which are different from feature-based testing. The testing of a software system is performed at several levels and at several times during the development process.

Testing is an essential part of any software development. Even though a program does what it is intended to do, it might still be a possibility that there exists errors that makes the program do things it is not supposed to do. A test can not prove that an error is not present, it can only prove that an error is present. Even though a progran does what it is supposed to do, it might still be full of errors if it also does things it is not supposed to do.

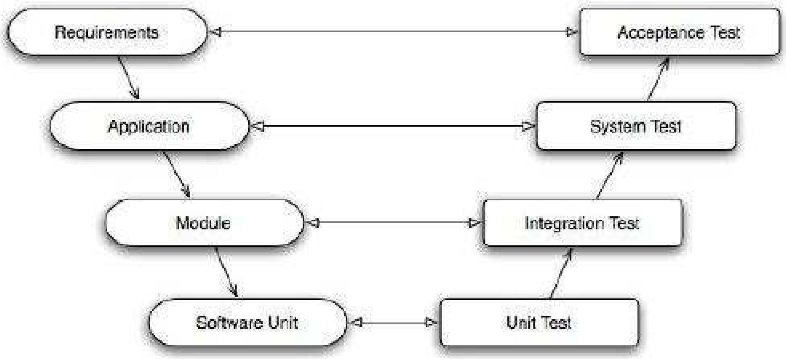
Testing is the process of executing a program with the intent of finding errors [13]. This definition of testing leads to the realization of what constitutes a well designed test. A well designed test is one that has a high probability of proving the presence of errors in the code it tests. A successfull test is one that finds bugs. On the other hand, testing is to show that we have delivered a product as promised, verify that the user requirements have been achieved, and to provide confidence in the functionality of the system.

The first section of this chapter takes a look at V-model of testing. The next section describes different types of testing. The chapter ends with a distinct of some characteristics of testing of Web applications.

6.1 V-model

The V-model [18] was originally developed from the waterfall software process model and is shown in Figure 6.1. This model shows a traditional view of how to develop an application and how testing relates to development.

Figure 6.1: V-model of testing



Any software development process starts with a set of requirements from a customer. The requirements describe what the customer wants to achieve by development the software product. Acceptance tests are created to make sure that the system does what the customer wants and are used by customer to validate that the system actually does what he/she wants it to do. The goal of acceptance testing is to verify that the user requirements have been achieved, and to provide condidence in the functionality of the system.

System testing will compare the system specifications against the actual system and ensure the system behaves as prescribed. Test cases are derived from the systems use cases, and the functionaliy of the system is tested based on these tests. System testing demonstrates that the system works end-to-end to provide the business functions specified in the high-level design.

Software systems will be divided into several modules or subsystems. A module is a part of the system responsible for providing a part of its total functionality. The decomposition of the system into modules lead to a set of integration tests. In integration testing the separate modules will be tested together to find weaknesses and bugs in the system. Integrating testing demonstrates that two or more units or other integrations work together properly, and tends to focus on the interfaces specified in low-level design.

A module can also contain several smaller modules or parts. These parts are called components or units. The unit tests are performed on the units while the programmer is coding them. A unit test will check that the unit performs correctly in isolation, such as its state is correct after operations are executed. Unit testing is code-based and performed primarily by developers to demonstrate that their smallest pieces of executable code function suitably.

The V-model is valuable because it highlights the existence of several levels of testing and depicts the way each relates to a different development phase. Starting from the requirements, the system is developed one phase at a time until the lowest phase, the implementation phase, is finished. At this stage testing begins, starting from unit testing and moving up one test level at a time until the acceptance testing phase is complete [18]. Accordingly, there does not seem to make much sense to perform any of the tests at a higher level before all the tests at lower levels are passed, i.e. a integration test of a module will never be expected to pass when some of the units in the module do not pass their unit tests.

6.2 Types of testing

6.2.1 Black Box Testing

Black box testing [17] focuses on software's external attributes and behavior. Black box testing consists of a set of input values and expected output values. The results of running the software with the input values are compared to the expected output values. If they match, the test pass, if not it fails.

In testing, several inputs are exercised and the outputs are compared against the specification to validate the correctness. All test cases are derived from the specification and no implementation details of the code are considered. It is obvious that the more we have covered in the input space, the more problems we will find and therefore we will be more confident about the quality of the software.

The black box testing focuses on how to maximize the effectiveness of testing with minimum cost, usually the number of test cases. It is not possible to exhaust the input space, but it is possible to exhaustively test a subset of the input space. Partitioning is one of the common techniques. If we have partitioned the input space and assume all the input values in the partition is equivalient, we only need to test one representative value in each partition to sufficently cover the whole input space. Black box testing can be less effective at uncovering certain error types, such as data flow errors or boundary condition errors at the source level.

6.2.2 White Box Testing

White box testing [17] is another way of testing software. Contrary to black box testing, software is here viewed as a white box. In white box testing, the structure and flow of the software under test are visible to the tester [17]. White box test aims to execute all statements in a part of the software. Sometimes white box testing is also referred to as logic-driven testing since the test design is driven by knowledge of the software's internal logic when designing the tests. White box testing is also called glass box testing or design based testing.

There are many techniques available for white box testing, since the problem of intractability is eased by specific knowledge and attention to the structure of the software under test. One goal when designing white box tests is to achieve exhaustive path testing. This means executing all possible paths through a program. A program will usually have many points where a logic decision (i.e. an if-statement) determines the further execution. However, one can design tests that will produce both a true and a false outcome at each branch point. This will lead to all branches in a program being executed at least once.

One problem with white box testing is that it is quite possible to test all paths through the software, but if you ignore the specification, the software might still be erroneous. An exhaustive path coverage test may not show that the software actually produces the results it should according to its specification.

6.3 Testing of Web Applications

Developing web applications introduces new testing challengs [9]. These challenges come both from the architecture of the web applications, the results of users using different web browser and the high focus on user interface in the applications.

A Web application is based on a client-server architecture. The user interacts with the application though a web browser, sending requests and receiving responses from the server. The server processes all the business logic in the application while the client just shows the results to the user. Layering of Web applications make testing of these applications harder.

Most research on testing Web application has focused on client side validation and static server side validation. There exists some difficulties with both of these tests. When a component must be deployed in an application server to work, it is difficult to unit test the component. It needs to have a running instance of the application server, and the database must be set up. The order of which each test is performed will influence the results. For instance, in order to test that a user can log on, the user must be registered in the system before the test can be executed. The execution order of two tests also make impact. The state of the system might be changed by the first test, affecting the second. This is a problem with all systems that maintain state across different interactions, not only Web applications.

Also the client side must be tested. Traditionally, client side testing has been performed manually, typing values into forms, reading text and clicking on links and buttons and so on. Such testing is time-consuming and error prone. A tester need to type in all input text, and manually move the mouse around to click on links. He/She must also wait for the responses from the web server to reach him. When typing values, it is easy to make mistakes, and it is easy to miss a typing error when reading text.

Moreover, different browser or different versions of the same browser have different capabilities. The layout may vary, text may be misplaced and images may be located at different places. In addition, many Web applications use JavaScript as part of interaction. JavaScript capabilities are different in different versions. Due to the different capabilities of different browser, at least some tests must be run using several browser.

A common technique in Web applications is to perform input validation on the client with scripting languages such as JavaScript. An insidious problem with client-side input validation is that end users can bypass this validation. Bypassing validation can cause failures in the software, and can also break the security on Web applications, leading to unauthorized access to data, system failures, invalid purchases and entry of bogus data. Thus, there is need to create client-side tests for Web application that intentionally try to violate explicit and implicit checks on user inputs.

6.4 Tools for Automatic Acceptance Test

An acceptance test is a formal test conducted to determine whether or not a system satisfies its *acceptance criteria* from a user's point of view and enable the customer to determine whether or not to accept the system [21]. Acceptance tests contribute two things to a software project. First, they capture the system's requirements in a directly verifiable way. Second, they expose problem that other type of technically oriented tests miss, i.e. bugs that are not found by unit tests, even if the unit tests provide a full code coverage.

Many tasks of the acceptance tests are mundane and highly repetitive, such as filling in forms and validating their results. Furthermore, running all acceptance tests is a bottleneck before product delivery, and can take several weeks. This affects the time required to find all bugs. That's why automated acceptance tests considered as a must-have. Automated tests can also be run more often in the development process, thereby improving product quality.

Automating acceptance tests tend to be difficult. An automated acceptance test framework is not only affected by the programming language used, but by the particular interface, maybe the window system, operating system. A large variety of tools, frameworks, products and techniques for automating acceptance tests are availabe, i.e. FAT[1], FIT[2], FitNesse[3].

This section presents AutAT and Watir that will be used to test DAIM. AutAT is an open source Eclipse plugin to enable test driven development of web applications. Input robustness of DAIM is the only one quality requirement that will be

[1] http://sourceforge.net/projects/fat

[2] http://fit.c2.com

[3] http://fitnesse.org

test. Thus, the chosen tools will support the automatic acceptance test at the client side.

The program AutAT is chosen and recommended by my supervisor, Tor Stålhane to test the DAIM system. Research and evaluating of existing automatic acceptance test tools are not included in this project.

6.4.1 AutAT

The AutAT tool was developed at Norwegian University of Science and Technology (NTNU) in Trondheim, in cooperation with Bekk Consulting AS (BEKK). BOSS[4] is BEKK's Open Source Software website where they host information about their Open Source projects. There are several interesting aspects related to the AutAT tool, like the testing framework, Eclipse and GEF, see [27] for more information about AutAT. The Eclipse, GEF and AutAT have to be installed before creating a test. The differences between AutAT and other acceptance testing tools for web applications are many. One of them is the graphical user interface that is easy to work with. AutAT also introduces the use of aspects to reduce the amount of redundancy in the tests.

An installation guide from [28] is used to install AutAT. See Figure 6.2 to get an idea of AutAT user interface. To run a test, the test has to be created in AutAT. The test in Figure 6.2 will use the login page and log in with different users. After creating the test, it's possible to convert it to Watir, Canoo WebTest, JWebUnit, html Skelton or FIT test. I've chosen to convert the my tests to Watir.

Not all tests can be automate by AutAt. There are some form elements like for instance radio button that are not implemented. Furthermore, the AutAT tool doesn't support testing of dynamic Web pages. Thus, a dynamic Web page or a Web page that includes radio button can not be tested by AutAT.

6.4.2 Watir

Watir[5] stands for "Web Application Testing in Ruby". Watir is a free, open source functional testing tool for automating browser-based tests of web applications.

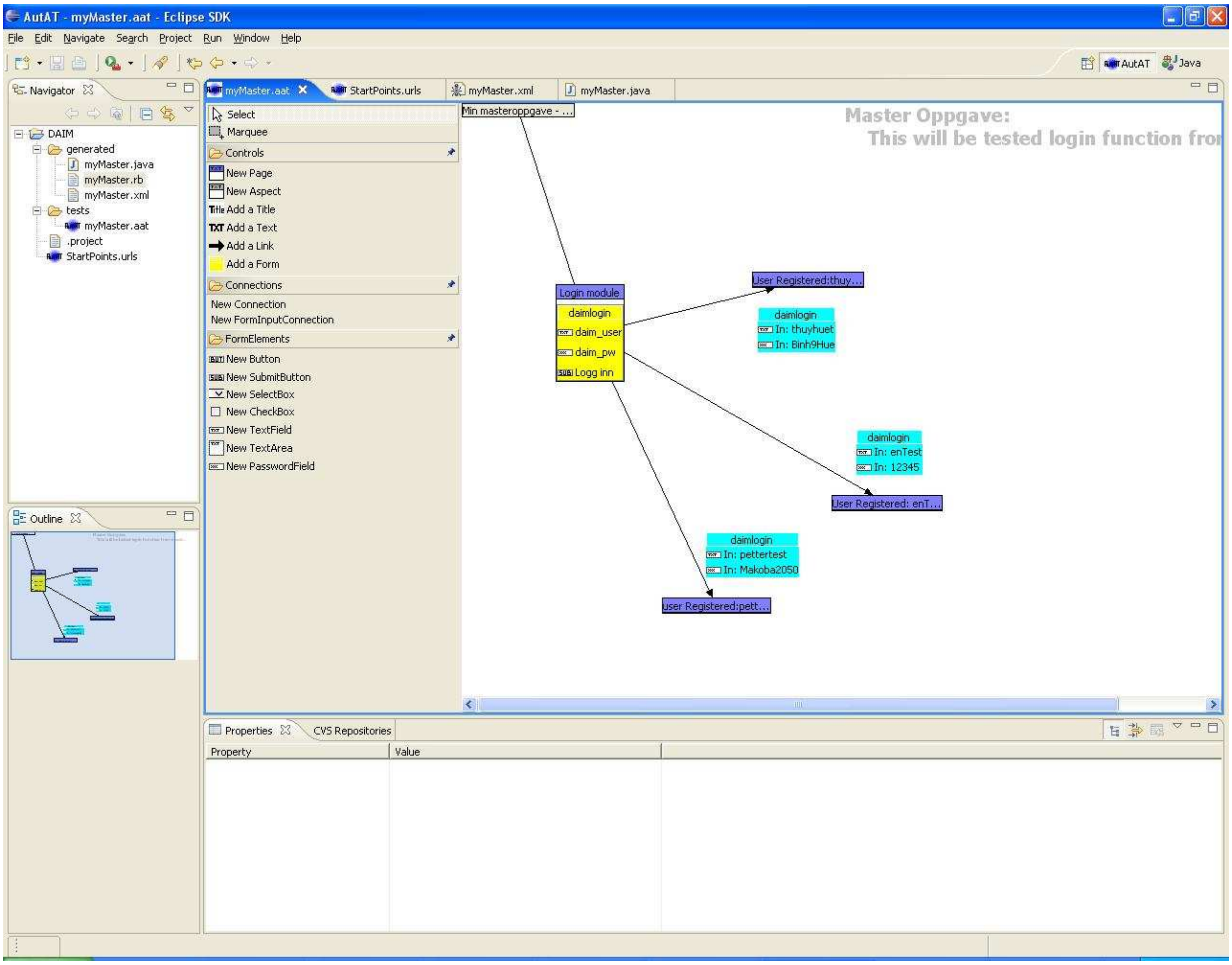
This type of testing drives the Internet Explorer in the same way as people do, by clicking on links, fill in forms, press buttons, check result whether expected text appears on the page and so on. Furthermore, Ruby gives you the power to connect to databases, read data files, export XML and structure the code into reusable libraries.

Watir has to be installed in order to convert test in AutAT to Watir. After creating the test in AutAT, the test will be converted to Watir and run. I've used the user's guide from [12] to help me understanding, installing and running tests in Watir.

[4] http://boss.bekk.no

[5] http://wtr.rubyforge.org

Figure 6.2: An example of a test in AutAT



6.5 Summary

It's important to test new software to ensure high quality. Software testing will occur at several levels of detail within the application. A test can be designed either with or without detailed knowledge of the internal logic of the software, and either before or after the application is implemented.

Testing of Web applications introduces new chanllenges related to the nature of the applications. Some tests for Web applications can hardly be automated, but many can. The automated tests can be executed more often, at a much lower cost. This will probably increase the quality of the applications.

AutAT will automate most acceptance test, but not all. Testing in AutAT will be performed as black box testing. While creating tests in AutAT, several invalid inputs are exercised and the outputs are compared to validate the robustness of the DAIM system.

Chapter 7

Goal/Question/Metric method

The Goal/Question/Metric method can be used to build research questions in a structured and organized matter. Here we follow the GQM template for definition [2]. Furthermore, the GQM method is used in the testing process. The method provides a systematic approach to the process, which will be helpful for software testing in general.

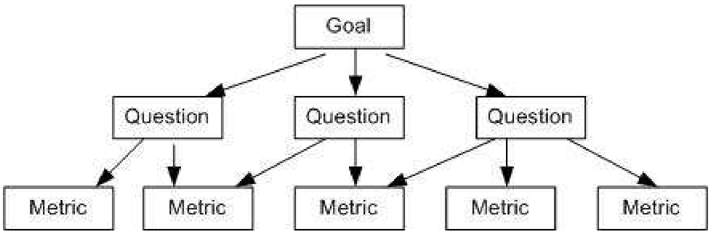
The focus of this section is to define the goal, questions and metrics for the testing process of this project. Since input robustness is the focus of the project, metrics can be related to the system's users or its testers . The test will be automate as far as possible from the point of view of the testers, with the purpose of assessing the robustness of DAIM.

Sometimes, it's difficult or impossible to automate tests. The reason can be that the metric require a subjective answer or chosen tools don't support i.e. the dynamic Web pages, the form in the tool is not included the radio button. The GQM method consists four phases as follows:

- Planning: In this phase the project is selected , defined, characterised, and planned. The result is a project plan
- Definition: During this phase, the goal, questions, metrics and hypotheses are defined and documented
- Data collection: The actual data collection is performed. This results in a set of data
- Interpretation: The collected data is processed with respect to the defined metrics into measurement results. This will provide answers to the questions, which ultimately leads to an evaluation of the goal attainment.

The basic idea is to derive software measures from measurement questions and goals. The method is built up in a hierarchical way with three levels as shown in Figure 7.1. As the Figure 7.1 shows, the goal is located at the highest level. This is the purpose of the measurements. At the middle level, the questions are related to the measurements. These questions characterize the way the achievement of a specific goal is going to be

Figure 7.1: An example of GQM-tree



performed. At the lowest level is the metrics, associated with the questions are located. These metrics define a measureable way to obtain answers to the questions.

7.1 Goal

This section describes the goal of applying the method. There might be several goals for one project. The method uses a standard template to defining its goal according to Wohlin [22]:

Analyze <*Object(s) of study*>
for the purpose of <*Purpose*>
with respect to their <*Quality focus*>
from the point of view of the <*Perspective*>
in the context of <*Context*>

In this project, the goal of testing is defined as follows:

Analyze **DAIM**
for the purpose of **estimating the robustness**
with respect to their **Input**
from the point of view of the **Tester**
in the context of **DAIM**

7.2 Questions

The questions must contribute to the goal and should preferably not be answered with a simple yes or no. This section describes questions that support our goal. The questions are associated with one or several metrics.

To reach our goal, the question must contribute to the goal when answered. I believe that the following questions capture most of the important issues involved in reaching the goal.

Question 1: Invalid input tolerance
How is the invalid input tolerance of the DAIM system?

Input robustness is the major focusing of this project and has to be tested. It's measured by the metrics M1, M2 and M3. These metrics will use AutAT to test and give answer to the questions. In some cases, the tests has to be performed manually when the functionality in the AutAT system does not support the type of testing needed. Answers of the question will tell us how robust the DAIM system is and give an estimating of the input robustness.

Question 2: DAIM interface
Have the DAIM interface contributed to increasing the input robustness of DAIM?

This question covers many aspects concerning the front-end the users will need to work with, and will be answered with the metrics M4 and M5. Input can be validated on the client by using HTML input boxes to restrict the size or contents of inputs. This forms a syntactic restrictions and the interface contributes to increase the input robustness by syntactic restictions. In order to answer this question, a tester has to check the GUI of the DAIM manually, without using AutAT.

7.3 Metrics

The metrics answer the questions by measuring aspects. The metrics can be answered either objectively or subjectively, and there can be several metrics that contributes to one questions. The GQM-metrics are used to give answers to the questions by measuring areas of interest. The metrics are listed in Tables 7.1 to 7.5.

Table 7.1: Metric 1 - Error tolerance

Name	M1: Error tolerance
Definition	The system should be able to handle invalid user input by correct results or prompting the user
When to measure	During the test session
Procedure for measuring	This will be qualitative test with different test data for the different kinds of data fields. Count the number of tests the system passed by invalid input (TP) compared to the total number of tests (T). X = TP/T, the closer X is to 1 the more robust. The tests will be created in AutAT and converted to Watir to run.

Continued on next page

Table 7.1 – continued from previous page

Name	M1: Error tolerance
Expected Value	A subjective, qualitative judgement will have to be made as how robust the system is, or how error tolerant the system is. Based on the works described in chapter 9, the expected input robustness value X will be approximately 0.59.

Table 7.2: Metric 2 - Error meassages

Name	M2: Error messages
Definition	The error messages of the system should be meaningful, non-abusive and devoid of computer gibberish
When to measure	During test session
Procedure for measuring	Count the number of error messages that are possible to understand without any special computer knowledge (A), compared to the total number of error messages (B). X = A/B, the closer X is to 1 more meaningful are the error message.
Expected Value	Error messages have to be intelligible. The system has high exception handling capability and we expect X to be approximately 0.70.

Table 7.3: Metric 3 - Feedback

Name	M3: Feedback
Definition	When a user performs an action, the system should respond in an intuitive way. For example, if a user push a button, a clear confirmation message or action should be presented. This is important in order to prevent that the user misunderstanding and repeating an action, which could also lead to database errors.
When to measure	During the test sesion

Continued on next page

Table 7.3 – continued from previous page

Name	M3: Feedback
Procedure for measuring	Count the number of confirm/register functions that produce a clear confirmation message or action (A) and compare to the total number of confirm functions (B). X = A/B, the closer X is to 1 the better feedback of the system to the user
Expected Value	This has not been a priority and the expected value X is approximately 0.70.

Table 7.4: Metric 4 - DAIM precision

Name	M4: DAIM precision
Definition	The user should be able to precisely determine how to enter data into the system
When to measure	During the test session
Procedure for measuring	Measure if the syntax of the various input data is clearly defined by counting the number of special syntax input fields that have an explanation (A) compared to the number of fields required special syntax (B). X = A/B, the closer X is to 1 the more precise
Expected Value	This has not been a priority and the expected value X is approximately 0.50.

Table 7.5: Metric 5: Syntactic input validationn

Name	M5: Syntactic input validation
Definition	HTML input boxes can also be used to impose several types of syntatic restrictions, and to avoid invalid user input
When to measure	During test session

Continued on next page

Table 7.5 – continued from previous page

Name	M5: Syntactic input validation
Procedure for measuring	Count the number of fields that user can fill in for an input by choosing a select box, check boxes, radio boxes or text field with included max-length (A), and compare to the total number of fields that it's possible to use select box, check box, included max-length in a text field etc. to avoid invalid user input (B). X = A/B, the closer X to 1, the more syntactic input validation is used to avoid invalid user input and contribute the input robustness
Expected Value	This has not been a priority and the expected value X is approximately 0.50.

The relationships between the metrics and the questions are present in Table 7.6. The results from the data collection and the interpretation will be presented i Chapter 11.

Table 7.6: The relationships between the metrics and the questions

Metrics	Questions	
	Q1: Input tolerance	Q2: DAIM interface
M1 - Error tolerance	X	
M2 - Error messages	X	
M3 - Feedback	X	
M4 - DAIM precision		X
M5 - Syntactic invalid input		X

Part III

Robustness Assessment of DAIM

Chapter 8

DAIM's requirements

This chapter will identify the DAIMs requirements. First, nonfunctional requirements will be presented. The database requirements and the robustness requirements will be included in the nonfunctional requirements because they are relevant topic to this project. After the nonfunctional requirement are treated, the functional requirements will be presented by a set of use case diagrams together with their interpretations.

In order to study DAIM, a use case diagram from [6] is used to describe DAIM's functional requirement in a high level, where each use case can consist of several user stories. DAIM aims to support students and administrator's work to handle master thesis at NTNU. Accessibility and preservation of master thesis is one of the major purposes of DAIM.

8.1 Nonfunctional requirements

This section presents the nonfunctional requirements of DAIM. Only topics that are relevant to the project are captured in this section.

8.1.1 Database

The system's database consists of two servers; "Web server" and "IDI database" server. Sometimes, the system also have to connect to the "NTNU innsida" database, "Tapir database server" and "Tapir Web server" which the DAIM system has no controll over. Thus, the database requirements are just considered for the "Web server" and "IDI database".

The database must be consistent at all times. It will have to work according to atomicity, consistency, isolation and durability (ACID) properties [3]. In addition, administrators should make sure that there is always a backup of the database. The database has following requirements:

- The database should be available 95% of the time. This requirement concerns availability. Keeping the database available at all times is neither realistic nor neccessary. It will have to be unavailable during for instance during repairs and update. Furthermore, this application is not a critical one.

- The database should be able to cope with simultaneous access. The requirement states that simultaneous access not should cause trouble for the system.

8.1.2 Robustness requirements of DAIM

Different robustness definitions should demand different requirements for robustness. [24] has pointed out that reliability techniques can be applied to improve the robustness of the system. Reliability is built up from completeness and correctness.

Robustness requirements for a Web-based system will vary. In this project, the robustness requirement is that DAIM should always respond to the user's action either by correct results or prompting the user to try again due to user mistake or internal component failure. This requirement consists of the following subrequirements:

- DAIM should have ability to detect invalid input.
- DAIM should have ability to recover from invalid input.
- DAIM should have ability to give intelligible feedback
- The system should have the ability to survive incorrect input from user or from another system.

The acceptable robustness value for DAIM system is 0.9 or higher.

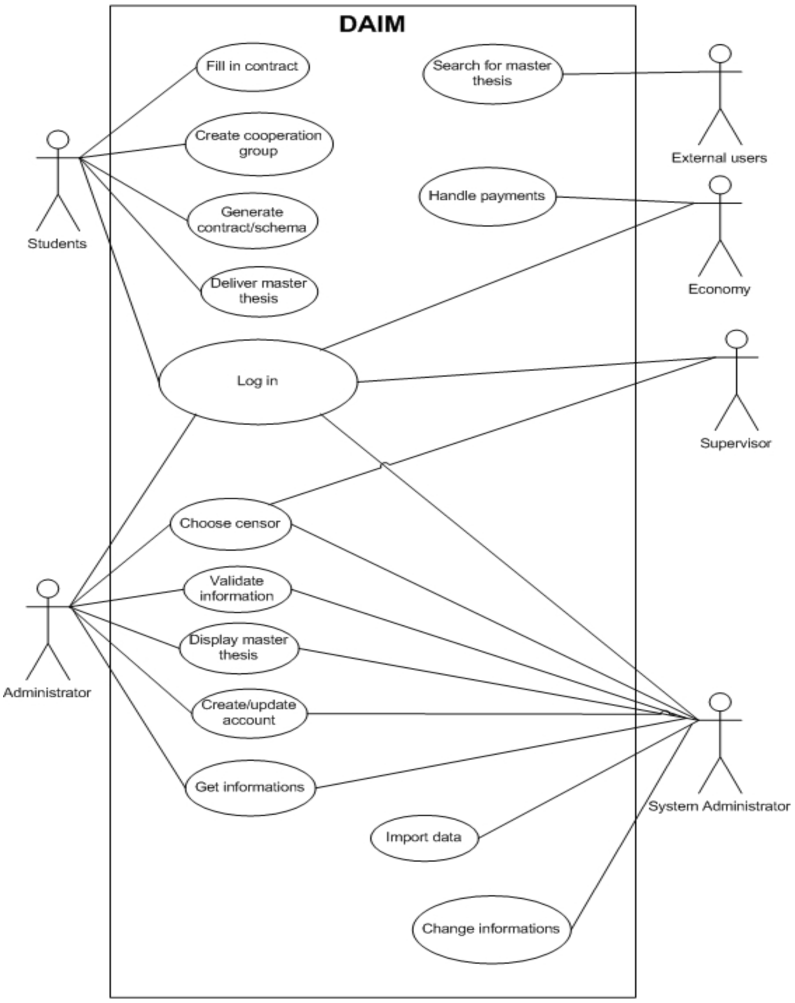
8.2 Functional Requirements

This section presents the functional requirements of the system. The major purpose of this section is to get an idea of which functions/tasks the system can perform. DAIM is a role-based system and use cases are categorized by various roles. Figure 8.1 illustrates the major tasks that the system's roles can perform. The use cases have the following interpretations:

Student

- UC01: Log in. Every internal actor, such as student, administrator, system administrator, supervisor and economy have to be log in before they can do anything else. External users who just want to search for master thesis at NTNU don't have to log in.
- UC02: Fill in contract. The required informations about a master thesis have to be filled in before deadline for the selection of the project.
- UC03: Create cooperation group. Two or maybe three students can cooperate in writing a master thesis. The students cooperating have to be added when the contract for the master is filled in.

Figure 8.1: Use cases diagram for DAIM



- UC04: Generate contract/schema. Students can generate contract/schema if they have filled in all required informations and the administrator has approved the contract. Again, this use case contains three user stories as follows:
  - UC04-01: Generate contract for a master of science.
  - UC04-02: Generate contract for a cooperation group. This service is available when all the students in a group have approved the cooperation.
  - UC04-03: Generate delivery schema. This function is available when the delivery process is finished.
- UC05: Deliver master thesis. Students can upload several files for their master thesis. Two days after delivery, the paper versions of master thesis is delivered at institute's expedition. Extra copies can also be ordered with the delivery process. This use case contains the following user stories:
  - UC05-01: Upload a picture of the front-page for the project.
  - UC05-02: Upload an attachment as a zip file.

  - UC05-03: Upload a report in PDF format and fill in extra informations that related to paper version, such as total page numbers, page numbers that will be printed in colour, comments that will send to Tapir and/or institute. In addition, decision of the publishing of the master thesis is allowed or not.
  - UC05-04: Order extra copies of student's master thesis.

External users

- UC06: Search for master thesis. The search module is available for everybody and doesn't require a logon. Master thesis will be shown if:
  - Master thesis has examination results.
  - Students have accepted publishing.
  - The master thesis is not restricted.

Economy

- UC07: Handle payments. The economy role handles every payments for master thesis. When the actor logs in, the default page will show the following information:
  - UC07-01: A list of master thesis which still miss the payments for censoring
  - UC07-02: Change status of master thesis which have paid for censoring
  - UC07-03: A list of master thesis with paid censoring
  - UC07-04: A list of the institute's ordinary censors.

Administrator and System administrator user
The use cases for both administrator and system administrator are as follows:

- UC08: Display master thesis. The user can get an overview of delivered master thesis by choosing a combination of several parameters or by searching. This use case consists of following stories:
  - UC08-01: The master thesis can be chosen by several parameters, such as status of the master thesis, or type of study. The student name, supervisor and the title will be shown.
  - UC08-02: The user can also sort the master thesis found by student, supervisor, and title
  - UC08-03: It is possible to search for the master thesis by student name, supervisor, censor, title or the date that the master thesis was delivered, censored date, etc.
- UC09: Choose censor. Administrator, system administrator, and supervisor can choose a censor for one specific master thesis. The censor can be found in a list. The supervisors can only choose a censor for the master thesis that they has been a guidance.

- UC10: Validate information. A master thesis has several status, e.g. "Registered, but master contract is not ready", "Writing the master thesis", "Finished, but the character is not desired", "Finished, the character is desired" and so on. Generally, there are different information that can be validated, dependent on the status of the master thesis. When a student has filled in the required informations in the contract for his master's degree, the administrator/system administrator have to check if the informations are correctly filled in, and then approve the master contract. If the informations are not correctly filled in, the administrator tells the student to change information. The delivery process is available for students when the master contract is approved by an administrator or system administrator. Both actors can change the deadline of the master thesis if required. The users can also fill in more informations that related to the master thesis, such as comments, the date sent to a censor and so on.
- UC11: Create/update account. This use case contains the following user stories:
  - UC11-01: Create a student account if the student has not yet registred.
  - UC11-02: Create/Update a supervisor's account. Informations of registered supervisor can be updated. A supervisor account can be created if it doesn't registered.
  - UC11-03: Create/Update a censor's account. A censor account is created if it doesn't exist. Informations of registered censors can also be updated.
  - UC11-04: Find a censor/supervisor by searching
- UC12: Get information. Both the administrator and the system administrator can get information in order to get an overview of all master thesis. This use case includes following user stories:
  - UC12-01: Get an overview of important deadlines for censuring.
  - UC12-02: Get an overview of information about restricted master thesis.
  - UC12-03: Get an overview of information about secondary-supervisors and external supervisors.
  - UC12-04: Get an overview of information about delivered master thesis.

System Administrator
The system administrator can perform the same use cases as the administrator actor. In addition, the system administrator can carry out some extended functions as follows:

- UC13: Import data. System administrator can import data from local database for students and censors. The data must be saved as CSV format.
- UC14: Change information. UC14 consists of two user stories:
  - UC14-01 Change information for an institute
  - UC14-02 Add an administrator user.

Supervisor

- Choose censor: See use case "Choose censor" from administrator actor.

Chapter 9

DAIM and Robustness Assessment

This chapter describes the robustness failure modes and the robustness assessment of DAIM. Let's consider the DAIM system to illustrate the use of the frameworks presented in [24] and [26].

Before moving on, we have to make clear that only the robustness failure modes will be identified and described in this chapter. Other failure modes which related to other quality attributes, such as availability or reliability will not be considered.

9.1 Jacobson's analysis method and FMEA

This section described how the frameworks from [24] can be used to assess DAIM's current robustness. This framework integrates Jacobson's analysis, a systematic way to decompose the system, and a simplified version of FMEA.

9.1.1 UC01: Log in

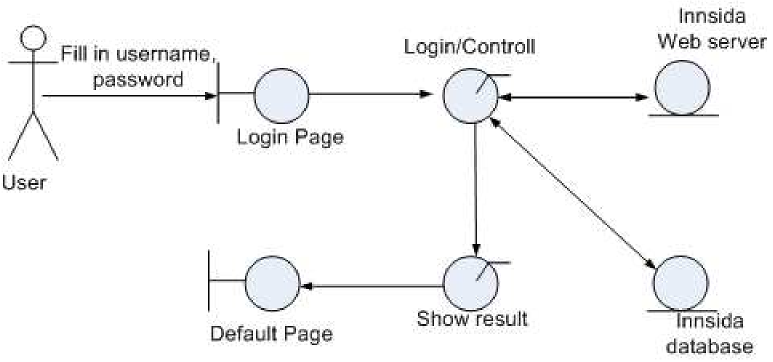
Each internal actor has to be logged on before performing any other tasks. There is different logon page for students and other administrator roles, such as supervisor, economy, etc.. Hence, students and the administrator roles have to use different page to log on, and a different default page is displayed after login.

The system uses the login module from "Innsida" to control and check the login. "Innsida" is a login module that checks that the username exists in the NTNU database when logging on to the NTNU internal page for internal users, such as students. The system has no control of the "Innsida" database.

Figure 9.1 shows the result of using Jacobson's analysis diagram for the use case "Log in". The user types the username and password in the "Login Page". The "Login/control" checks the username and password by interacting with the "Innsida database" and displays the result on the "Default Page" by the module "Show result".

The application logic of this use case is captured by "Login/Control" and "Show result"

Figure 9.1: Analysis diagram for Log In use case



Control Objects. The FMEA worksheet for these Control Objects are shown in Table 9.1.

Table 9.1: FMEA for Log in

System: DAIM Use case: Log in			Performed by: Hue Pham Date: 08.03.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Login/ Control	No response is produced at all	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. "Login/-Control" detects the failure of the object and interacts with "Show Result" to prompt the user appropriately
		Database contains incorrect/ damaged data	User cannot log in with correct username and password	Users suspect the quality of the system	Manage data in "IDI database" and ensure its correctness. Interact with "Show Result" to give feedback to the user

Continued on next page

Table 9.1 – continued from previous page

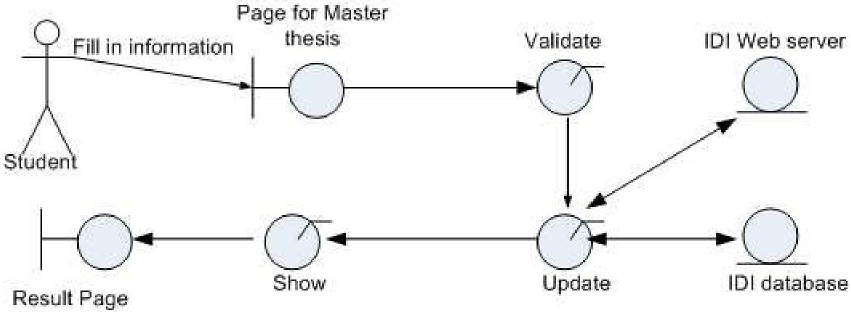
Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
Show result	No response is produced at all	Error output of "Login/-Control"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Login/Control" and prevent serious errors from entering the object. Prompt the user appropriately

9.1.2 UC02: Fill in contract

Students have to fill in required information about themselves and their master thesis. It's important here that system should give intelligible feedback for invalid input, such as date format, a incomplete email adress, etc.

Jacobson's analysis diagram for the use case "Fill in contract" is illustrated in Figure 9.2. The student fills in required information. First, "Validate" module validates the input at the client side. Then, the input will be validated at "IDI Web server" before updating to "IDI database". The result is displayed at "Result Page" by "Show".

Figure 9.2: Analysis diagram for Fill in contract



The use case "Fill in contract" in Figure 9.2 is structured into three Control Objects that will perform this service. Table 9.2 shows the failure modes when students fill in contract.

Table 9.2: FMEA for Fill in contract

System: DAIM Use case: Fill in contract			Performed by: Hue Pham Date: 09.03.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Validate	Invalid inputs are not recognized and will result in abnormal behavior or invalid data will be saved	Students typed in invalid input	Fail to respond to user's interaction or system saves invalid data	Prevent further use of the system or database will contain incorrect data	Control and validate user input and prevent serious error from entering the object. "Page for Master thesis" detects the failure and interacts with "Show" to prompt the user appropriately
Update	Abnormal behavior	Error output of "Validate"	Fail to respond to user's interaction	Prevent further use of the system.	Control input from "Validate" and prevent serious error from entering the object. "Update" detects the failure and interacts with "Show" to prompt the user appropriately
	1) Invalid data will be saved 2)Information found is incorrect	Error output of "Validate" and/or "IDI database" contains incorrect data	Incorrect data is presented to the user	User suspects the quality of the system	Control input from "Validate" and prevent serious error from entering the object. Manage "IDI database" and ensure its correctness

Continued on next page

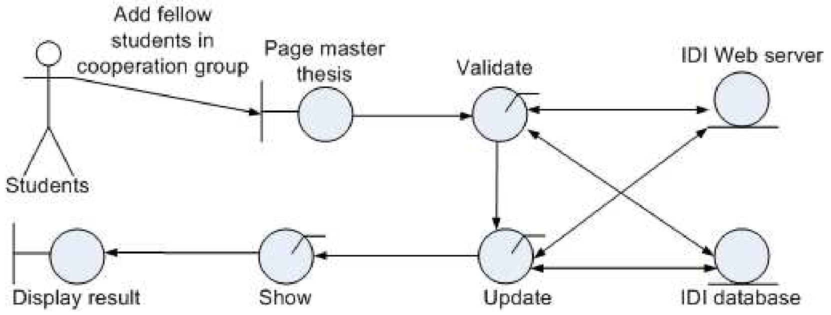
Table 9.2 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
Show	No response is produced at all	Error output of "Update"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Update" and prevent serious errors from entering the object. "Update" detects the failure of the object and interacts with "Show" to prompt the user appropriately
	Incorrect data is displayed	Error output of "Update"	Incorrect data is presented to user	Users suspect the quality of the system	Control input from "Update" and prevent serious error from entering the object.

9.1.3 UC03: Create cooperation group

Several students can cooperate in writing master thesis. The usernames of the fellow students in the cooperation group have to be filled in. Every student in the group must have approved the cooperation before generating the cooperation contract. The students can not cooperate if they have different types of studies. Figure 9.3 shows the result of using Jacobson's analysis diagram for the use case "Create cooperation group".

Figure 9.3: Analysis diagram for Create cooperation group



The student types usernames of the fellow students in the cooperation group. The system have to validate each username to ensure it is existed or valid cooperation. If it's a valid cooperation, the result will be updated in the "IDI database" and displayed to the user. System prints a message to the user if it's invalid cooperation, or student's name is not existed.

The structure of this function is split into "Validate", "Update" and "Show" Control Objects. The failure modes of this use case are captured in Table 9.3.

Table 9.3: FMEA for Create cooperation group

System: DAIM Use case: Create Cooperation group			Performed by: Hue Pham Date: 09.03.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Validate	No response is produced at all	Students typed in invalid input	Fail to respond to user's interaction	Prevent further use of the system	Control and validate user input and prevent serious error from entering the object. "Page for Master thesis" detects the failure and interacts with "Show" to prompt the user appropriately
	Information found is incorrect	IDI database contains incorrect data	Incorrect data is presented to the user	User suspects the quality of the system	Control user input and prevent serious error from entering the object. Manage IDI database and ensure its correctness
Update	No response is produced at all	Error output of "Validate"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Validate" and prevent serious error from entering the object. "Update" detects the failure and interacts with "Show" to prompt the user appropriately
					Continued on next page

Table 9.3 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
	Information found is incorrect	Error output of "Validate" and/or IDI database contains incorrect data	Incorrect data is presented to the user	User suspects the quality of the system	Control input from "Validate" and prevent serious error from entering the object. Manage IDI database and ensure its correctness
Show	No response is produced at all	Error output of "Update"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Update" and prevent serious errors from entering the object. "Update" detects the failure of the object and interacts with "Show" to prompt the user appropriately
	Incorrect data is displayed	Error output of Update	Incorrect data is presented to user	Users suspect the quality of the system	Control input from Update and prevent serious error from entering the object

9.1.4 UC04: Generate contract/schema

The contract can be generated when the student has filled in the required information. The delivery schema can be generated after delivering of the master thesis. Figure 9.4 illustrates the use case "Generate contract/schema". "Page for generating" is shown when required information is filled in. The system generates the contract/schema by interacting with "IDI database" to get information. The information is sent to "Generate" Control Object for generating the contract/schema. The result and the generated contract/schema are shown by the "Show" Control Object and displayed in the "Result Page". The robustness failure modes for these Control Objects are recorded in Table 9.4.

Figure 9.4: Analysis diagram for Generate contract/schema

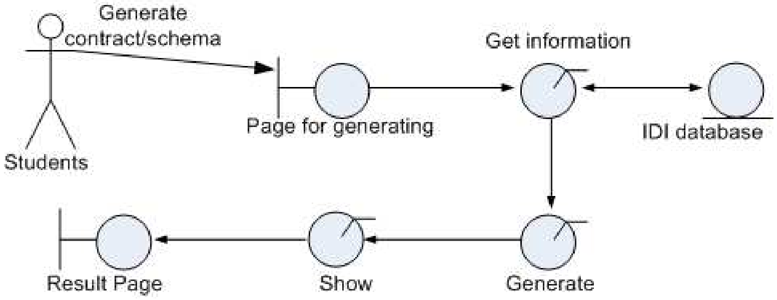


Table 9.4: FMEA for Generate contract/schema

System: DAIM Use case: Generate contract/schema			Performed by: Hue Pham Date: 09.03.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Get information	No response is produced at all	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from enering the object. "Page for generating" detects the failure of the object and interacts with "Show" to prompt the user
	Information found is incorrect	Database contains incorrect data or error user input	Generated contract/schema shows incorrect data	User suspects the quality of the system	Manage data in "IDI database" and ensure its correctness. Control user input and prevent serious errors from entering the object

Continued on next page

Table 9.4 – continued from previous page

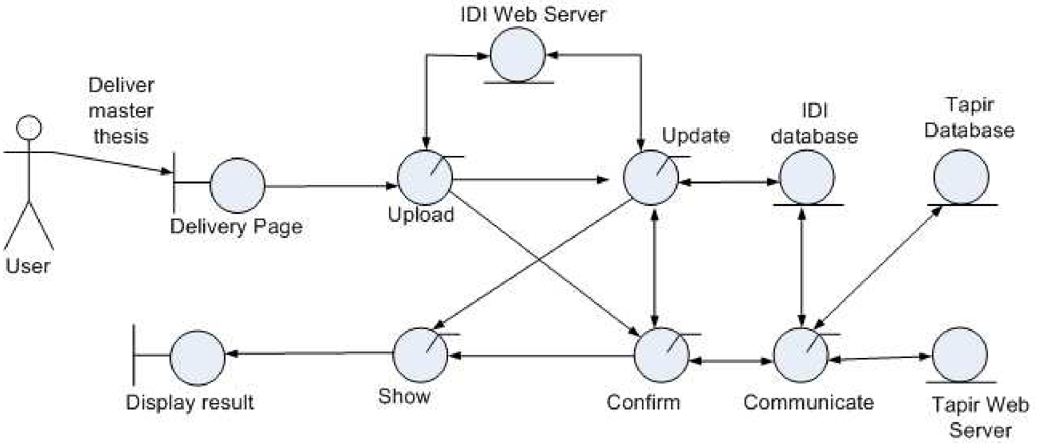
Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
Generate	No response is produced at all	Error output from "Get information"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Page for generating" and prevent serious errors from entering the object. Detect the failure of the object and interact with "Show" to prompt the user appropriately
Show	No response is produced at all	Error output from "Generate"	Contract /schema couldn't be generated	Prevent further use of the system	Control input from "Generate" and prevent serious errors from entering the object. "Generate" detects the failure of the object and interacts with "Show" to prompt the user
	Incorrect information is displayed	Error output of "Generate"	Incorrect information is presented to the user	The user suspect the quality of the system	Control input from "Generate" and prevent serious erors from entering the object

9.1.5 UC05: Deliver master thesis

Students can upload several files when the delivery their master thesis. A front-page picture and attachment are uploaded to "IDI Web server" and the "IDI database" is thus updated by using "Upload", "Update" and "Show".

The master thesis in PDF format will be uploaded to "IDI Web server", and update to "IDI database" when delivery process completes by "Confirm". "Confirm" and "Communicate" communicate with "Tapir Database" to order paper version. If the connection with "Tapir server" is approved, the PDF file will be saved in "IDI database" and "Tapir database". Other information related to the copying and publishing of the master thesis will be send to the "IDI database" and "Tapir database". The delivery process is dependent of the availability of "IDI Web server", "IDI database", "Tapir Web server", "Tapir database". But if the "Tapir server" is down, information of the delivery process will be saved at "IDI database" and will be sent to the "Tapir database" later by the administrator role.

Figure 9.5: Analysis diagram for Delivery master thesis



The application logic of this use case is captured bye "Upload files", "Update", "Confirm", "Communicate" and "Show". The FMEA worksheet for these Control Objects are shown in Table 9.5

Table 9.5: FMEA for Delivery master thesis

System: DAIM Use case: Delivery master thesis			Performed by: Hue Pham Date: 10.03.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Upload files	1)Invalid inputs are not validated and result in abnormal behavior	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. "Delivery Page" detects the failure of the object and interact with "Show" to prompt the user appropriately

Continued on next page

Table 9.5 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
Update	Abnormal behavior	Error output of "Upload", "Confirm"	Fail to respond to users interaction	Prevent further use of the system	Check the input from "Upload", "Confirm" and prevent serious errors from entering the object. "Delivery Page" detects the failure of the object and interacts with "Show" to prompt the user appropriately
	Invalid inputs are saved	Error output of "Upload", "Confirm"	Database contains incorrect data	Users suspect the quality of the system	As described above, and check the "IDI database" to ensure its correctness
Confirm	No response is produced at all	Error output of "Upload", "Update", "Communicate"	Fail to respond to user's interaction	Prevent further use of the system	Control the input of "Upload files", "Update", "Communicate" and prevent serious errors from entering the object. Interact with "Show" to prompt the user
Communicate	No response is produced at all	Error output of "Confirm"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Confirm" and prevent serious errors from entering the object. Interact with "Show" to display feedback to user immediately

Continued on next page

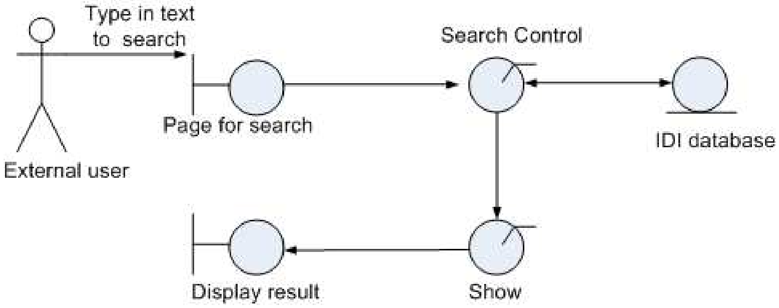
Table 9.5 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
Show	No response is produced at all or information shown are incorrect	Error output of "Confirm" and "Update"	Fail to respond to user's interaction	Prevent further use of the system and users suspect the quality of the system	Control input from "Confirm", "Update" and prevent serious errors from entering the object. Give intelligible feedback to user

9.1.6 UC06: Search for master thesis

External users can use the search module without a logon. Users can use either basic or advanced search. The basic search will search for the text in title, supervisor, author and abstract. The advanced search will perform the search by choosing several parameters. Figure 9.6 illustrates the use case "Search master thesis". The "Search Control" and "Show" are Control Objects for the use case. The "Search Control" connects to the "IDI database" and fetches the relevant information. The "Display result" is displayed by "Show".

Figure 9.6: Analysis diagram for Search master thesis



The FMEA worksheet for the use case is described in table 9.6

Table 9.6: FMEA for Search master thesis

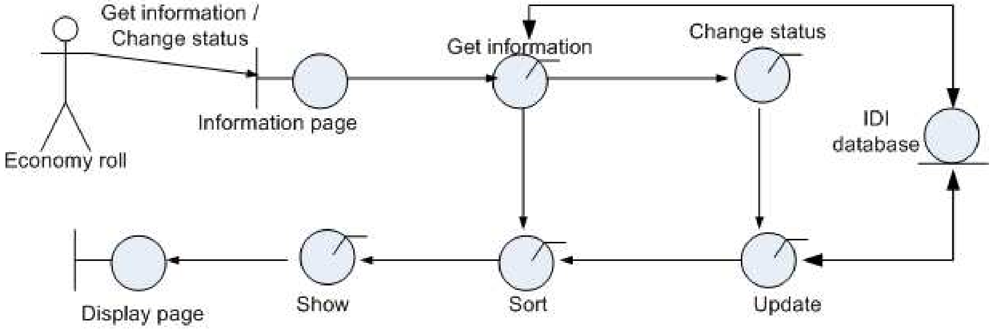
System: DAIM Use case: Search master thesis			Performed by: Hue Pham Date: 15.03.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Search Control	Invalid inputs are not recognized	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. "Search Page" detects the failure of the object and interacts with "Show" to prompt the user appropriately
	Information found is incorrect	Error user input and/or "IDI database" contains incorrect data	Incorrect data is presented to the user	Users suspect the quality of the system	Control user input and prevent serious errors from entering the object. Manage data in "IDI database" and ensure its correctness
Show	No response is produced at all	Error output of "Search Control"	Fail to respond to user's interaction	Prevents further use of the system	Control input from "Search Control" and prevent serious errors from entering the object. "Search Control" detects the failure and interacts with "Show" to prompt the user appropriately
	Incorrect information is displayed	Error output of "Search Control"	Incorrect data is presented to the user	User suspects the quality of the system	Control input from "Search Control" and prevent serious errors from entering the object

9.1.7 UC07: Handle payments

When the economy role logs in, the default page will show information of payments. Displayed informations can be sorted by several parameters, such as students, supervisors or censors.

The Jacobson's analysis model is shown in Figure 9.7. The information will be chosen, sorted and shown by "Get information", "Sort" and "Show". Status of a censoring can be changed to "paid", or "not paid" by the economy role. "Change status" and "Update" will handle changes and updating in the database.

Figure 9.7: Analysis diagram for Handle payments



The failure modes of this use case are charted in the Table 9.7.

Table 9.7: FMEA for Handle payment

System: DAIM Use case: Handle payment			Performed by: Hue Pham Date: 21.03.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Get information	No response is produced at all	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. "Information page" detects the failure of the object and interact with "Show" to prompt the user appropiately

Continued on next page

Table 9.7 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
	Information found is incorrect	Error user input and/or "IDI database" contains incorrect data	Incorrect data is presented to the user	User suspect the quality of the system	Control user input and prevent serious errors from entering the object. Manage data in "IDI database" and ensure its correctness
Change status	No response is produced at all	Error output of "Get information"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Get information" and prevent serious errors from entering the object. "Get information" detects the failure of the object and interacts with "Show" to prompt the user
Update	No response is produced at all	Error output of "Change status"	Fail to respond to user's interaction	User suspects the quality of the system	Check the input from "Change status" and prevent serious errors from entering the object. "Change status" detects the failure of the object and interacts with "Show" to prompt the user appropriately
					Continued on next page

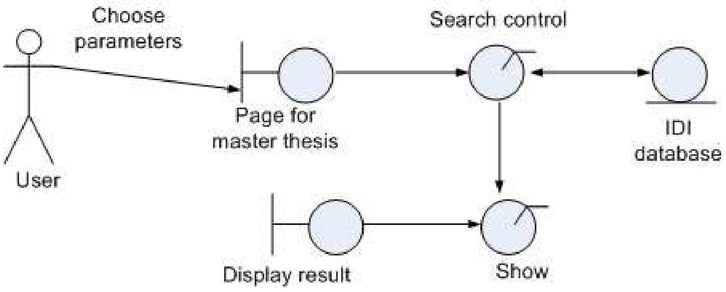
Table 9.7 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
	Information found is incorrect	Error output of "Change status" and/or "IDI database" contains incorrect data	Incorrect data is presented to the user	User suspect the quality of the system	Control input from "Change status" and prevent serious errors from entering the object. Manage data in "IDI database" and ensure its correctness
Sort	No response is produced at all	Error output of "Update", "Get information"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Get information", "Update" and prevent serious errors from entering the object. Detect the failure of the object and interact with "Show" to prompt the user
Show	No response is produced at all	Error output of "Sort"	Fail to respond to user's interaction	Prevent further use of the system	Control input of the "Sort" and prevent serious errors from entering the object. Give feedback to user
	Incorrect data is displayed	Error output of "Sort"	Incorrect data is presented to user	User suspects the quality of the system	Control input from "Sort" and prevent serious errors from entering the object

9.1.8 UC08: Display master thesis

Figure 9.8 illustrates the use case "Display master thesis". The application logic of this use case is similar to the use case "UC06: Search for master thesis". A user searchs for a master thesis by choosing several parameters, or by typing a text to search. UC06 and UC08 differ by the login module. UC08 is just available for administrator and system administrator roles. All master thesis will be available for searching from UC08.

Figure 9.8: Analysis diagram for Display master thesis

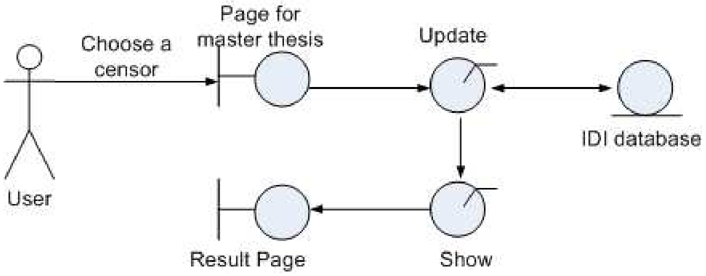


When a user has chosen or filled in desired parameters, "Search Control" will interact with the "IDI database" and find the relevant information. "Show" handles data and displays them to the user. The robustness failure modes of this use case are similar to the failure modes captured in Table 9.6.

9.1.9 UC09: Choose a Censor

When a student has performed the delivery process, the system administrator, administrator or supervisor can choose a censor for the master thesis. A list of censors will be shown to the user who can then choose a censor from the list. If the censor doesn't exist, the user has to add the censor in database. This function is, however, not included in this use case. Figure 9.9 illustrates the Jacobson's analysis model of the use case.

Figure 9.9: Analysis diagram for Choose censor



Changed information will be updated to the database by "Update". "Show" handles the result and display it in "Result Page". Table 9.8 shows the robustness failure modes when a user chooses a censor.

Table 9.8: FMEA for Choose censor

System: DAIM Use case: Choose censor			Performed by: Hue Pham Date: 07.04.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Update	No response is produced at all	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. Give feedback to the user
	Information found is incorrect	Database contains incorrect data	Incorrect data is presented to the user	User suspects the quality of the system	Manage data in "IDI database" and ensure its correctness
Show	No response is produced at all	Error output of "Update"	Fail to respond to user's interaction	Prevent further use of the system	Control input rom "Update" and prevent serious errors from entering the object. "Update" detects the failure of the object and interacts with "Show" to prompt the user appropriately
	Incorrect information is displayed	Error output of "Update"	Incorrect information is displayed to the user	User suspects the quality of the system	Manage data in "IDI database" and ensure its correctness

9.1.10 UC10: Validate information

When a student has filled in the required information for a master contract, the administrator role can validate information. First, the administrator has to choose a student. The details of the master thesis of the student will be shown. The user validates the informations and tells the students if the information must be changed. The administrator can also change deadline or write a comment related to the master thesis. Available information that can be validated by the user are depedent on the status of the master thesis. Figure 9.10 shows the result of using Jacobson's analysis diagram for the use case "Validate information".

Figure 9.10: Analysis diagram for Validate information

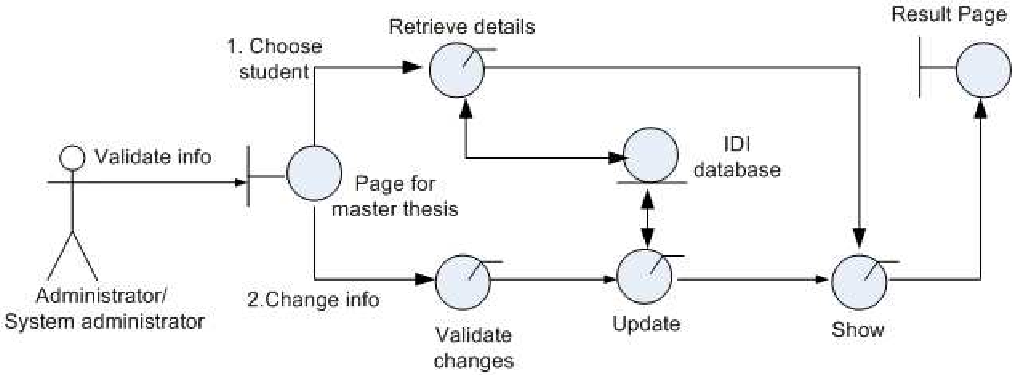


Table 9.9 describes the failure modes of use case "Validate information". Related to the robustness requirement defined earlier, the FMEA worksheet for these Control Objects are captured in Table 9.9

Table 9.9: FMEA for Validate information

System: DAIM Use case: Validate information			Performed by: Hue Pham Date: 06.04.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Retrieve details	No response is produced at all	Data is damaged and couldn't be retrieved	Fail to respond to user's interaction	Prevent further use of the system	Manage IDI database and ensure its correctness. Interact with "Show" to give feedback to the user
	Information found is incorrect	IDI database contains incorrect data	Incorrect data is presented to the user	User suspects the quality of the system	Manage data in IDI database and ensure its correctness

Continued on next page

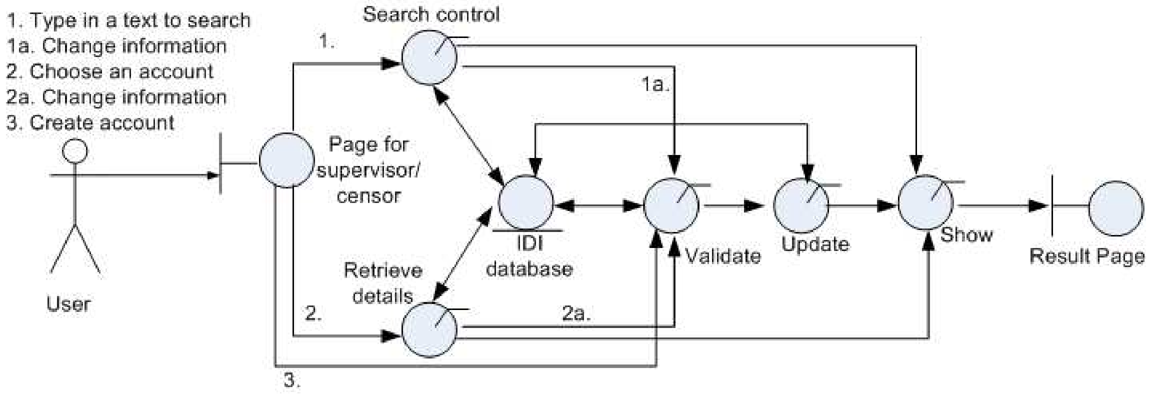
Table 9.9 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
Validate changes	Invalid inputs are not validated and will be saved or result in abnormal behavior	Error user input	Database will contain invalid data or fail to respond to user's interaction	Prevent further use of the system or user suspects the quality of the system	Control, validate user input and prevent serious errors from entering the object. "Page for master thesis" detects the failure of the object and interacts with "Show" to prompt the user
Update	No response is produced at all	Error output of "Validate changes"	Fail to respond to user's interaction	Prevent further use of the sytem	Control input from "Validate changes" and prevent serious errors from entering the object. "Update" detects the failure of the object and interacts with "Show" to prompt the user appropriately
Show	No response is produced at all	Error output of "Retrieve details", "Update"	Fail to responde to user's interaction	Prevent further use of the system	Control input from "Retrieve details", "Update" and prevent serious errors from entering the object
	Incorrect information is displayed	Error output of "Retrieve details"	Incorrect data is presented to the user	User suspects the quality of the system	Control input from "Retrieve details" and prevent serious errors from entering the object

9.1.11 UC11: Create/update account

An analysis diagram for the use case Create/update account is shown in Figure 9.11. A user can get information of a censor or a supervisor by choosing a name in a list or by entering a text to search. The system retrieves information of the selected account, i.e. supervisor or censor and displays to the user. The information can be changed and updated by the user. Sequence number 1, 1a, 2, 2a in Figure 9.11 will capture these tasks. A new student, censor or supervisor can also be add in the database by sequence number 3. Sequence number 1a and 2a are dependent on sequence number 1 and 2.

Figure 9.11: Analysis diagram for Create/update account



A FMEA table that described the robustness failure modes of the use case "Create/update account" is presented in Table 9.10.

Table 9.10: FMEA for Create/update account

System: DAIM Use case: Create/update account			Performed by: Hue Pham Date: 08.04.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Search control	Invalid inputs are not validated	Error user input	Invalid inputs will be saved	Users suspect the quality of the system	Control, validate user inputs and prevent serious errors from entering the object. "Page for supervisor/-censor" detects the failure of the object and interacts with "Show" to prompt the user appropriately

Continued on next page

Table 9.10 – continued from previous page

Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Retrieve details	Information found is incorrect	Error user input/ Database contains incorrect data	Incorrect data is presented to the user	User suspect the quality of the system	Control user input and prevent serious errors from enterign the object. Manage data in "IDI database" and ensure its correctness
Validate changes	Invalid inputs are not validated	Error output of "Search control", "Retrieve detais", error user input	Fail to respond to user's interaction	User suspects the quality of the sytem	Control input from "Search control", "Retrieve details", user input and prevent serious errors from entering the object. Interact with "Show" to display feedback to the user
Update	Invalid inputs are saved	Error output of "Validate changes"	Database will contain invalid information	Users suspect the quality of the system	Control input from "Validate changes" and prevent serious errors from entering the object. Interact with "Show" to display feedback to the user
Show	No response is produced at all	Error output of "Update", "Retrieve details", "Search control"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Update", "Retrieve details", "Search control" and prevent serious errors from entering the object.
					Continued on next page

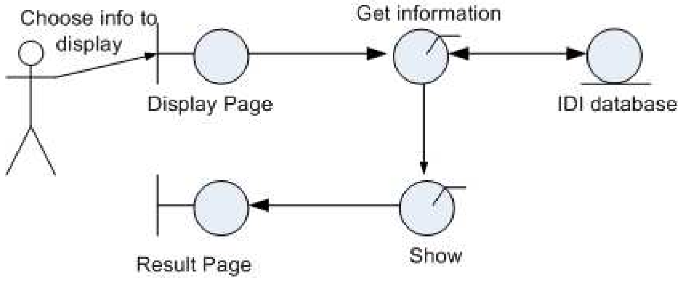
Table 9.10 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
	Incorrect information is displayed	Error output of "Update", "Retrieve details", "Search control"	Incorrect data is presented to the user	User suspect the quality of the system	Control input of "Update", "Retrieve details", "Search control" and prevent serious errors from entering the object

9.1.12 UC12: Get information

This use case contains several functions. The structure of each function is to select relevant informations in the "IDI database" and display to the user. For example the function "Get an overview of important deadlines for censoring" will select information of deadlines of censoring and display it to the user. The users choose information they want to display by choosing them. The system retrieves information from the "IDI database", selects relevant information and then displays it to the user.

Figure 9.12: Analysis diagram for Get information



The application logic of this use case is captured by "Get information" and "Show". The robustness failure modes of this use case are captured in Table 9.11.

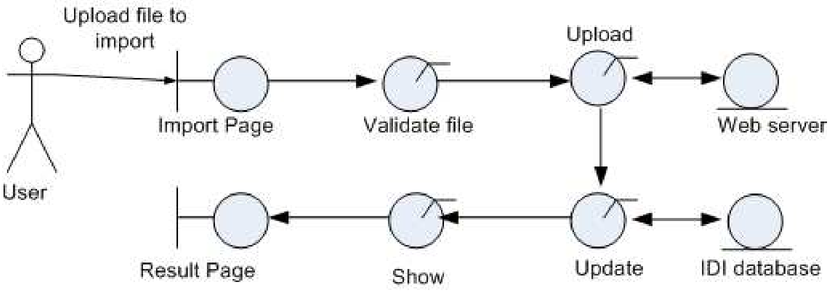
Table 9.11: FMEA for Get information

System: DAIM Use case: Get information			Performed by: Hue Pham Date: 09.04.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Get information	No response is produced at all	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. "Display Page" detects the failure of the object and interacts with "Show" to prompt the user appropriately
	Information found is incorrect	IDI database contains incorrect data	Incorrect data is presented to the user	User suspects the quality of the system	Manage data in "IDI database" and ensure its correctness
Show	No response is produced at all	Error output of "Get information"	Fail to response to user's interaction	Prevent further use of the system	Control input from "Get information" and prevent serious errors from entering the object.
	Incorrect information is displayed	Error output of "Get information"	Incorrect data is presented to the user	User suspects the quality of the system	Control input of "Get information" and prevent serious errors from entering the object

9.1.13 UC13: Import data

A system administrator can import data of students or censors from FS and save in database. FS is national database for every universities in Norway which NTNU has a copy of FS in a local database that contains informations on students and censors. Data of the censors need to be imported one time only, and the administrator user can change the information afterwards. Students data have to be import once every year. The file have to be in CSV format and will be uploaded to the Web server and further read and save in the database. It's important that the file has the same format as defined in the system. The existed students in the database will not be overwritten. The analysis diagram for this function will be illustrated in Figure 9.13

Figure 9.13: Analysis diagram for Import data



The file is validated and then upload to the "Web server" if it's a valid file format. The file will further be updated to the database and the result will be displayed by "Show". The robustness failure modes are charted and captured in Table 9.12.

Table 9.12: FMEA for Import data

System: DAIM Use case: Import Data			Performed by: Hue Pham Date: 09.04.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Validate file	Invalid file are not validated and will be saved or caused abnormal behavior	Error user input, imported file is damaged or invalid format	Database contains invalid information or fail to respond to user's interaction	Users suspect the quality of the system. Prevent further use of the system	Control user input and prevent serious error from entering the object. "Import Page" detects the failure of the object and interacts with "Show" to promt to the user appropriately

Continued on next page

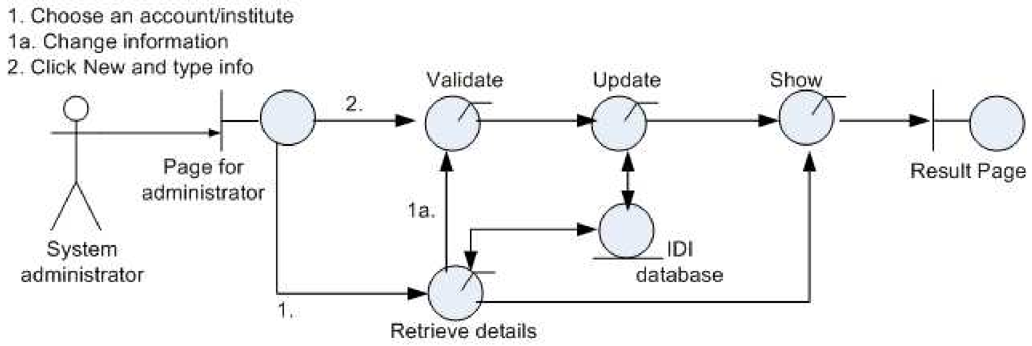
Table 9.12 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
Upload	Abnormal behavior	Error output of "Validate file"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Validate files" and prevent serious errors from entering the object. "Validate file" detects the failure of the object and interacts with "Show" to prompt the user appropriately
Update	Abnormal behavior or invalid data will be saved	Error ouput of "Upload"	Fail to respond to user's interaction. Database contains incorrect information	Prevent further use of the system. Users suspect the quality of the system	Control input from "Upload" and prevent serious errors from entering the object. "Upload" detects the failure of the object and interacts with "Show" to prompt the user appropriately
Show	No response is produced at all	Error output of "Update"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Update" and prevent serious errors from entering the object. "Update" detects the failure of the object and interacts with "Show" to prompt the user appropriately

9.1.14 UC14: Change information

Figure 9.14 is the result of using the Jacobson's analysis on this use case. The sequence number 1a. from Figure 9.14 is dependent on sequence number 1. Sequence number 1 and number 2 are independent of each other. Information of an administrator can be changed by choosing the administrator and then change information. Information of the system administrator users can be changed by clicking the "My Institute" button

Figure 9.14: Analysis diagram for Change information



and then change displayed information. "Retrieve details", "Validate", "Update", and "Show" are used to choose an account/institute and change information.

A new administrator can be add by pressing the "New" button and filling in the required information. "Validate", "Update" and "Show" modules are used to perform the task with the sequence number 2. The robustness failure modes of these Control Objects are charted in Table 9.13.

Table 9.13: FMEA for Change information

System: DAIM Use case: Change information			Performed by: Hue Pham Date: 09.04.06		
Control Object	**Robustness failure mode**	**Possible cause**	**Local effect**	**System effect**	**Preventive means**
Retrieve details	No response is produced at all	Error user input	Fail to respond to user's interaction	Prevent further use of the system	Control user input and prevent serious errors from entering the object. "Page for administrator" detects the failure of the object and interacts with "Show" to prompt the user appropriately

Continued on next page

Table 9.13 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
	Information found is incorrect	Error user input / IDI database contains incorrect data	Incorrect data is presented to the user	User suspects the quality of the system	Control user input and prevent serious errors from entering the object. Manage data in "IDI database" and ensure its correctness
Validate	Abnormal behavior or invalid inputs will be saved	Error user input/ error output of "Retrieve details"	Fail to respond to user's interaction. Database contains invalid data	Prevent further use of the system. Users suspect the quality of the system	Control user input, input from "Retrieve details" and prevent serious errors from entering the object. "Page for administrator" detects the failure of the object and interacts with "Show" to prompt the user appropriately
Update	No response is produced at all or invalid inputs are saved	Error output of "Validate"	Fail to respond to user's interaction. Database contains incorrect information	Prevent further use of the system. Users suspect the quality of the system	Control input from Update and prevent serious errors from entering the object. "Validate" detects the failure of the object and interacts with "Show" to prompt the user appropriately
Show	No response is produced at all	Error ouput of "Update", "Retrieve details"	Fail to respond to user's interaction	Prevent further use of the system	Control input from "Update", "Retrieve detail" and prevent serious errors from entering the object. Prompt the user appropriately

Continued on next page

Table 9.13 – continued from previous page

Control Object	Robustness failure mode	Possible cause	Local effect	System effect	Preventive means
	Incorrect information is displayed	Error output of "Retrieve detail"	Incorrect data is presented to the user	User suspect the quality of the system	Control input from "Retrieve detail", "Update" and prevent serious errors from entering the object

9.1.15 Conclusion

After considering these use cases by using the framework that J. Zhou et al. [24] has proposed, I believe that the framework can be used as an early assessment method for robustness. Both Jacobson's analysis model and FMEA can give information at different areas. The Jacobson's analysis decomposes the system into components or modules, which enables us to reason about the structure of the moduls in the system at an early stage.

Furthermore, the FMEA table captures the failure modes, the effects and preventive means which are important. The FMEA can also be used as a checklist at the design or implentation stages, check that every failure mode is handled and the preventive means are implemented.

9.2 Application of Hazard analysis to Software Quality Modelling

After studying the framework that H. Zhu [26] has proposed, I have decided not to use this framework to assess the robustness of DAIM system. Some use cases of DAIM are considered to illustrate the framework. The quality models and SFMEA of selected use cases are included in Appendix B.

After constructed some quality models and charted the SFMEA, I found that this framework is more suitable for assessment and estimating several quality attributes at the same time, since the proposed quality model enables explicit references to the components of the information systems.

The project focuses on the input robustness, and thus only failure modes related to invalid user input are of interest here. The SFMEA and quality models will not stop to consider failure modes that is related to input robustness, but will further consider failure modes and causes, which related to other quality attributes. Thus, it's easy to lose the focus and end up with several not interesting failure modes or quality attributes.

Furthermore, both SFMEA and the quality models give us the same informations. Many information from the quality models are repetition of the SFMEA table. The "relevant quality property" is only the new one information that can be found in the quality model, but not in the SFMEA table.

The principle of the causes and subcauses as charted in the SFMEA is an importan concept. Thus, the component that contains the root cause for a failure becomes clear. The SFMEA can be used as a tool to construct the quality model. Based on found failure modes and causes, the quality model is constructed. During constructing the quality model, you can go back and change the SFMEA if something is missing. Changing in SFMEA will, however, result in changes to the quality model. Thus, if you have done a good job in SFMEA, you will avoid a lot iteration of both the SFMEA and the quality model.

The framework that H. Zhu [26] has proposed is suitable at an early stage, such as the functional requirement analysis and the architecture design where the major components and functional requirements of the system are defined. Information of the module's structure is not required. Thus, the framework will not give informations of the detail design stage. The preventive means are not described, but the explanations of the failure modes are captured. Thus, descriptions and explanations of the failure modes and the causes of each failure mode in the use case are the main focus.

9.3 Estimate DAIM's robustness

Based on previous works of the qualitative robustness assessment, the mathematical expression from [19] will be used to give an estimating of the current robustness of DAIM. This section presents how the current robustness of DAIM is estimated, by using the principle described by T. Stålhane in [19], with the changes I propose below, i.e. the changes of the FMEA table and the mathematical expression.

The author in [19] proposes to use classes as starting point, but the adapted FMEA has use cases as starting points. The adapted FMEA table for robustness is captured in Table 9.14.

The FMEA table for robustness is a bit different from the original FMEA table presented in [19]. The columns (1) and (4) are changed and have the following interpretations:

- Column (1), it will contain Control Objects that have been considered by Jacobson's analysis model.
- Seriousness: An estimation of how serious this failure mode is. The entry will be ranked from 0 to 5. The value 5 is highest and tells us that this is a critical failure mode. The value 0 is lowest and the failure mode is not critical at all. This is captured in column (3).

Table 9.14: FMEA with class as starting point

Use case ID:					
Control object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
(1)	(2)	(3)	(4)	(5)	(6)
Use case sums:					
Robustness for the use case:					

- Mitigation: The value of mitigation tells us the system level when this failure mode is handled. The value could be between 0 and 5. Value 0 is the lowest level and means that the failure mode is not handled at all. The failure mode is handled and considered if the value is 5. This is described in column (4).
- The columns (2), (5) and (6) have the same interpretations as described in section 5.1.3.
- At the end of the FMEA table, the sum of seriousness, the sum of mitigation, and the robustness value will be calculated.

Based on the changes we have done, the robustness for use case is estimated from the below expression:

$$Robustness(i) = \frac{Mitigation_i}{Seriousness_i} \tag{9.1}$$

Based on the previous definition of robustness for a use case, the robustness for an entire system is estimated as follows:

$$Robustness = \frac{\sum Mitigation}{\sum Seriousness} \tag{9.2}$$

The identified failure modes in section 9.1 will be used to estimate the robustness of each use case. The estimating of the robustness is included in Appendix C.

A summary of the robustness of all use cases is described in Table 9.15. At the end of the table, the robustness average is identified. The robustness average could also interprete as the robustness of DAIM system.

Table 9.15: Summary Robustness Estimating

Use case ID	**Use case name**	**Seriousness**	**Mitigation**	**Robustness**
UC01	Log in	13	11	0.85

Continued on next page

Table 9.15 – continued from previous page

Use case ID	Use case name	Seriousness	Mitigation	Robustness
UC02	Fill in contract	23	11	0.48
UC03	Create cooperation group	28	22	0.78
UC04	Generate contract/schema	19	16	0.84
UC05	Deliver master thesis	22	17	0.77
UC06	Search for master thesis	18	4	0.22
UC07	Handle payment	32	28	0.87
UC08	Display master thesis	18	10	0.55
UC09	Choose censor	20	17	0.85
UC10	Validate information	29	15	0.52
UC11	Create/Update account	29	9	0.31
UC12	Get information	18	16	0.89
UC13	Import data	15	14	0.93
UC14	Change information	30	7	0.23
Robustness Average		314	187	0.59

The optimal robustness value is 1. DAIM has the robustness value at 0.59. This is too low value. Remember that DAIM is dependent on external database. DAIM must connect to Innsida, Tapir Web server and Tapir database server systems that the DAIM system has no control over. Furthermore, DAIM's database is imported from FS database, which could contain old data and the database is not necessarily 100% up to date.

In addition, human factor plays a major role for the correctness of system's database. It's important that the administrator user has to change or update database manually if the information is changed. For example, if the censor has a new adress or telephone number, the administrator user must change information manually to hold the database up to date. The mentioned factors have influence in the robustness of the DAIM system.

9.4 Conclusion

In this chapter, the DAIM's robustness is considered by using the frameworks from [24, 19]. The contribution of this chapter is two-fold. First, the DAIMs robustness is studied and a rough assessment of the robustness by using the framework from [24].

The robustness failure modes of each use case are described and studied. In order to get a better understanding, the preventive means of the failure modes also were described.

Second, the robustness of each use case is further estimated by using the principle from [19], with the proposed changes. The robustness value will be a number between 0 and 1, dependent on the failure modes and how well the failure modes are handled. The robustness of the DAIM system is estimated and have a value as 0.59. In Section 8.1.2, the robustness is defined as acceptable if it's 0.90 or higher. The estimated value of robustness by using early assessment methods is too low, compared with the robustness goal for DAIM system.

The framework proposed by H. Zhu [26] was studied by considering some use cases of the DAIM system. A short evaluation and conclusion of this framework also included.

Part IV

Testing DAIM

Chapter 10

User Testing Session

In this chapter we present the process of testing DAIM. An description of the possible type of software responses and the templates that we're going to use for testing will also include in this chapter.

Many use cases described in Section 8.2 require the users to type in input to perform testing. The others let the users choose the inputs from a list, check box, or radio button. In order to test the input robustness, only invalid inputs will be tested. In the cases where the users can not fill in invalid inputs, the tests are intended to check that system should respond to the user's action due to possible internal component failure.

The tests are automated as far as possible. The login page for students is statical, thus the passwords and usernames can be filled in automated. The other roles used the "Innsida" as the login page and the "Innsida" a dynamical Web page. Thus, the usernames and passwords of these roles must be filled in manual, even if the tests are automated.

The tests data are included in Appendix D. This appendix includes also the details description of the test specification if the tests are manually performed. The system I'm testing is located at NewDAIM[1]. NewDAIM is the same as DAIM, but it was established for testing and used the test database. The use case "UC06: Search for master thesis" will be test at DAIM[2] because the search module is updated at the DAIM, but not in the NewDAIM. In addition, database in the DAIM contains more data than in the NewDAIM.

There are two levels of input that will be used to test the DAIM system; the parameter level and the value level [16]. At the parameter level, some parameters are removed from the form and the form is then submitted. At the value level, various values for parameters are tried, including values that are normally not expected by the software.

[1] http://newdaim.idi.ntnu.no

[2] http://daim.idi.ntnu.no

When the DAIM system receives invalid input, there are four following possible types of software responses:

- Type 1: The invalid inputs **are recognized** and adequately processed by the software. System prompts the user without saving the invalid inputs. The feedback must be intelligible without any special computer knowledge.

- Type 2: The invalid inputs **are recognized**. Invalid inputs are **changed to default values and saved in the DAIM's system without prompting** the user. This is not a critical failure.

- Type 3: The invalid inputs **are not recognized** and will **be saved in database without prompting** the user. The invalid inputs will not cause abnormal software behavior. This is a critical failure.

- Type 4: The invalid inputs are **not recognized** and the abnormal software behavior is **exposed** directly to the users. This is a critical failure

These behaviors are the most common, but the system can have other responses. Abnormal software behavior includes responses like run time exceptions, system can not do what it is intended to do or revealing confidential information to unauthorized user. The type 1 response is the proper software behavior, while type 3 and type 4 responses represent inadequate software behavior and are considered to be failures. According to the definition of robustness for the DAIM system as described in Section 8.1.2, type 2 behavior will also be considered failures.

It will be interesting if the DAIM system can be tested by using several browsers or browsers that system is intended to support. Because of the time pressure, only the Internet Explorer Version 6.0 is used for testing. Table 10.1 is the template that will be used for testing the DAIM system.

Table 10.1: Test data for <Use case ID: use case name>

<Use case ID: use case name>		Date:
Precondition	(0)	
Field: Input Data/ Actions	Expected Reaction	Reaction
(1)	(2)	(3)
Total tests:		Tests passed:

The table headings from Table 10.1 have the following interpretations:

- An description of preconditions is recorded in row (0). It also contains actions that should be done before any user input is submitted.

- The column (1): This column can be either "fields: inputs data" or "actions", dependent on the test is automated or manually performed. Fields are labels that the Web pages will be shown at the browser beside text fields, text areas, select box and so on. I have not translated these fields onto English because it will result in confusions.

- The expected reaction is presented in column (2). Usually, the expected reaction refers to type 1 of the software responses that described aboved.

- The DAIM's response is described in column (3). Sometimes, the reaction is referred to the type of software reactions that described above. In order to answer the metrics described in the Chapter 7, some observations that can be used to answer the metrics will also be included here.

- At the end of the table, the total number of tests and tests passed are summarized.

All test cases in AutAT, Watir and test files that I've used to test the DAIM system are included in an attachment. The file lesmeg.txt in the attachment will explain what different folders contain.

Chapter 11

Test Results

The purpose of this chapter is to analyse and evaluate the robustness of the DAIM system with respect to the GQM-model. The metrics in Chapter 7 will be answered by testing the DAIM system. Some metrics can be answered by using AutAT to perform the tests, others require manual tests. We assume that the AutAT and Watir are installed before the tests can be performed. The outcome of applying each metric defined in Chapter 7, and how it affects the questions and the goal will describe here. It ends with a section on the threats to validity for the method used for testing.

11.1 Metrics

The data collection and the results for each metrics are presented here.

11.1.1 M1: Error tolerance

This metric can be answered by testing the robustness of the DAIM system. A summary of the tests data for the functional requirements of the DAIM system, which included in the Appendix D are captured in the Table 11.1. The table heading of the Table 11.1 are as follows:

- The column "Use case ID" contains ID of each use cases
- The column **M**anual/**A**utomatic (**M/A**) tells us whether the test is performed manually (M) or automatically (A).
- The column "Test specification" contains the name of the file in both AutAT and Watir, i.e the file name "UC01" is both UCO1.aat (AutAT format) and UC01.rb (Watir format). This column can also contain a table ID that included the test case if the tests are performed manually.
- The column total tests (T) records the total number of tests that are performed for the use case.
- The total number tests passed (TP) are recorded in the second to last column.

- An estimating of the robustness for each use cases are shown in the last column. The robustness for each use case is computed by the total number of tests passed (TP) divided the number of total tests (T).

Table 11.1: A summary of test results for each use case

Use case ID	**M/A**	**Test specification**	**Total tests**	**Total tests passed**	**Robustness**
UC01	A	UC01	2	2	1.0
UC02	A	UC02Biblio UC02DateDeadline UC02Students UC02Supervisor	25	10	0.40
UC03	A	UC03	2	2	1.0
UC04	A	UC04	3	3	1.0
UC05	M	Tables D.8 to D.11	17	13	0.76
UC06	A	UC06	8	0	0
UC07	M	Table D.14	2	2	1.0
UC08	M, A	D.15 and UC08-03	15	9	0.60
UC09	M	Table D.17	2	2	1.0
UC10	M	Table D.18	11	5	0.45
UC11	A	UC11-01 UC11-02 UC11-03	25	7	0.28
UC12	A	UC12	4	4	1.0
UC13	M	Table D.24	6	6	1.0
UC14	M, A	UC14-01 Table D.26	11	0	0
		Total	T = 131	TP = 65	0.50

$$Robustness(X) = \frac{TP}{P} = \frac{65}{131} = 0.50 \tag{11.1}$$

The inputs robustness for every use cases vary from 0 to 1 and the average robustness for all use cases is 0.50. The most frequent responses too wrong input was type 3. This is a critical failure. Only three times system reacted as type 4, which is considered a critical failures. The critical failures occured during testing the use case "UC06: Search for master thesis", which tells us that the search module are not robust. The failure of type 2 occured four times. Most failures occured because the system didn't validate invalid inputs. Hence, the invalid or meaningless data is saved without prompting the user.

Conclusion: The DAIM system has a input robustness of approximately 0.5. The failures that occured are critical failures and will result in incorrect or meaningless information are saved in database. A few failures will cause abnormal software behaviour, such as the search module. I feel that the system has low error tolerance. Invalid inputs are saved without prompting the user. Sometimes, invalid input are recognized and changed to default values without prompting the user. There is also a chance that erroneous data are accepted and cause abnormal behaviour in some cases.

11.1.2 M2: Error messages

This metric can be answered by observation the error messages during testing. The appendix D includes data that was collected to answer this metric. The column "Reaction" describes the system's reaction. The expression "As expected" is used in the row where the system reacted as expected.

To collect data for this metric, go through all tables in the appendix D. Count the error message(s) in a row that has both the expression "As expected" in the column "Reaction" and the intelligible feedback is expected in the column "Expected reaction". One row may include several error messages. A comment will also be included in the column "Reaction" if the expected feedback is not intelligible.

A function can be tested by various user inputs and the system can print the same error message several times. For instance, during testing the login page, the system can print the error message "Invalid username and password" two times if incorrect usernames and passwords are filled in two times. Many of the error messages occurred several places in the tests, only one of each has been taken into consideration.

Based on the appendix D, data are collected and shown in the Table 11.2. Column **B** in Table 11.2 captures the total number of error message that system has printed for each use case. The number of error messages that are possible to understand without any special computer knowledge are recorded in the column **A**.

Table 11.2: Collected data for M2: error message

Use case name	**B**	**A**
UC01: Log in	1	1
UC02: Fill in contract	3	3
UC03: Create cooperation group	2	2
UC04: Generate contract/schema	0	0
UC05: Deliver master thesis	11	11
UC06: Search for master thesis	0	0
UC07: Handle payments	2	2
UC08: Display master thesis	2	2
UC09: Choose censor	0	0
UC10: Validate information	5	5
UC11: Create/Update account	7	7

Continued on next page

Table 11.2 – continued from previous page

Use case name	B	A
UC12: Get information	0	0
UC13: Import data	6	6
UC14: Change information	0	0
Total:	32	32

$$X = \frac{A}{B} = \frac{32}{32} = 1.0 \tag{11.2}$$

Conclusion: The error messages in the system are meaningful and intelligible.

11.1.3 M3: Feedback

Appendix D can also be used to collect data to answer this metric. During testing, if a confirm or register function that has **not** produced a clear confirmation message or action, denoted as **(!A)**, will be described in the column "Reaction".

Table 11.3 identifies the collected data for this metric. The total number of confirm/register functions are captured in column B. The same button can be clicked several times and produced several clear confirmation messages/actions, dependent on the input data. The following rules are used to count the value of B for each use case:

- If the **same button** is clicked several times and **produced the same** confirmation messages/actions, only **one** of each message/action will be taken into consideration.
- If the **same button** is clicked several times and **produced different** confirmation messages/actions, **all** of them will be taken into consideration.

For instance, two valid usernames and passwords were filled in and the button "Logg inn" was clicked two times. The value B is 1 if both of them produced the same actions. If both valid and invalid usernames and passwords were filled in and the button "Logg in" was clicked, the value B is 2 because both of them produced two different actions/confirmation messages.

The column A records the number of confirm/register functions that produce a clear confirmation message or action. The value A for each use case can be computed by:

A = B - (!A)

In order to compute the value of this metric, two tasks must be done. First, the value B can be counted during testing, by observation the number of confirm/register function and the actions that have been produced for each use case. Second, go through all tables in Appendix D, count the number of actions that has not produced a clear confirmation message or action (!A).

Table 11.3: Collected data for M3: Feedback

Use case name	B	A
UC01: Log in	1	1
UC02: Fill in contract	16	15
UC03: Create cooperation group	4	4
UC04: Generate contract/schema	2	2
UC05: Deliver master thesis	28	28
UC06: Search for master thesis	4	4
UC07: Handle payments	4	4
UC08: Display master thesis	14	14
UC09: Choose censor	2	2
UC10: Validate information	7	7
UC11: Create/Update account	15	15
UC12: Get information	4	4
UC13: Import data	18	18
UC14: Change information	7	5
Total:	126	123

$$X = \frac{A}{B} = \frac{123}{126} = 0.98 \qquad (11.3)$$

Conclusion: The system responds to user actions in an intuitive and consistent way.

11.1.4 M4: DAIM precision

This metric can be computed by navigating through all Web pages in the DAIM system and counting the number of fields that required special syntax input (B), i.e. date format. In addition, the number of special syntax input fields that have an explanation (A) will also be counted.

Table 11.4 presents the collected data to answer this metric. Only use cases that require special syntax will be included in Table 11.4.

Table 11.4: Collected data for M4: DAIM precision

UC ID	B	A	Explanation
UC02	2	2	Date format in the "Student(er)" area and "Datoer og frister"
UC03	1	1	Usernames in the cooperation is separated by comma. This is well explained
UC05	4	4	The picture of front-page, zip file, pdf file, the colour pages to print. All are well explained.

Continued on next page

Table 11.4 – continued from previous page

UC ID	B	A	Explanation
UC08	1	1	The date format that used to search the master thesis in a period. A default value is displayed and the user can use this as a template
UC10	5	0	The user has to fill in 5 date format as input, none of them are explained.
UC11	2	0	The page for a censor, in the area "Oppnevnt" contains date format for the fields "Fra dato" and "Til dato". But the date format does not explain.
UC13	2	2	The CSV files format that used to import data for students and censors are well explained.
UC14	2	0	UC14-01 used two fields "Kostnadssted" and "IBX number" that have special syntax, but the syntax was not explained
Total	19	10	

$$X = \frac{A}{B} = \frac{10}{19} = 0.53 \tag{11.4}$$

The system used the same date format for all use cases, but some Web pages have clearly defined syntax, and the others have not. Seven of the nine fields that lacked clearly defined syntax descriptions could be considered self-explanatory, since several identical fields with a complete description were used. If these were considered sufficiently described, X would be 0.89.

Conclusion: The user will in 53% of the cases be able to determine how to enter data into the system.

11.1.5 M5: Syntactic Input Validation

To collect data for this metric, all Web pages of the DAIM system must be visited. Count the number of fields that user **can fill in** by **choosing** a select box, check box, text field with included max-length etc. The number of fields that it's **possible to use** select box, check boxes, etc. to **avoid** invalid user input (B) will also be counted.

Table 11.5: Collected data for M5: Syntactic input validation

UC ID	B	A	Explanation
UC02	11	9	The text field for student's date of birth can include max-length. But this was not handled The text field for "Oppstartdato" in the area "Datoer og frister" doesn't include a max-length
UC05	4	2	The text fields for "Totalt antall sider" and "Antall ekstra kopier" can include max-length. But these were not handled
UC06	8	8	The advanced search has used 8 select boxes that can prevent invalid user input. Well handled
UC08	20	18	The date format that used to change a period that reports of master thesis were saved in the system can be included max-length. But this is not handled
UC09	1	1	The censor is chosen from a select box
UC10	11	2	All text fields for date format can include a max-length, but none of them were handled. Only two select boxes are used for choosing a period to restrict either the master thesis or only the attachment of the master thesis.
UC11-02	3	3	Well handled
UC11-03	10	6	The text fields for "Fødselsdato/Personnr", "Fra dato" and "Til dato" can include max-length. But this is not handled
UC11-04	3	3	Well handled
UC14-01	6	0	All text fields that used can include max-length, except the text area for "Adresse". None of them were handled
UC04-02	8	8	Well handled
Total	105	60	

$$X = \frac{A}{B} = \frac{60}{105} = 0.57 \tag{11.5}$$

The syntatic input validation, such ascheck boxes, select boxes, etc. were not sufficiently used to avoid invalid user inputs. Most inputs can use a text field with included max-length to avoid the users submit invalid inputs, but this was not utilized by DAIM.

Conclusion: The syntatic restrictions are not sufficiently used to avoid invalid user inputs.

11.2 Questions

The metrics in the previous section are used to answer the questions in the GQM-method as described in Chapter 7. A summary of the results from the applications of the metrics are presented in this section to answer the questions.

11.2.1 Quesion 1: Input Tolerance

Answer from the metric M1 gives us enough information to answer this question. M1 tells us that the robustness of the DAIM system is 0.50. Specially, the robustness of the search module (with use case ID "UC06") which perform the search for external user is too low. The system can not do what it's intended to do when invalid input were submitted in the use case UC06. Furthermore, the system omitted a lot of data value validation, such as allowing integers to be submitted and saved in the fields where it is reasonable to submit the letters as inputs data, and conversely.

However, the validation for parameter constraint is well handled. The system checks if the selected file type and the file submitted really match. For example, it is impossible to select the file type to be pdf, but submit an doc file instead.

Answers from "M2 - Error messages" and "M3 - Feedback" help us to understand other characterizations that may influence the robustness of the DAIM system, such as error messages and system's feedback. The error messages in the system are meaningful and intelligible. Furthermore, the system responds to user actions in an intuitive and consistent way. This contributes to increasing the input robustness of the DAIM system.

11.2.2 Question 2: DAIM interface

This answer depends on answers from metrics "M4 - DAIM precision" and "M5 - Syntactic input validation". In nearly half of the cases the users will not be able to determine how to enter correct data into the system. This can cause the users to submit invalid input.

Many of the fields that lacked clearly defined syntax descriptions could be considered self-explanatory, since several indentical field with a complete description was located nearby.

The syntactic input validation can be used to decrease the chance that the users submit invalid inputs. More than half of the cases (57%) that the system used the syntactic validation to avoid user submit invalid input.

To conclude, the interface contributed to increase the input robustness of the DAIM system in more than half of the cases. This is not an optimal value. The closer this value is to 1, the more interface is used to avoid invalid user input and possibly contribute the input robustness. The DAIM system has not used the interface to sufficiently avoid user invalid inputs.

11.3 Goal Attainment

The Goal is based in the GQM-method described in the Section 7.1. As seen from the tester's point of view I conclude that the input robustness of the DAIM system is low. The robustness for each use case vary from 0 to 1. Only the invalid inputs for the use case "UC06: Search for master thesis" can result in software abnormal behavior and the system can not do what it is intended to do.

In most cases, the system omits a lot of data value validation. Hence, invalid inputs were saved in system without prompting and resulted in incorrect/meaningless data in database. These invalid inputs will not cause abnormal software behavior, but we considered as critical failures.

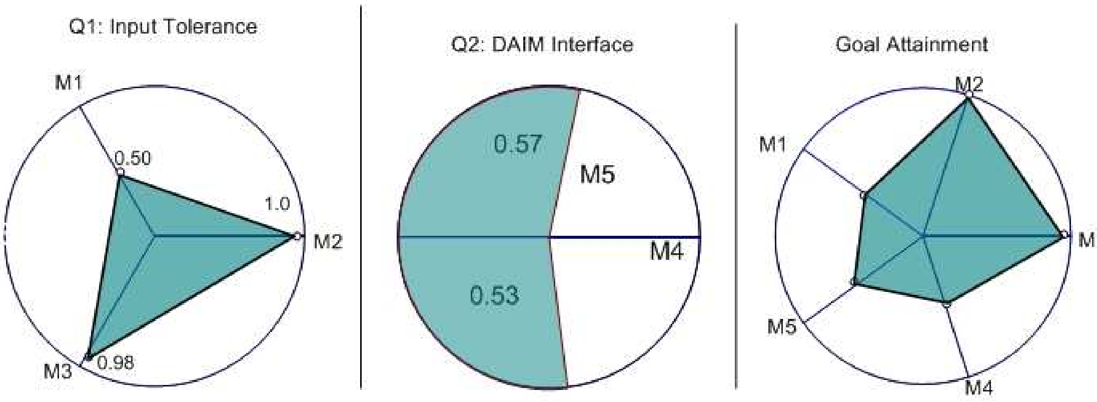
However, the error messages and system's responses contribute to increase the input robustness. The validation for parameter constraint is well handled. Only validations at value level are poor handled.

One way to increase the input robustness is to prevent the user from submitting invalid input. This can be done by explaining the input with special syntax, or let the user choose input from a list, select box, included max-length in a text field etc. Thus, the user interface can be used to avoid invalid user input and possibly contribute the input robustness. But this is not well handled in the DAIM system. Only 57% of the cases where the DAIM used the syntactic restrictions to avoid invalid input. The user will in 53% be able to determine how to enter dat into the system.

For easy viewing metrics and goal attainment, a Kiviat diagram is produced, like the one shown in Table 11.1, which graphically depict the collected data in the metrics M1, M2, M3, M4 and M5. Each "spoke" of the Kiviat chart represents a metric, and the results are plotted.

The questions are separately illustrated in Figure 11.1, based on answers from metrics. As the subfigure "Goal Attainment" illustrated, the issues related to the metrics M2 and M3 are well handled, but M1, M4 and M5 are poorly handled.

Figure 11.1: Kiviat diagram for questions and goal attainment



11.4 Threats to Validity

The most important threats to validity will be presented in this section. The list of threats that is presented here is based on a set of threats proposed by Wohlin et al. [22].

Wohlin et al. [22] discuss validity threats especially for experiments, but some of these might also be a threat to other empirical strategies, for example a case study as in our case. There are different classification schemes for different types of threats to the validity of an experiment. Wohlin introduce four types of threats; *conclusion, internal, construct and exteneral validity.* This categorization will not be used directly and each threat will not be described in detail. However, the list of threats from Wohlin [22] will be used as a basis for finding the validity threats for testing in this project. This validation mainly concerns the results of testing the DAIM system.

- **Low statistical power**
  The power of a statistical test is the ability of the test to reveal a true pattern in the data. All use cases in the DAIM system has been tested and various invalid inputs were used. The partitioning technique is used to test one representative value in each partition and considered as the whole input space. Considering the total number of tests, it seems like we have enough answers to draw an overall conclusion.

  Some factors that can have influence the true pattern are the test database contains rather few data and two use cases can not be adequately tested, namely use cases UC07 and UC12. Furthermore, only the Internet Explorer 6.0 is used to test while the DAIM system is intended to support both Internet Explorer and Firefox. This factor will not influence the result if the conclusion is generalized only for Internet Explorer Version 6.0.

- **Fishing**
  Searching or "fishing" for a specific result is a threat, since the analyses are no longer independent and the tester may influence the result by looking for a specific outcome and possibly overlook other results. There are other quality attributes

that may influence the result, such as reliability and correctness. We were trying to test and estimate the input robustness of the DAIM system, but other quality attributes were not included.

- **Reliability of measures**
  One should get the same outcome if one measure a phenomenon twice. In other words, objective measures, that can be repeated with the same outcome, are more reliable than subjective measures. The collected data depend on the observation during testing. The observation involves subjective judgement. For example, collecting answers for the metric M3 is dependent on the human judgement. When the system has performed a register function and produced a clear confirmation message. Definition of a clear confirmation message is dependent on human judgement. However, rules and guidance to how one should collect data helps.

- **Instrumentation**
  This is the effect caused by the artifacts used for the experiment execution, such as data collection forms. The AutAT program that has been used to automate the tests may also influence the result. Since a report can not be generated by the AutAT, and data must be collected by human observation. There is a risk that not all details are recorded and described. A template is used to collect data during testing, if this is badly designed, the test is affected negatively. There is a risk that I've omitted some data and the template is not good enough.

- **Selection**
  This is the effect of natural variation in input data performance. Dependent on how the input data are selected to test the input robustness, the selection effects can vary. There is a chance that some types of data are omitted and have not been tested.

To conclude, there are a lot of factors that can have influence in the results of analysis and evaluating the robustness of the DAIM system. Firstly, the test database contains rather few data and not all use cases will be tested suffficiently. Furthermore, only the input robustness is estimated and tested, but other quality attributes that may possibly have influence the robustness are not included. It's impossible for us to say anything about other quality attributes that have affects on the robustness of the system.

The process of collecting data to answer the metrics involves a subjective judgement. A subjective measure is not reliable as an objective measure, such as result of the test tool can not be generated as a report, to collect data during testing by observation. By that very fact the subjective judgement involves in the most cases, possibly it will have influence the realibility of the result. The selection effects of input data can also vary, dependent on the selected input.

In most cases, several tactics are used to minimize the influences between the threats to validity and the outcome of testing. For example, rules and guidance to how one should collect data is used, and the partitioning technique is used to select a representative input from a partition. Even if the threats are well handled, there is still a risk that threads described above have influence in the result. But the risk will be inside an acceptable value and will not have a big influence in the result.

Part V

Evaluation

Chapter 12

Discussion and Evaluation

The research in this project has been divided into two major parts; a judgement of assessment methods that can be used early to estimate the robustness for Web-based system and testing for robustness. The first part concerned two areas of interests, a rough evaluation of assessment methods and a method used to estimate the robustness of the DAIM system. The goal of the first part is to make sure that the chosen methods can really be used to assess the robustness of a Web-based system.

The second part consists of evaluation and analysis of the robustness of the DAIM system by testing. The purpose of this part is to collect data or information that can tell us whether the results from the first part are valid or not. Based on the data collected during testing, the robustness is estimated. This project ends with an evaluating and discussion of the results from the first and the second part.

12.1 Conclusion and Discussion

A short summary and discussion of the two major parts of interest present in this section.

12.1.1 Early robustness assessment methods

A combination of both qualitative and quantitative assessment methods give us a robustness estimating of the DAIM system at 0.59. The Jacobson's analysis model enables us to organize the structure of the system, and the FMEA is used to analyse the robustness failure modes. The level of the Jacobson's analysis modelling can vary in details, i.e. it can vary from a use case to a user story. A use case can consist of one or several user stories. In this project, the Jacobson's analysis method is illustrated at a use case detail level, which is considered a high level. Each use case is illustrated by a Jacobson's analysis model and a FMEA table.

Since the modelling is at a high level, analysis of the robustness failure modes and possible causes naturally will also end at a high level, i.e. error user input or abnormal behavior. The possible cause "error user input" can be split up in to illegal character, invalid format etc. The robustness failure modes and causes are considered at a high level, mainly due to the chosen level of Jacobson's analysis model, but also because this assessment method will be used at an early stage, where we still lack much informations.

To consider each use case by illustrating the Jacobson's analysis enables us to organize the structure of the modules in the system at an early stage. The FMEA has been used to check that every failure mode is handled and that the preventive measures are implemented. By using this framework, we get enough informations to assess the robustness at an early stage. The acceptable robustness value is dependent on the goal and will vary from project to project.

To analyse the consequences and indicators of each failure mode, a quantitative method proposed by T. Stålhane [19] is used. In our project, the Web-based system has been implemented and the FMEA table from [19] has been changed. The robustness is estimated by the value of seriousness and mitigation from the FMEA table. A seriousness value is used to set the seriousness of each robustness failure mode and the mitigation value tell us how well this failure mode has been handled. The mitigation value is used because the system is already implemented. The mitigation can be replaced by other term, such as avoidance cost or feasibility if the system has not been implemented yet. An estimation of the robustness of the DAIM system is included in Appendix C, see for example Table C.1 for more details.

To find out how well a failure mode is handled, I must navigate the system, and discuss with one of those who has developed the DAIM system. He has also told me why the robustness is not a priority in the DAIM system. The reason is that the end user of the DAIM system will be students and employees at NTNU. The DAIM system is a role-based system and requires a logon for each user. If somebody try to crack the system, the administrator of the system knows who it is.

Moreover, human validating is important for the DAIM system. The administrator role must validate information and change the information if it is not up to date, such as information of a censor or a supervisor. Hence, most information that students filled in for the master contract will be checked by human. Thus, incorrect information will be stopped and changed by the administrator role. In addition, there are a lot of value checking that the developers has omitted.

By using the combination of both qualitative and quantitative assessment methods, the robustness of the DAIM system is estimated as 0.59. This is not a optimal value and not acceptable, since the value is too low.

12.1.2 Testing

AutAT is used to automatize the input robustness testing of the DAIM system. There are a lot of limitation when AutAT is used as an automatical tool for testing. The following characteristics of the AutAT restrict the automated testing:

- There are some form elements, like for instance radio button, that are not implemented.

- It's impossible to start a test from where the previous test ends. Hence, a user has to logon each time the test is running, before it can do anything else.
- The AutAT has no support for dynamical Web-pages
- It is impossible to run tests in AutAT. The tests must be export to Watir or FIT before running.

Because of the mentioned limitation, the tests must in many cases be manually performed. Moreover, a dynamical page is used as logging page for administrator role, and the password and username must be entered manually even if the tests are automated.

The Goal/Question/Metric method is defined to analysis and evaluate the robustness of the DAIM system. Several metrics are used, but only one of them is related to the input robustness. The other metrics are used to give us information of other factors that can increase or decrease the robustness of the DAIM system.

Based on collected data during testing, the robustness of the DAIM system is estimated as 0.50. The robustness of each use case can vary from 0 to 1. Luckily, the most important use cases have high focus on the robustness, i.e. UC05. Even if the robust value is not optimal, metric M2 tells us that the system has printed meaningfull and intelligible error messages. Furthermore, as recorded in the metric M3, the system responds to user actions in an intuitive and consistent way.

As a result of the metric M4, approximately half of the fields that have special syntax input will also have an explanation. Many of the fields that lacked clearly defined syntax descriptions could be considered as self-explanatory. Hence, the user will be able to determine how to enter data into the system. However, the system has not sufficiently used the syntactic input validation to avoid invalid user input and possibly contribute to increase the input robustness.

The system has omitted a lot of value validation, such as checking for integer value where it requires letters and conversely. Hence, the database contains many meaningless or invalid data since many input are saved without validating. The parameter constraints, however, are well handled by the system.

12.2 Evaluation

A combination of the Jacobson's analysis model and FMEA proposed by J. Zhou [24], and the quantitative method proposed by T. Stålhane [19] were used to assess and estimate the robustness of the DAIM system. Estimating the robustness by using these two frameworks give us a value at 0.59, while the outcome from testing is 0.50. The difference between these outcomes is 0.09. There are several causes that can influence this result.

Jacobson's analysis model, the FMEA table, the robustness failure modes and the causes are assessed at a high level. Estimating the robustness at a high level will not give us an exact value. For example, when the failure modes of user input are well handled, it can be either an illegal character or an invalid format that is well handled, or maybe both.

Furthermore, to illustrate the Jacobson's analysis model at a lower level, such as a user story or sub-use case instead of a use case will result in analysing the robustness failure modes and causes in more details. This will give us a more exact value. An example is considered in Figures 12.1 and 12.2 to see the difference between a high and a low level during analysis the Jacobson's analysis model.

Figure 12.1: An example of Jacobson analysis diagram at high level

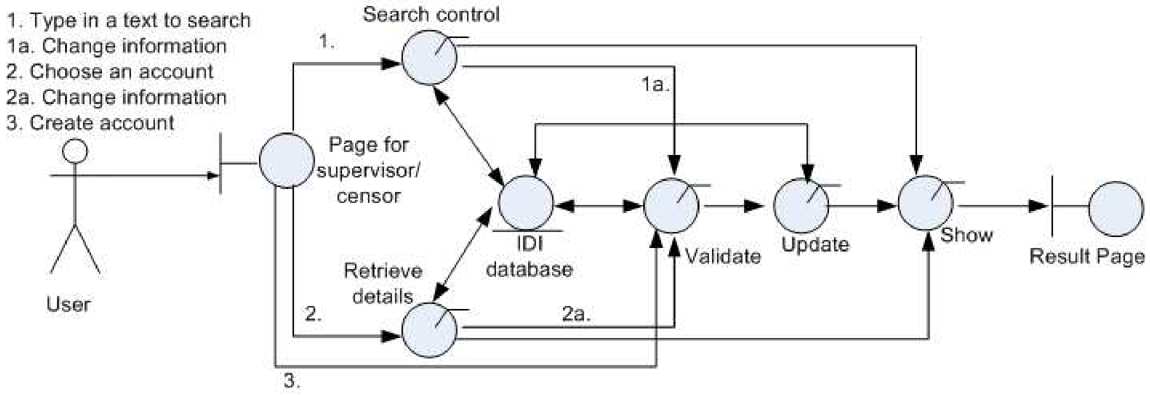
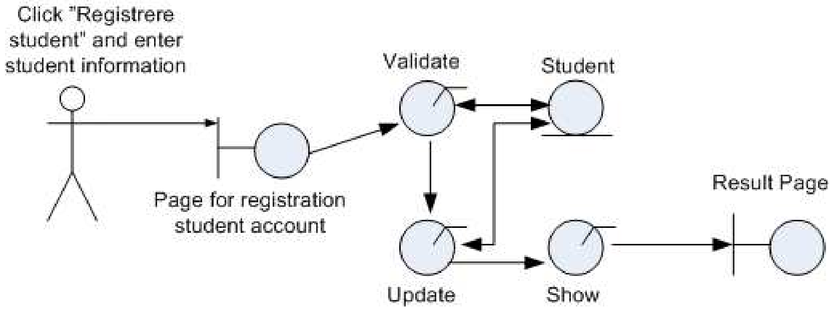


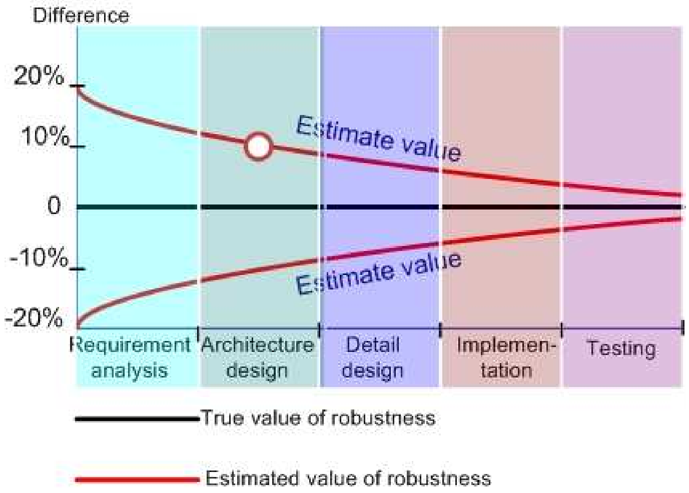
Figure 12.2: An example of Jacobson analysis diagram at low level


Click "Registrere student" and enter student information
Page for registration student account
Validate
Student
Update
Show
Result Page

An other factor that can have influence in the result of assessment methods is that the DAIM system is already implemented. Usually, the Jacobson model is used early to assess the robustness, where it still lacks much informations. In this project, the DAIM is already implemented and all required information are available. I believe that this factor give us more exact value of the robustness than if the DAIM system is not implemented yet, even if I've assessed the robustness at a high level.

Figure 12.3 illustrates the accuracy between the estimated and true value of the robustness. This figure is plotted based on what I've learned in this project. However, how exactly can the robustness of a Web-based system be estimated, is dependent on how early the robustness is assessed. The more exact value of the robustness can be estimated, the more details and information about the system are required. A little circle in Figure 12.3 is the point where the difference between the true and the estimated robustness occured in this project, approximately 0.1.

Figure 12.3: Difference between the estimated and true value of robustness



During testing, the robustness failure modes are considered in details, such as illegal character, distinguished between integer and non-integer, etc. The estimated robustness by testing give us therefore more exact value. Hence, I believe that considering and estimating the robustness at different levels influence the outcome and give us an variable in the estimation. The more detail the Jacobson's analysis model is illustrated, the better robustness value can be estimated.

Selecting the tests data can also give us a false result from testing. For example, if the same invalid input type is used to test the same functionality twice, it will not be considered as two tests, but only one. This is important because it can give us a false pattern of data. There are several factors that can influence the outcome of testing, which has described as threats to validate in Section 11.4.

To conclude, the robustness of a Web-based system can be estimated early and assessed by using the frameworks proposed by J. Zhou et al. [24] and T. Stålhane [19]. To have an exact value of the robustness, the Jacobson's analysis model must be assessed as detail as it's possible. If the Jacobson's analysis models are assessed at a high level or in early development phase, there may be an error of 10-20%. The more detail information of system we have, the more exactly robustness value is estimated.

Chapter 13

Further Work

This project could not do a larger prestudy or testing all features of the DAIM system. In this chapter includes what will be considered as important future work.

- Assess and estimate the robustness of the DAIM system by using the framework that Hong Zhu et al. [26] has proposed. The result of this framework will be compared with the result of assessment methods that has been used for this project.
- Test the robustness of the DAIM system by using several browser, i.e. Firefox. Because DAIM is intended to act as well in Firebox as in Internett Explorer 6.0
- The DAIM system is still under development. There are a lot of new function that have not been tested. It will be interesting to test all functions.
- Assess the robustness of the DAIM system by illustrating the Jacobson's analysis model in a lower level and in more detail. In this project, the Jacobson's analysis model is illustrated for each use case and a use case consist of one or several user stories. It's interesting to illustrate the Jacobson's model for each user story and check that the robustness is the same as what we've estimated.
- Use the same assessment methods and process to assess the robustness of an other Web-based system. Compare and evaluate the results of this project and the new one.

Part VI

Appendix

Appendix A

Abbreviations and Glossary

A.1 Abbreviations

WebSys	A WEB-based SYStem – A nonfunctional requirement project onTime-to-market vs. Reliability
AutAT	Automatic Acceptance Testing of Web Application
TTM	Time-to-Market
WWW	World Wide Web
UML	Unified Modeling Language
FMEA	Failure Mode and Effect Analysis
DFD	Data Flow Diagram
SFMEA	Software Failure Mode and Effect Analysis

A.2 Glossary

Here are explanation to words that might come in handy reading the report.

ACID	These four properties are considered to be key transaction processing features of database management system. Without them, the integrity of the database cannot be guaranteed. The four properties are atomicity, consistency, isolation and durability [3].
Fault (defect)	Abnormal condition that may cause a reduction in, or loss of, the capability of a functional unit to perform a required function
Failure	The inability of a system to perform its required funcitons within specified performance requirements [10].
Error	Discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition
Ripple fault	Fault in one component will be result in an another component
Abnormal software behavior	Including responses like run time exceptions, system can not do what it is intended to do or revealing confidential information to unauthorized user
Qualitative	This methodology is concerned with studying objects in their natural setting. A qualitative researcher attempts to interpret a phenomenon based on explanations that people bring to them [22]
Quantitative	This methodology is mainly concerned with quantifying a relationship or to compare two or more groups. The aim is to identify a cause-effect relationship. The quantitative research is often conducted through setting up controlled experiments or collecting data through case studies.

Appendix B

Quality Models and SFMEA

This appendix includes some quality models and SFMEAs that I've used the framework [26] to consider the DAIM system.

B.1 Choose sensor

Table B.1: SFMEA for Choose censor

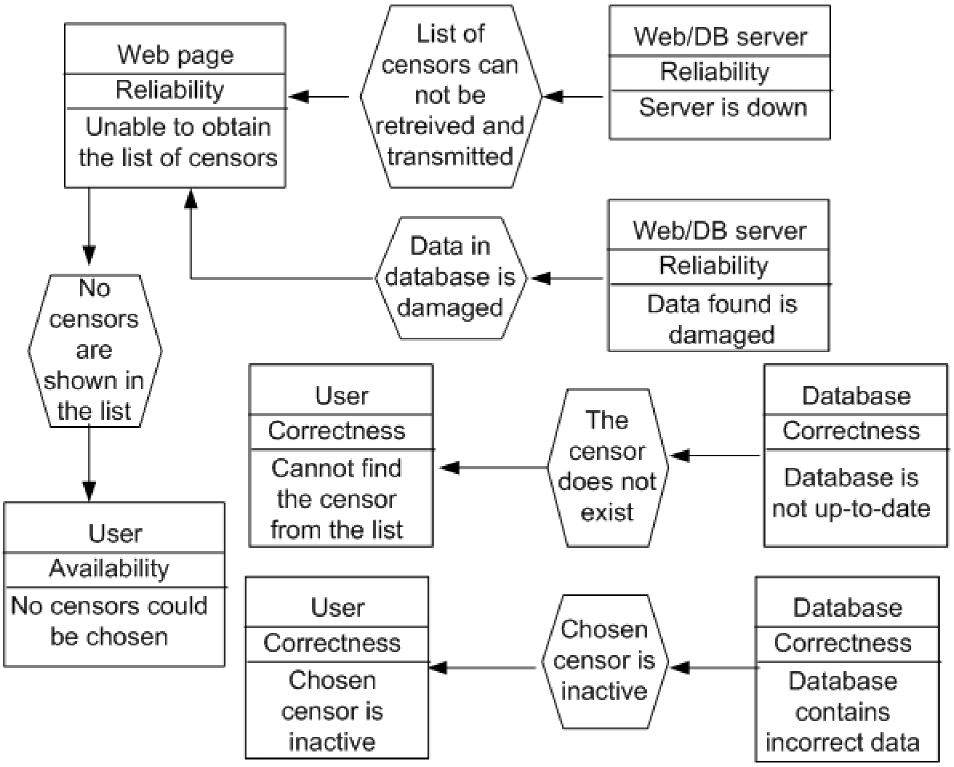
Software FMEA for Choose censor					
No.	**Failure modes**		**Possible Cause**		**Explanation**
No.	**Component**	**Phenomena**	**Component**	**Fault/Failure mode**	**Explanation**
1	The user	No censors could be chosen	Web page	Unable to obtain the list of censors	When the user chooses a censor for a specific master thesis, no censors are shown in the list
2		Cannot find the censor from the list	Database	The censor doesn't exist in database	Database is not up to date
3		Chosen censor is inactive	Database	Information found is incorrect	The database contains incorrect information

Continued on next page

Table B.1 – continued from previous page

No.	Failure modes		Possible Cause		Explanation
No.	**Component**	**Phenomena**	**Component**	**Fault/Failure mode**	**Explanation**
4	Web page	Unable to obtain the list of censors	Web server and/or database server	Web server and/or database server are down	The list of censors can not be retrieved and transmitted
5			Database	Data is damaged	Data in database is damaged and couldn't be shown by Web page

Figure B.1: The quality model for Choose censor

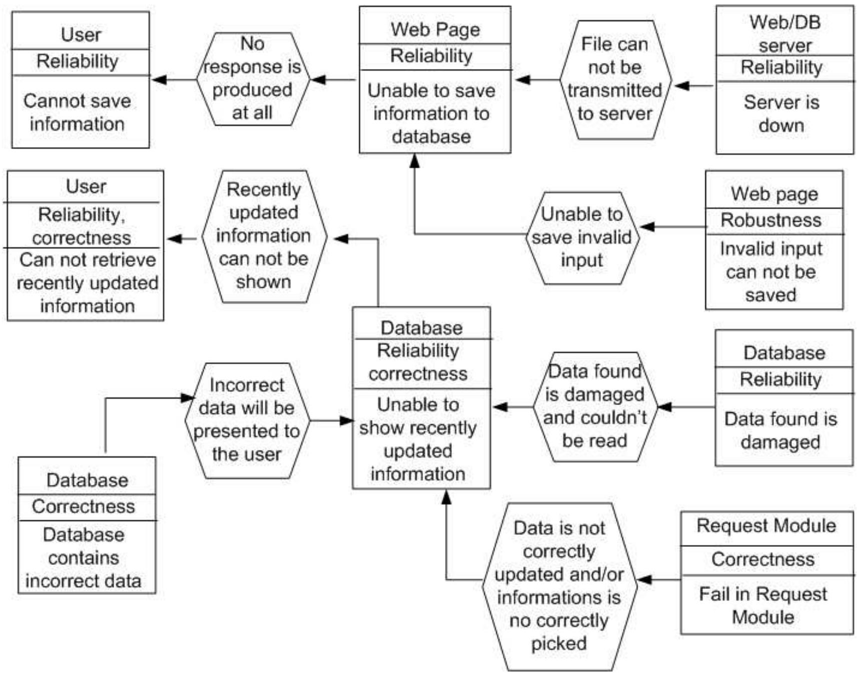

Web page
Reliability
Unable to obtain the list of censors
List of censors can not be retreived and transmitted
Web/DB server
Reliability
Server is down
Data in database is damaged
Web/DB server
Reliability
Data found is damaged
No censors are shown in the list
User
Availability
No censors could be chosen
User
Correctness
Cannot find the censor from the list
The censor does not exist
Database
Correctness
Database is not up-to-date
User
Correctness
Chosen censor is inactive
Chosen censor is inactive
Database
Correctness
Database contains incorrect data

B.2 Validate information

Table B.2: SFMEA for Validate information

Software FMEA for Validate informationr					
No.	**Failure modes**		**Possible Cause**		**Explanation**
No.	**Component**	**Phenomena**	**Component**	**Fault/Failure mode**	**Explanation**
1	User	Cannot save information	Web page	Unable to save information to database	No response is produced at all when user saves information
2		Cannot retrieve updated information	Database / Request Module	Unable to show recently updated information	Updated information is saved, but can not retrieve the updated information
3	Web page	Unable to save information to database	Web server /database server	Server is down	The file cannot be transmitted to server
4			Web page	Invalid input cannot be saved to database	Unable to save invalid input to database
5	Database	Unable to show recently updated information	Database	Data found in database is damaged	Data found is damaged and couldn't be read
6			Database	Database contains incorrect data	Incorrect data is presented to the user
7			Request module to database	Fail in Request module	Data is not correctly updated and/or incorrect data is picked

Figure B.2: The quality model for Validate information


User
Reliability
Cannot save information
No response is produced at all
Web Page
Reliability
Unable to save information to database
File can not be transmitted to server
Web/DB server
Reliability
Server is down
User
Reliability, correctness
Can not retrieve recently updated information
Recently updated information can not be shown
Unable to save invalid input
Web page
Robustness
Invalid input can not be saved
Database
Reliability correctness
Unable to show recently updated information
Incorrect data will be presented to the user
Data found is damaged and couldn't be read
Database
Reliability
Data found is damaged
Database
Correctness
Database contains incorrect data
Data is not correctly updated and/or informations is no correctly picked
Request Module
Correctness
Fail in Request Module

B.3 Get information

Table B.3: SFMEA for Get information

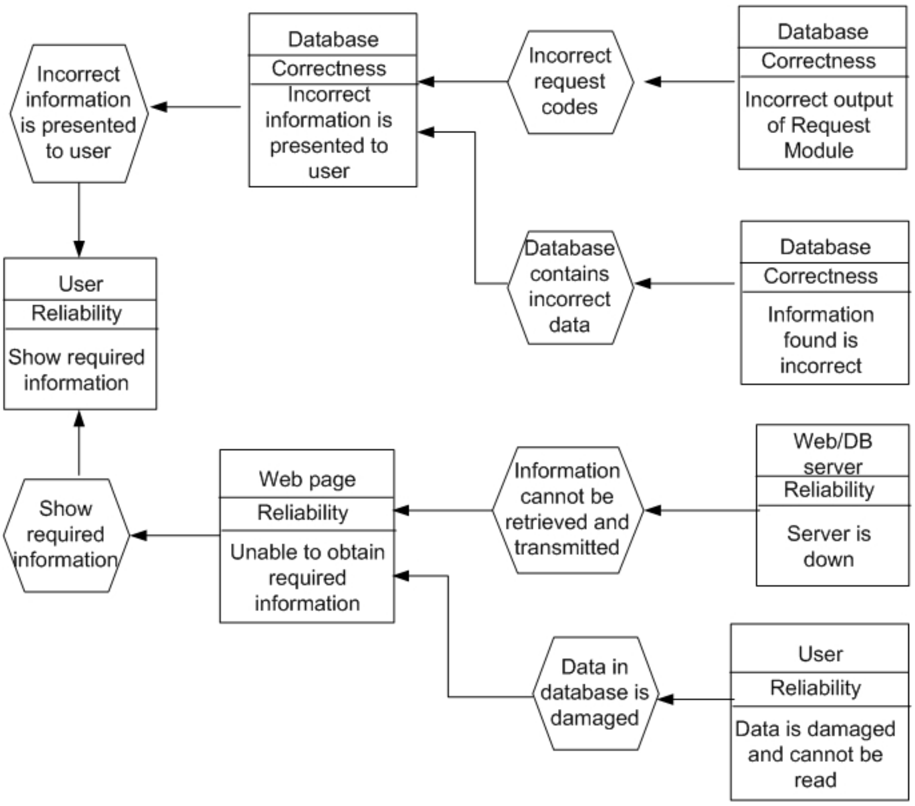
Software FMEA for Get informtaion					
No.	**Failure modes**		**Possible Cause**		**Explanation**
No.	**Component**	**Phenomena**	**Component**	**Fault/Failure mode**	**Explanation**
1	The user	Show required information	Web page	Unable to obtain required information	When the user wants the system to show required information
2			Database	Incorrect information is shown	Incorrect information is presented to user
3	Web page	Unable to obtain required information	Web server / Database server	Server is down	The file cannot be retrieved and transmitted

Continued on next page

Table B.3 – continued from previous page

No.	Failure modes		Possible Cause		Explanation
No.	Component	Phenomena	Component	Fault/Failure mode	Explanation
4			Database	Data in database is damaged	Data is damaged and cannot be read
5	Database	Incorrect information is shown	Request Module	Incorrect output of Request Module results in incorrect information	The request code in Request Module picks incorrect information
6			Database	Information found is incorrect	Database contains incorrect data

Figure B.3: The quality model for Get information

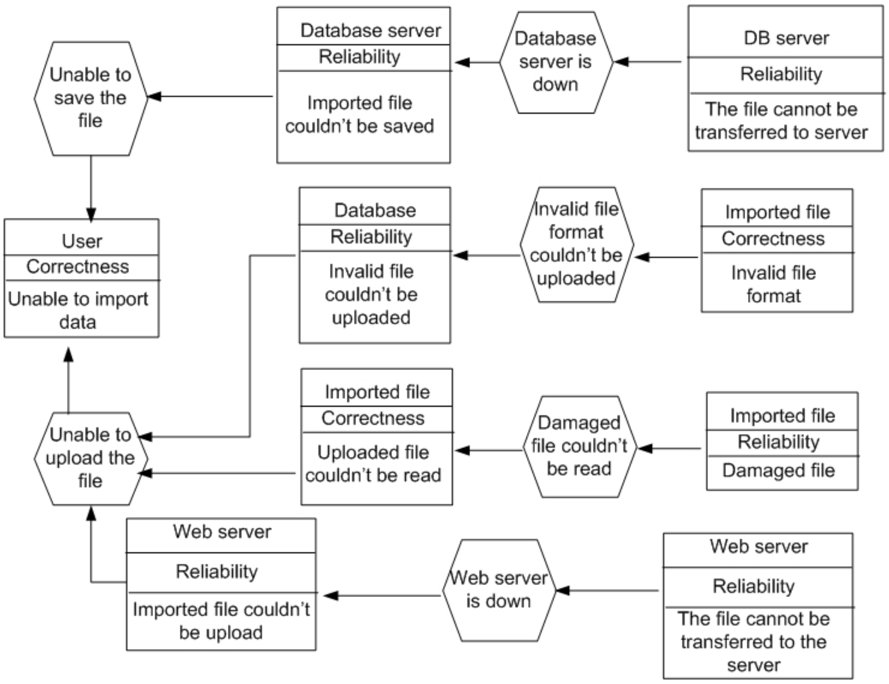

Incorrect information is presented to user
Database
Correctness
Incorrect information is presented to user
Incorrect request codes
Database
Correctness
Incorrect output of Request Module
User
Reliability
Show required information
Database contains incorrect data
Database
Correctness
Information found is incorrect
Show required information
Web page
Reliability
Unable to obtain required information
Information cannot be retrieved and transmitted
Web/DB server
Reliability
Server is down
Data in database is damaged
User
Reliability
Data is damaged and cannot be read

B.4 Import data

Table B.4: SFMEA for Import data

Software FMEA for Import data					
No.	**Failure modes**		**Possible Cause**		**Explanation**
No.	**Component**	**Phenomena**	**Component**	**Fault/Failure mode**	**Explanation**
1	The user	Unable to import data	Web page	Unable to save the file to database	No response is produced at all when the user imports data
2			The imported file/Web page	Unable to upload the file	No response is produced at all when the user imports data
3	Web page	Unable to save the file	Database server	Database server is down	The file cannot be saved in database server
4		Unable to upload the file	Web server	Webserver is down	Server is down and the file couldn't be uploaded
5	The imported file	Unable to upload the file	Imported file	Invalid file format	The file contains invalid character or is invalid format. Thus, the file couldn't be uploaded
6			Imported file	Imported file is damaged	Imported file is damaged and couldn't be read

Figure B.4: The quality model for Import data



Appendix C

Qualitative Robustness Assessment

This appendix contains an estimation of the robustness of the DAIM system, by using the principle described by T. Stålhane from [19], with the proposed changes presented in the section 9.3.

Table C.1: The FMEA for Log in

Use case id: UC01					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Login/ Control	Incorrect username and password are typed, but not handled	5	5	Retype username and password	No response is produced
	Data found is incorrect/damaged	5	4	User cannot log in with correct username and password	No response is produced
Show result	Error output from "Login/-Control" but not handled	3	2	User cannot log in	Error from "Login/Control" module, no response is produced
Use case sums: UC01		13	11		
Robustness for the use case Log in: 0.85					

The robustness for the use case "Log in" is as follows:

$$Robustness = \frac{Mitigation}{Seriousness} = \frac{11}{13} = 0.85 \tag{C.1}$$

Table C.2: The FMEA for Fill in contract

Use case id: UC02					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Validate	Invalid input is received, but not handled	5	2	System crash or invalid inputs are saved in database	No response is produced. Invalid inputs are saved.
Update	Error output of "Validate" received, but not handled	5	2	System crash or invalid inputs are saved in database	No response is produced. Fail in "Validate information" module
	Database contains incorrect data/error user input, but not handled	5	3	Incorrect data is presented to user	Information found is incorrect
Show	Error output of "Updated" module, but not handled	3	2	Retype information	No response is produced
	Output of "Update" contains incorrect data, but not handled	5	2	Incorrect data is presented to the user	Database contains incorrect data
Use case sums: UC02		23	11		
Robustness for the use case Fill in contract: 0.48					

Table C.3: The FMEA for Create cooperation group

Use case id: UC03					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Validate	Invalid input is received, but not handled	5	4	System crash or invalid inputs are saved	No response is produced
	IDI database contains incorrect data, but not handled	5	3	Incorrect data is presented to the user	Information found is incorrect
Update	Error output of "Validate" received, but not handled	5	5	System crash, or invalid input will be saved in database	No response is produced. Fail in "Validate information" module
	Database contains incorrect data/ error user input, but not handled	5	3	Incorrect data is presented to user	Information found is incorrect
Show	Error output of "Updated" module, but not handled	3	3	Retype information	No response is produced
	Output of "Update" contains incorrect data	5	4	Incorrect data is presented to the user	Database contains incorrect data
Use case sums: UC03		28	22		
Robustness for the use case Create cooperation group: 0.78					

Table C.4: The FMEA for Generate contract/schema

Use case id: UC04					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Get information	The used browser cause error user input, but not handled	3	3	The user has to generate the contract/schema again by using another browser	No response is produced
	IDI database contains incorrect data, but not handled	5	3	Incorrect data will be shown to the user	Information found is incorrect
Generate	Error output from "Get information", but not handled	3	3	Fail to respond to user's interaction	
Show	Error output of "Generate", but not handled	3	3	Try to generate contract/schema again	No response is produced
	Generated contract contains incorrect information	5	4	Incorrect information is presented to the user	Database contains incorrect data
Use case sums: UC04		19	16		
Robustness for the use case Generate contract/schema: 0.84					

Table C.5: The FMEA for Deliver master thesis

Use case id: UC05					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Upload files	Invalid file format, or the file contains invalid data, but not handled	5	4	System crash and must reupload the file. Invalid file will be saved	Time out exception. No response is produced. Invalid file is saved
Update	Error output of "Upload" and/or "Confirm", but not handled	5	3	Reupload the file	
Confirm	Error output from "Upload", "Update" and "Communicate"	4	3	Uploaded file couldn't be saved in database	
Communicate	Error output from "Confirm" module	4	3	The user can not perform the delivery process	
Show	Error output of "Confirm"	4	4	Fail to respond to user's interaction	No response is produced.
Use case sums: UC05		22	17		
Robustness for the use case Deliver master thesis: 0.77					

Table C.6: The FMEA for Search for master thesis

Use case id: UC06					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Search Control	Illegal/ unreasonable character received, but not handled	5	0	System performed the search with invalid inputs. Retype the text to search	No response is produced and/or uninteresting results will be shown
	Database contains incorrect data, but not handled	5	3	Incorrect data is presented to the user	Data found is incorrect
Show	Error output of "Search Control"	5	0	Retype the text to search	No response is produced at all
	Output of the "Search Control" contains incorrect data	3	1	Incorrect data is presented to the user	IDI database contains incorrect data
Use case sums: UC06		18	4		
Robustness for the use case Search master thesis: 0.22					

Table C.7: The FMEA for Handle payments

Use case id: UC07					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Get information	The user browser causes error user input, but not handled	3	3	Fail to respond to user's interaction	No response is produced

Continued on next page

Table C.7 – continued from previous page

Control Object	Failure mode description	Seriousness	Mitigation	Robustness consequences	Indicators
	Error output of "Get information" and/or IDI database contains incorrect data, but not handled	5	3	Incorrect data is presented to the user	Information found is incorrect
Change status	Error output of "Get information", but not handled	3	3	Changed information will not be updated	
Update	Error output of "Change status"	3	3	Changed information will not be updated	
	Output of "Change status" contains incorrect data, but not handled	5	4	Incorrect data is presented to the user	Database contains incorrect data
Sort	Error output of "Update", "Get information", but not handled	4	4	Fail to respond to user's interaction	
Show	Error output of "Sort", but not handled	4	4	Fail to respond to user's interaction	
	Output of "Sort" contains incorrect data, but not handled	5	4	Incorrect information will be shown to the user	Incorrect data is displayed
Use case sums: UC07		32	28		

Continued on next page

Table C.7 – continued from previous page

Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Robustness for the use case Handle payment: 0.88					

Table C.8: The FMEA for Display master thesis

Use case id: UC08					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Search Control	Illegal/ unreasonable character received, but not handled	5	3	The search performed with invalid character. Retype the text to search	No response is produced and/or uninteresting results will be shown
	Database contains incorrect data, but not handled	5	3	Incorrect data is presented to the user	Data found is incorrect
Show	Error output of "Search Control"	3	2	Retype the text to search	No response is produced at all
	Output of the "Search Control" contains incorrect data	5	2	Incorrect data is presented to the user	IDI database contains incorrect data
Use case sums: UC08		18	10		
Robustness for the use case Display master thesis: 0.55					

Table C.9: The FMEA for Choose censor

Use case id: UC09					
Failure mode	**Failure mode description**	**Seriou-sness**	**Mitig-ation**	**Robustness conse-quences**	**Indicators**
Update	The user browser cause error user input	5	5	Try to choose the censor again by using another browser	
	Database contains incorrect data	5	3	Incorrect information is displayed to the user	Information found is incorrect
Show	Error output of "Update"	5	5	Try to choose the censor again	No response is produced
	Output of "Update" contains incorrect information	5	4	Incorrect information is displayed	Database contains incorrect information
Use case sums: UC09		20	17		
Robustness for the use case Choose censor: 0.85					

Table C.10: The FMEA for Validate information

Use case id: UC10					
Failure mode	**Failure mode description**	**Seriou-sness**	**Mitig-ation**	**Robustness conse-quences**	**Indicators**
Retrieve details	Data is damaged and couldn't be retrieved, but not handled	5	4	Retype damaged information.	No response is produced

Continued on next page

Table C.10 – continued from previous page

Failure mode	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
	Information found is incorrect	5	3	Incorrect information is displayed	Database contains incorrect data
Validate changes	Error user input, i.e. illegal date format, character, but not handled	5	2	System crash. Invalid inputs are saved	
Update	Error output of "Validate changes", but not handled	5	2	Changed information will not be updated to database	
Show	Error output of "Retrieve details", "Update"	4	2	Fail to respond to user's interaction	No response is produced
	The output of "Retrieve details" contains incorrect data, but not handled	5	2	Incorrect information is displayed to the user	Database contains incorrect data
Use case sums: UC10		29	15		
Robustness for the use case Validate information: 0.52					

Table C.11: The FMEA for Create/update account

Use case id: UC11					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Search Control	Error user input, i.e. illegal character to search, but not handled	5	0	Retype text to search	No response is produced
Retrieve details	Error user input/Database contains incorrect data	5	3	Incorrect data is presented to the user	Information found is incorrect
Validate changes	Error output of "Search Control", "Retrieve details", error user input	5	2	System crash and/or invalid inputs are saved	
Update	Error output of "Validate changes"	5	1	Changed information will not be updated	
Show	Error output of "Update", "Retrieve details", "Search control"	4	1	Fail to respond to user's interaction	No response is produced
	Output of "Update", "Retrieve details" and "Search control" contains incorrect data	5	2	Incorrect data is presented to the user	Found information is incorrect
Use case sums: UC11		29	9		
Robustness for the use case Create/update account: 0.31					

Table C.12: The FMEA for Get information

Use case id: UC12					
Failure mode	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Get information	The browser causes error user input, but not handled	5	5	Choose information to display again by using another browser	No response is produced
	IDI database contains incorrect data, but not handled	5	3	Incorrect data is presented to the user	Information found is incorrect
Show	Error output of "Get information"	4	4	Fail to respond to user's interaction	No respond is produced
	Output of "Get information" contains incorrect data	4	4	Incorrect data is presented to the user	Database contains incorrect data
Use case sums: UC12		18	16		
Robustness for the use case Get information: 0.89					

Table C.13: The FMEA for Import data

Use case id: UC13					
Failure mode	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Validate file	Error user input, imported file is damaged or invalid format, but not handled	5	4	System crash, and/or invalid file is saved in the database	
Upload	Error output of "Validate file", but not handled	3	3	System crash and/or invalid file will be updaded to the system's database	
Update	Error output of "Upload", but not handled	4	4	System crash and/or invalid file format will be updated to the system's database	
Show	Error output of "Update", but not handled	3	3	System crash and/or invalid file format will be updated to the system's database	No response is produced
Use case sums: UC13		15	14		
Robustness for the use case Import data: 0.93					

Table C.14: The FMEA for Change information

Use case id: UC14					
Control Object	**Failure mode description**	**Seriousness**	**Mitigation**	**Robustness consequences**	**Indicators**
Retrieve details	Illegal/ Unreasonable character received, but not handled	5	0	Invalid inputs are saved in database	
	Error user input/IDI database contains incorrect data	5	3	Incorrect data is presented to the user	Information found is incorrect
Validate	Invalid user input are not validated	5	0	Invalid inputs will be saved in database	
Update	Error output of "Validate", but not handled	5	1	Abnormal behavior or invalid inputs are saved	
Show	Error output of "Update", "Retrieve details"	5	1	Fail to respond to user's interaction	No response is produced
	Output of "Retrieve details" contains incorrect data	5	2	Incorrect data is presented to the user	Database contains incorrect data
Use case sums: UC14		30	7		
Robustness for the use case Change information: 0.23					

Appendix D

Tests Data

This appendix presents input data that I've used to test the DAIM system. The input data is related to the use cases and metrics.

Usually, each invalid input is considered as a test and each row can consist of several tests. When a test is intended to check the logic and need several inputs, it's considered as one test, and will be marked by **[L]** in the first column in the row.

Username and password for all administrator roles are saved in a file AdministratorPassword.txt, which included in attachment.

Table D.1: Test data for UC01 - Log in

UC01: Log in		Date: 24.04.06
Precondition	The inputs data are invalid usernames and passwords.	
Field: Input Data	Expected Reaction	Reaction
[L] Brukernavn: thuyhuet Passord: abcde3	System prints a message to tell the user that username and password are incorrect	As expected
[L] Brukernavn: enTest Passord: 12345	System prints a message to tell the user that username and password are incorrect	As expected
Total tests: 2		Tests passed: 2

There are several required information fields to be used when the master contract is filled in. The information is splitted in four categories, based on the Web Page that displays this information: student, date and deadline, bibliographic data, and supervisors. Tables D.2 to D.5 presents the input data for UC02.

Table D.2: Test data for UC02: Students

UC02: Fill in contract		Date: 24.04.06
Precondition	For all categories in UC02. There is a student account in "IDI database" which has not an approved master contract.	
Field: Input Data	Expected Reaction	Reaction
Fornavn: 23432 Etternavn: 3434343	First name and last name can not include only integers. Expected reaction as type 1	System's response was as type 3
Fødselsdato: 30.13.2006, dfadfaf	The date formats are invalid. Expected reaction as type 1	As expected
[L] Fødselsdato: 31.12.2007	The date format 30.12.2007 is a valid format, but it's not logical date, since the year 2007. Expected reaction as type 1.	System didn't recognize 31.12.2007 as invalid input. System's reaction was as type 3
Merknad: 1234232	The comment field could include none or a combination of letters and digits, but cannot include digits only. Since digits give no meaning. As type 1.	System react as type 3.
Telefon: dfadafda	The telephone format can include letters and digits, but not letters only. Expected reaction as type 1.	System react as type 3
E-post, privat: hue pham	System could have ability to recognize a completely e-mail adress. Expected reaction as type 1.	As type 3
[L] Fakultet: Fakultet for Informasjonsteknologi Field of study: Intelligente systemer	Missing/ incorrect chosen data are recognized and prompt to the user	As expected
Total tests: 9		Tests passed: 3

Table D.3: Test data for UC02: Startdate and deadline

UC02: Fill in contract		Date: 24.04.06
Field: Input Data	Expected Reaction	Reaction
Oppstartdato: 13.13.2006, dafadfad	Date format is invalid, and expected reaction as type 1.	Invalid inputs are saved as 00.00.0000 and no feedback is printed. System react as type 2
Oppstartdato: 20.12.2015, 30.01.1970	System cannot support a start date which has passed or will be pass about 9 years. Intelligible feedback is printed to the user. Expected reaction as type 1.	System doesn't validate that type of checking. System react as type 3.
Total tests: 4		Tests passed: 0

Table D.4: Test data for UC02: Bibliographic data

UC02: Fill in contract		Date: 24.04.06
Field: Input Data	Expected Reaction	Reaction
Språk: Norsk (bokmål) Utført ved: 12345 Hovedtittel: 54321 Undertittel: 12345 Title: 54321 Subtitle: 12345 Oppgavetekst på norsk: 54321	These types of input can not include integers only. Expected as type 1	As type 3
[L] Språk: Norsk (bokmål) Utført ved: en test Hovedtittel: en test	Missing inputs are detected. System prompts the user	As expected reaction
Total tests: 8		Tests passed: 6

Information of the supervisors are saved in database and will be shown by a selected box. A user can only type information of the secondary supervisors in UC02. Input data of the supervisor is add and tested in UC11. Only information of secondary supervisor will be test.

Table D.5: Test data for UC02: Supervisor and Secondary Supervisor

UC02: Fill in contract		Date: 25.04.06
Field: Input Data	Expected Reaction	Reaction
Fornavn: 98765 Etternavn: 56789 Veiledertype: Ekstern veileder Institusjon: 45678	These types of input can not include integers only. Expected as type 1	The type of supervisor has to be filled in by choosing from a select box. For other inputs, the system respond as type 3.
Total tests: 4		Tests passed: 1

Table D.6: Test data for UC03 - Create cooperation group

UC03: Create cooperation group		Date: 25.04.06
Precondition	Input data contain invalid usernames for cooperation	
Field: Input Data	Expected Reaction	Reaction
Brukernavn: thuyhuet and 123456	System validates usernames/-cooperation group are valid or not and give feedback to the user, expected as type 1	As expected
Total tests: 2		Tests passed: 2

A student will generate contract/schema by clicking the button "Generate". Information will be generated as a pdf file and shown to the user. No user inputs are required. The "generate" button is available only when the requirements for generating are satisfied, i.e. enough information, cooperation are approved if it's generating a cooperation contract.

Table D.7: Test data for UC04 - Generate contract/schema

UC04: Generate contract/schema		Date: 25.04.06
Precondition	To generate master contract: All required information of master contract are filled in. To generate cooperation contract: All students in cooperation are approved. To generate delivered schema: The student's master thesis is delivered	
Actions	Expected Reaction	Reaction
Generate the master contract, the contract for cooperation, delivery schema	Relevant information is fetched and generated by system.	As expected. The feedback is unnecessary.
Total tests: 3		Tests passed: 3

The AutAT tool does not support testing of a dynamic Web page and the uploaded file can not be browsed automatic. Thus, the test of this use case is performed manually. The use case "UC05: Delviery master thesis" consists of several user stories which captured in Tables D.8 to D.11.

Table D.8: Test data for UC05-01 - Upload a picture of front-page

UC05-01: Upload a picture of front-page		Date: 26.04.06
Precondition	For all user stories in UC05. The student's master contract has been approved by administrator role and has the status "Writing the master thesis". Do following actions: 1) Go to http://newdaim.idi.ntnu.no/login.php 2) Log in with "evatest" as username and "Makoba2050" as password 3)Click the button or link with the label "Min masteroppgave" 4) Click the button "Endre5"	
Actions	Expected Reaction	Reaction
5) Click "Browse" and choose the file TestFile.doc. Then click "Last opp"	A doc file is invalid format. Expected as type 1	As expected
6) Click "Browse" and chose a picture that has less than 600x600 pixels resolution, i.e. meg.jpg. Then click "Last opp"	The file format has low resolution. Expected as type 1	As expected
7) Click "Browse" and choose a picture that is not square, i.e. sunset.jpg. Then click "Last opp"	The picture is not square. Expected as type 1	System recognized the picture has 800x600 pixels resolution. But the picture was saved without prompting.
8) Click "Browse" and choose a picture that larger than 2MB, i.e. picTooLarge.jpg. Then click "Last opp"	The size of the picture is too large. Expected as type 1	System recognized the size of the picture is too large. But the picture was saved without prompting
Total tests: 4		Tests passed: 2

Table D.9: Test data for UC05-02 - Upload an attachment as a zip file

UC05-02 - Upload an attachment as a zip file		Date: 24.04.06
Precondition	As described in UC05-01. The sequence number 4) is replaced with 4) Click the button "Endre...6"	
Actions	Expected Reaction	Reaction
5) Click "Browse" and choose the file TestFile.doc. Then click "Last opp"	Invalid file format is detected. System prompts the user. The expected reaction is as type 1	As expected
6) Click "Browse" and choose the file emptyZipFile.zip. Then click "Last opp"	This is an empty zip file. System recognizes the file is empty. Expected as type 1	As expected
Total tests: 2		Tests passed: 2

Table D.10: Test data for UC05-03 - Upload the report in pdf format

UC05-03 - Upload a report in pdf format		Date: 24.04.06
Precondition	As described in UC05-01. The sequence number 4) is replaced with 4)Click the button "Start innlevering av masteroppgaven" 5) Click "Neste"	
Field: Input Data/ Actions	Expected Reaction	Reaction
6) Click "Browse" and choose the file TestFile.doc. Click "Last opp"	Invalid file format is detected. Expected as type 1	As expected

Continued on next page

Table D.10 – continued from previous page

Field: Input Data/Actions	Expected Reaction	Reaction
[L] 7) Click "Browse", choose the file aTest.pdf. Click "Last opp". 8) Fill <50> in the field "Totalt antall sider", and <15,56> in the field "Sidetall for fargesider". Click "Neste"	System recognizes the colour page is higher than total sides and prompts the user.	As expected
9) Fill in <xx> in the fields "Totalt antall sider" and "Sidetall for fargesider". Click "Neste"	The inputs are invalid and system responds as type 1.	As expected
10) Fill in <5000> in the field "Totalt antall sider". Click "Neste"	The total number of pages is too high and system prompts the user.	System saves input data without prompting the user
11) Fill in <10> in the field "Totalt antall sider". Click "Neste"	The total number of pages is too low and system prompts the user.	As expected
12)Let the fields "Totalt antall sider" and "Sidetall for fargesider" be empty and click "Neste"	System recognizes that input fields are empty and prompts the user.	As expected
13) Fill in <123345> in the field "Merknader til institutt/-fakultet/trykkeri" and click "Neste"	The comment field could be empty, a combination of letters and integers, or letters only, but not only integers. System recognizes that the comments included only integers and prompts the user.	System does not validate that type of checking. Reaction as type 3.
Total tests: 9		Tests passed: 7

Table D.11: Test data for UC05-04 - Order extra copy

UC05-04 - Order extra copy		Date: 24.04.06
Precondition	This user story has the same precondition as in UC05-03. 6) Click "Browse" and choose the file aTest.pdf . Click "Last opp" 7) Fill <50> in the field "Totalt antall sider", and <15,40> in the field "Sidetall for fargesider". Then click "Ekstra bestilling".	
Field: Input Data/ Actions	Expected Reaction	Reaction
8) Fill in <100> in the field "Antall ekstra kopier". Click "Neste"	The number of extra copies will not be too high. System prompts the user	As expected
9) Fill in <xx> in the field "Antall ekstra kopier" and click "Neste"	The number of extra copies can not be letters. System prompts the user	As expected
Total tests: 2		Tests passed: 2

UC06: Search for master thesis will test the search module in DAIM system. The search module consists of a basic search and an advanced search. Test data for basic and advanced search are captured in the Tables D.12 and D.13.

Table D.12: Test data for UC06 - Basic search

UC06: Basic search		Date: 19.05.06
Precondition	For both basic search and advanced search, the IDI database must contain several delivered reports.	
Field: Input Data/ Actions	Expected Reaction	Reaction
Click "Søk" button without writing a text to search	No results will be found and displayed	System found 146 master thesis and results are displayed. The system respond as type 4
Total tests: 1		Tests passed: 0

Table D.13: Test data for UC06 - Advanced search

UC06: Advanced search		Date: 19.05.06
Field: Input Data	Expected Reaction	Reaction
med alle ordene: 12345 med den nøyaktige setningen: 54321 med noen av ordene: 12345 uten ordene: 54321 Veileder for masteroppgaver: 12345	A word must include letter(s), or a combination of letter(s) and digit(s). System could recognize there are only integers are filled in and prompt the user. Expected as type 1	System has performed the search without prompting the user
med den nøyaktige setningen: OnlyOneWord	System could recoginze that only one word is filled in, not a sentence or at least two words and prompt the user	System found 146 master thesis. But none of them included the world "OnlyOneWord". The search module doesn't work if only one word is filled in the text field for a sentence. System's reaction was as type 4

Continued on next page

Table D.13 – continued from previous page

Field: Input Data	Expected Reaction	Reaction
Click the button "Søk" without filling a text to search	No text can be used to search. The system prompts the user or performs the search and the result found is zero	System found 146 master thesis. The system respond as type 4
Total tests: 7		Tests passed: 0

The economy role in the use case "UC07: Handle payments" can only change the status of censoring by choosing "Paid" or "Not Paid" in a check box. The user can not type in input data and there are no user inputs to test. One disadvantage is that the test database contains little data or none data which can be used to test the UC07.

Table D.14: Test data for UC07 - Handle payments

UC07: Handle payments		Date: 24.04.06
Precondition	The system has an account as economy role. 1) Go to http://newdaim.idi.ntnu.no/admin. 2) Log in as administrator 3) Choose the role "Økonomi" in the select box "Rolle"	
Field: Input Data/ Actions	Expected Reaction	Reaction
4) Click the buttons "Mangler honorering", "Betalte honorar"	Registered informations are displayed to the user. Give feedback if there are nothing to display	As expected
Total tests: 2		Tests passed: 2

Several input parameters of the use case "08: Display master thesis" can be filled in by choosing radio buttons, check boxes and select boxes. The input data are defined by the DAIM system in advance and the tests for the user stories UC08-01 and UC08-02 must perform manually, and are captured in Table D.15. Only the search function with the use case id "UC08-03" requires inputs data and tests data are identified in Table D.16. UC08-03 is automated.

Table D.15: Test data for UC08-01 and UC08-02 Display and sort

UC08-01 and UC08-02 Display and sort		Date: 24.04.06
Precondition	For all user stories in UC08. The IDI database must contain a administator account and several registered master thesis. Do following actions: 1) Go to http://newdaim.idi.ntnu.no/admin 2) Log in as an administrator 3) Choose the role as "Admin" in the role select box 4) Click the button with label "Masteroppgaver" and do following actions as described below:	
Field: Input Data/ Actions	Expected Reaction	Reaction
5) Choose the radio button "Alle" in both "Vis status" and "Vis studietyper" areas	Relevant master thesis are displayed	As Expected
6) In the area "Vis valgte felter", set "Tittel" check box to false	The title parameter will not be displayed in the list of master thesis.	As Expected
[L] 7) In the area "Vis valgte felter", set "Tittel", "Veileder", "Kandidat" check boxes to false	The default parameters will be displayed in the list of master thesis.	As Expected
8) In the area "Sorter etter", select to sort by "Tittel", "Veileder", "Kandidat"	Displayed master thesis are sorted by title, supervisor and candidat	As Expected
Total tests: 7		Tests passed: 7

Table D.16: Test data for UC08-03 - Display master thesis by searching

UC08-03: Display master thesis by searching		Date: 26.04.06
Field: Input Data	Expected Reaction	Reaction
Kandidat: 12345 Veileder: 54321 Tittel: 12345	None of these parameters can include only integers. The expected reaction is as type 1.	System performs searching with invalid input and the results are shown.
Start Fra: enTest Start Til: enTest	System recognizes that the date formats are invalid and prompts the user	As expected
Start Fra: 30.07.2008 Start Til: 30.07.2009	The date after current date will be detected by system and not allowed. System prompts the user.	System has performed the search with invalid input data and no master thesis is found.
[L] Start Fra: 25.07.2006 Start Til: 25.07.2004	To search master thesis from 25.07.2006 to 25.07.2004 is invalid. System detects invalid input and prompts the user.	System has performed the search with invalid input dates and no master thesis is found.
Total tests: 8		Tests passed: 2

First, a user has to log in as administrator, supervisor or system administrator. To choose a censor for a specific master thesis, the master thesis must be delivered. The user chooses a censor for the selected master thesis and then click "Lagre" button. Information of the censor must be saved in the system's database in advance.

Table D.17: Test data for UC09 - Choose a censor

UC09: Choose a censor		Date: 04.05.06
Precondition	The master thesis has the status "delivered", and the censor is not chosen. Do following actions: 1) Go to http://newdaim.idi.ntnu.no/admin 2) Log in as an administrator 3) Choose the role as "Admin" in the role select box 4) Click the button with label "Masteroppgaver" 5) Choose the button "Alle" in the area "Vis status" and do following actions as described below:	
Field: Input Data/ Actions	Expected Reaction	Reaction
6) Choose a delivered master thesis from the list that has the status "Ferdig, men karakter ikke satt", i.e. the master thesis of the candidate "Einstein, Mari"	Detail informations of the master thesis are displayed	As expected
7) Choose a censor for this master thesis and click "Lagre" button	Updated information are saved	As expected
Total tests: 2		Tests passed: 2

Table D.18: Test data for UC10 - Validate information

UC10: Validate information		Date: 02.05.06
Precondition	For all user stories in the UC10, the IDI database must contain several master thesis with different status. Do following actions: 1) Go to http://newdaim.idi.ntnu.no/admin 2) Log in as an administrator 3) Choose the role as "Admin" in the role select box 4) Click the button with label "Masteroppgaver" 5) Choose the button "Alle" in the area "Vis status" 6) Choose a delivered master thesis from the list that has the status "Ferdig, men karakter ikke satt", i.e. the master thesis of the candidate "Einstein, Mari" 7) Choose a censor from the list of censors and and fill in following <data> in the "fields"	
Actions	Expected Reaction	Reaction
8) "Utsettelse til": <letterFormat>	The date format of delay is invalid. The expected reaction is as type 1	As expected
9) "Merknader" of the details for master thesis and the students : <12345>	The comments text area in details for master thesis and students fields include digits only. The data inputs are meaningless and considered as invalid. Expected react as type 1	System didn't check that type of invalid input. System respond as type 3
10) "Sendt til sensur": <24.07.2011>. Click "Lagre endringer". Choose an other kandidate, go back to the kandidate "Einstein, Mari" again. 11) "Sensur mottatt":<15.07.2010> and "Karakter": <B>. Then click "Lagre endringer"	The date sent to censor and the date of examination result received will be current date or earlier. Date that later than current date will consider as invalid input. Expected as type 1	System didn't check that type of invalid input. Invalid user inputs are saved. System's reaction was as type 3

Continued on next page

Table D.18 – continued from previous page

Actions	Expected Reaction	Reaction
[L] 12) Change the status to "Ferdig, men karakter ikke satt" and click "Lagre endringer". Choose an other kandidate, go back to the kandidate "Einstein, Mari" again. 13) "Sendt til sensur": <30.04.2006> "Sensur mottatt": <25.04.2005> and click the button "Lagre endringer"	The date sent to a censor will be earlier than the date of censoring result received. System detects invalid input and prompts the user	System didn't check that type of invalid input. System respond as type 3
[L] 14) Change the status to "Ferdig, men karakter ikke satt" and click "Lagre endringer". Choose an other kandidate, go back to the kandidate "Einstein, Mari" again. 15)"Sensurfrist": <06.04.2002> "Sendt til sensur": <30.04.2006> and click "Lagre endringer"	The date sent to a censor will be earlier than the deadline of censor. System detects invalid input and prompts the user	System didn't check that type of invalid input. System respond as type 3

Continued on next page

Table D.18 – continued from previous page

Actions	Expected Reaction	Reaction
16) Change the status to "Ferdig, men karakter ikke satt" and click "Lagre endringer". Choose an other kandidate, go back to the kandidate "Einstein, Mari" again. 17)"Sensurfrist": <testDate> "Sendt til sensur": <testDate> and click "Lagre endringer" 18) "Sensur mottatt": <testDate> "Karakter": <5> and click "Lagre endringer"	System detects invalid user input and prompts the user	As expected
Total tests: 11		Tests passed: 5

UC11 - Create/update account contains several use cases. Tables D.19 to D.22 captured the tests data for this use case.

Table D.19: Test data for UC11-01 - Add a student
Category - Bibliographic data

UC11-01: Add a student		Date: 02.05.06
Precondition	For all user stories in the use case UC11, the user must log in as an administrator.	
Field: Input Data	Expected Reaction	Reaction
Fornavn: 54321 Etternavn: 12345 Brukernavn: 56789	First name, last name and username can not include digits only. Expected reaction is as type 1	System didn't check that type of invalid inputs. System react as type 3
Fornavn: kiet Etternavn: ve tran Brukernavn: kietve	This username is already existed. System prints a message to tell that to the user	As expected

Continued on next page

Table D.19 – continued from previous page

Field: Input Data	Expected Reaction	Reaction
Total tests: 4		Tests passed: 1

Table D.20: Test data for UC11-02 - Change status/information for supervisor

UC11-02: Change status/information for supervisor		Date: 24.04.06
Field: Input Data	Expected Reaction	Reaction
Fornavn: 12345 Etternavn: 54321 Brukernavn: 54321 E-post: aMail	System detects invalid input and prompts the user	System didn't check that type of invalid inputs. Invalid inputs are saved
Total tests: 4		Tests passed: 0

Table D.21: Test data for UC11-03 - Change status/information for a censor

UC11-03 - Change status/information for a censor		Date: 24.04.06
Field: Input Data	Expected Reaction	Reaction
Fornavn: 12345 Etternavn: 54321 Grad/Tittel: 12345 Foreslått av: 54321 Hjemmeadresse: 345 Jobbadresse: 789 Merknad: 234	Those types of inputs can not include digits only. System detects invalid input and prompts the user	Invalid input are saved without prompting
Fødselsdato: enTest Personnr: TestAgain	System detects invalid input and prompts the user	As expected
Fra dato: fromDate Til dato: toDate	System detects invalid input and prompts the user	As expected

Continued on next page

Table D.21 – continued from previous page

Field: Input Data	Expected Reaction	Reaction
[L] Fra dato: 12.03.2009 Til dato: 12.07.2006	A censor is active from date 12.03.2009 to 12.07.2006. It's not logical, "From Date" has to be before than "To Date". System detects invalid inputs and prompts the user	Invalid input are saved without prompting
E-post i hjemmeadresse: thuyhuet E-post i Jobbadresse: enTest	The e-post adress are invalid and expected reaction is as type 1	As expected
E-post i hjemmeadresse: thuyhuet@	The e-post adress are invalid and expected reaction is as type 1	Invalid input is not recognized. System react as type 3
Telefon (Hjemmeadresse): tryAgain Telefon (Jobbadresse): againAgain	System detects telefon fields included only letters and prompts the user	Invalid inputs are saved without prompting
Total tests: 17		Tests passed: 6

Table D.22: Test data for UC11-04 - Find a supervisor/-censor by searching

UC11-04 - Find a supervisor/censor by searching		Date: 04.05.06
Field: Input Data	Expected Reaction	Reaction
Navn/Del av navn: 12345	The same input data is used to search censor and supervisor. Neither name of censor or supervisor can include digits only. System detects invalid input and prompts the user.	System didn't check that type of invalid input and searching are performed.
Total tests: 2		Tests passed: 0

There is no need for typing user inputs in use case "UC12 - Get information" . The user gets the required information by choosing a link. System fetches relevant information, process and publish to the user. This use case will be test to check that the system could respond to the user's action due to possible internal component failure.

Table D.23: Test data for UC12 - Get information

UC12: Get information		Date: 24.04.06
Precondition	The user is logged in as an administrator. In addition, some delivered master thesis are saved in database, so some data can be shown to the user.	
Actions	Expected Reaction	Reaction
Click the links with labels "Viktige sensurfrister", "Leverte masteroppgaver", "Båndlagte oppgaver", "Biveiledere"	System gets relevant information from database, process and publish to the user	As expected
Total tests: 4		Tests passed: 4

Table D.24: Test data for UC13 - Import data

UC13: Import data		Date: 09.05.06
Precondition	The user can log in as a system administrator. Do the following actions: 1) Go to http://newdaim.idi.ntnu.no/admin 2) Log in as an administrator. 3) Change your role to "System administrator" if it's not your current role. Click the button "Sysadmin" and continue to perform the actions as described below:	
Actions	Expected Reaction	Reaction
4) Click the button with label "Importere sensorer". Click "Browse" and choose the file TestFile.doc. Click "Importere"	The input file is invalid. The expected reaction is as type 1	As expected

Continued on next page

Table D.24 – continued from previous page

Actions	Expected Reaction	Reaction
5) Click "Browse", choose the file "MATHfail.csv" and click the button "Importere".	The file is missing one data column. System detects missing data in the input file. Expected response is as type 1	As expected
6) Click "Browse", choose the file "MATHfail2.csv" and click the button "Importere".	The file is included one extra column of data. System detects the file contains invalid data and responds as type 1	As expected
7) Click the following buttons "Sysadmin" –> "Importere studenter" –> "Browse" and choose a file "TestFile.doc". Then click "Importere"	System detects that the file is invalid and responds as type 1	As expected
8) Click "Browse" and choose the file "MATHfail.csv". Then click "Importere"	The file is missing one data column. System detects missing data in the input file and responds as type 1	As expected
9) Click "Browse" and choose the file "MATHfail2.csv". Then click "Importere"	The file is included one extra column of data. System detects the file contains invalid data and responds as type 1	As expected
Total tests: 6		Tests passed: 6

Table D.25: Test data for UC14-01 - Change institute information

UC14-01: Change institute information		Date: 04.05.06
Precondition	For all user stories in the use case UC14. The user is logged in as an system administrator.	
Field: Input data	Expected Reaction	Reaction
Norsk navn: 12345 Engelsk navn: 67890 Alias: 56789012 brukernavn@: 789 adresse: 234	None of input data include digits only. System expects to respond as type 1	The system respond as type 3
Kostnadsted: NotTextHere Ibx number: CannotBeAText	None of inputs data should include letters. System recovers, detects invalid inputs and prompts the user	Input "NotTextHere" is replaced as 0 value, "CannotBeAText" is replaced as "CannotBeAT". Invalid inputs are saved without prompting. The system respond as type 2.
Total tests: 7		Tests passed: 0

Table D.26: Test data for UC14-02 - Add an administrator user

UC14-02: Add an administrator user		Date: 04.05.06
Precondition	The user must log in as an system administrator. Do the following actions: 1) Go to http://newdaim.idi.ntnu.no/admin 2) Log in as an administrator 3) Change your current role to "System administrator" if you are not 4) Click the buttons "SysAdmin" –> "Administratorer" and Fill in the following <data> in the "fields":	
Field: Input Data/ Actions	Expected Reaction	Reaction
5) "Fornavn": <12345> "Etternavn": <6789> "Brukernavn": <12345> "E-post": <45678>. Choose also "aktiv" for the roles "admin" and "økonomi". Click "Lagre"	These inputs cannot include only digits. System responds as type 1	System didn't recognize that types of invalid inputs and respond as type 3
Total tests: 4		Tests passed: 0

Appendix E

Original DAIM Document

Here we present a description of DAIM system. This document is written by those who have developed DAIM system and it's original norwegian form. This is just a summary of the entire document. A attachment from this project includes an entire document of the DAIM system.

Digital **A**rkivering og **I**nnlevering av **M**asteroppgaver (DAIM), et informasjonssystem for studenters og administrasjonens arbeid med masteroppgaver.

Student
DAIM skal støtte studenten fra uttak av masteroppgaven, til den er ferdig innlevert.

Kontrakter
Studentene bruker systemet for å fylle ut kontrakter og skjema (masterkontrakt, samarbeidskontrakt og innleveringsskjema). Når nødvendige opplysningene er utfylt kan studenten skrive ut ferdig utfylte kontrakter og skjema, som bare trenger signering.

Samarbeidsgrupper kan opprettes på nett. To (eller i særskilte tilfeller tre) studenter kan samarbeide om å skrive masteroppgave. Etter at studentene har godkjent at de skal samarbeide i gruppe, kan de skrive ut samarbeidskontrakt via DAIM.

Etter innlevering av masteroppgaven via DAIM kan ferdig utfylt innleveringsskjema skrives ut fra DAIM. Lever kontrakter og skjema i instituttresepsjonen.

Digital innlevering av masteroppgaver
Masteroppgaven innleveres digitalt som en PDF fil. Det er viktig at PDF filen har fonter inkludert. Inkluderte fonter forhindrer bytte av font dersom en font brukt i dokumentet ikke finnes på skriveren til Tapir. Digitale vedlegg (kildekode, simuleringer, animasjoner, kjørende program og lignende) kan leveres som en ZIP fil.

Etter innlevert masteroppgave skal studenten levere innleveringsskjema ferdig signert i instituttresepsjonen.

Omtrent to arbeidsdager etter at masteroppgaven er innlevert, vil den ferdig trykte papirutgaven leveres til instituttresepsjonen. Ekstrabestillinger som studenten kan ha gjort ved innlevering, vil bli levert til Tapir bokhandelen studenten spesifiserte.

Administrasjon
DAIM er et verktøy for administrasjonen av masteroppgaver. Verktøyet bør benyttes både ved institutt og fakultet. Et overblikk over hva de forskjellige administrasjonsrollene gjør i systemet:

- Admin
- Sysadmin
- Økonomi
- Hovedveileder

Økonomi og hovedveileder rollene er de som får fått minst tid til utvikling i denne omgang. Det er mulighet å gjøre mer ut av disse rollene, samt å legge inn en sensor rolle.

Admin
Admin rollen håndterer administrasjon av kontrakter og skjemaer for masteroppgaver, innlegging av studenter som mangler tilgang og vedlikehold av hovedveilederliste og sensorliste. Denne rollen er best for å hente ut oversikter fra systemet.

Masteroppgaver
Studentene fyller selv ut det meste av informasjonen som trengs for administrasjon av oppgavene, slik at administrasjonen må i større grad bare kontrollere at informasjonen er rett utfylt.

Når det legges inn en start dato for masteroppgaven, og det er valg et studieprogram, så skal DAIM regne ut hva som blir innleveringsfristen. Administrasjonen har mulighet til å endre på denne fristen.

Administrasjonen kan endre status for studenter, samt legge inn merknader for hver masteroppgave og hver student. Det er også mulig å legge inn båndlegging for hver oppgave.

Både hovedveiledere og administrasjon kan legge inn sensor for oppgaver. Hovedveiledere kan bare velge blant ordinære sensorer, mens administrasjonen kan legge inn både ordinære og ekstraordinære sensorer for en oppgave.

Informasjon om: hvilken sensor, sensurfrist, når en oppgave er sendt til sensur, når den er mottatt, og hvilken karakter masteroppgaven fikk blir lagret for hver oppgave.

Både institutt og fakultet må inn å behandle masteroppgavene.

Hovedveiledere
Institutt administrasjonen er ansvarlig for å holde informasjonen om hovedveiledere ved instituttet oppdatert. Listen inneholder fast ansatte hovedveiledere ved instituttet, og studentene bruker denne listen for å velge hovedveileder til masteroppgaven.

Personer som har veiledet en eller flere oppgaver kan ikke slettes fra listen, men de kan settes til inaktiv som betyr at de ikke lenger veileder oppgaver (da kan ikke studenter velge dem som hovedveileder lenger).

Sensorer
Institutt administrasjonen er ansvarlig for å holde informasjonen om instituttets sensorer oppdatert.

Informasjon om sensor skal legges inn i når man sender forespørsel om oppnevning av sensor til fakultetet. Dermed blir en kopi av opplysningene lagt inn, og man trenger ikke å huske på å oppdatere dette senere.

Sysadmin
Sysadmin har tilgang til de samme funksjonene som admin, men har utvidede rettigheter. De kan importere data fra FS for studenter og sensorer, samt endre på informasjon for institutt, og administrere admin brukere.

NB! Det skal være en lokal sysadmin ved hvert institutt. Lokal sysadmin for et institutt er ansvarlig for import av studenter, samt administrasjon av administratorroller ved det instituttet.

Utvalgte sysadmin har utvidede rettigheter til å simulere andre brukere, samt endre på opplysninger for opptrykk av masteroppgaver.

Økonomi
Økonomi rollen skal håndtere alt som har med betaling for masteroppgaver å gjøre.

- **Mangler Honorering**
  Ubetalte oppgaver som er levert og har får satt sensor og karakter, vil vises i en liste over masteroppgaver som mangler sensurhonorar.
- **Betalte Honorar**
  Oppgaver som har fått betalt honorar vises i en egen side. Innleverte oppgaver skal finnes i ei av de to listene.

Hovedveileder
Hovedveileder skal kunne få ut en oversikt over hvilke oppgaver de har veiledet som hovedveileder.

Mine masterstudenter
Oversikten visere hvilke masterstudenter som skriver eller har skrevet for veilederen.

Hvilken sensor som er valgt for hver oppgave. Om en oppgave har fått sensur, og når den ble innlevert. Hovedveileder skal kunne velge sensor for oppgaver som ikke har fått sensur.

Mine oppgavetekster
Dette skal være en side der hovedveiledere legger ut forslag til masteroppgaver som de ønsker at studenter skal ta. Studentene skal kunne lete frem i beskrivelsene via søking i studentgrensesnittet. (Ikke implementert)

Søkeside for DAIM
Lagring og gjenfinning av digitale masteroppgaver er et av hovedmålene for DAIM systemet. DAIM lagrer beskrivelsesdata/metadata, samt de digitalt innleverte masteroppgavene slik at man senere kan gjøre disse tilgjengelige for søking.

Tittelsiden skal inkluderes automatisk først i hver masteroppgave som leveres, slik at tittel, forfatter, veileder og NTNU logo er synlig på første side av hver masteroppgave.

Det skal være mulig å søke etter alle masteroppgaver som er levert i DAIM via søkegrensesnittet. Søking etter dokumenter skal ikke kreve noen form for innlogging.

Masteroppgavene som blir vist skal:

- Ha fått sensur
- Student skal ha godtatt publisering
- Ikke være båndlagt

Søkesiden er viktig for å vise forskningen som gjøres ved NTNU, og hvilke fagretninger forskjellige veiledere er villige til å veilede for.

Søkesiden vil kunne bli NTNU sitt vindu mot omverdenen. En kilde til inspirasjon og kunnskap for informasjonssøkere.

Bibliography

[1] Procedures for performing failure mode effects and critically analysis, November 1994. US MIL-STD-1629A/Notice 2.

[2] Egon Berghout and Rini Van Solingen. The goal/question/metric method: a practical guide for quality improvement of software development. Technical report, 1999.

[3] Thomas M. Connolly and Carolyn E. Begg. *Database systems. A Practical Approach to Design, Implementation, and Management.* Addison Wesley, second edition, 1999.

[4] F. Cristian. Exception handling and tolerance of software faults. Technical report, University of Newcastle, 1995.

[5] R. G. Dromey. A model for software product quality. In *IEEE Transactions on Software Engineering, Vol. 21, No. 2*, pages 146–162, Februar 1995.

[6] Martin Fowler and Kendall Scott. *UML Distilled. A Brief Guide to the Standard Object Modeling Language.* Addison-Wesley, December 2000.

[7] P. L. Goddard. Software fmea techniques. In *Proceedings Annual Reliability and Maintainability Symposium*, pages 118–123, 2000.

[8] Steven D. Gribble. Robustness in complex systems. Technical report, Department of Computer Science and Engineering. The University of Washington, 2001.

[9] Marnie L. Hutcheson. *Software Testing Fundaments.* Wiley Publishing, 2003.

[10] IEEE. *IEEE Standard Glossary of Software Engineering Terminology, IEEE Std 610.12-1990. Corrected edition*, February 1991.

[11] I. Jacobson and et al. M. Christerson. *Object-Oriented Software Engineering. A Use Case Driven Approach.* Addison-Wesley, 1992.

[12] Jonathan Kohl. *Watir User Guide.* Watir development, http://wtr.rubyforge.org/.

[13] Glenford J. Myers, Tom Badgett, Todd M. Thomas, and Corey Sandler. *The Art of Software Testing.* John Wiley and Sons, second edition, November, 2004.

[14] NTNU, http://newdaim.idi.ntnu.no. *Digital Arkivering og Innlevering av Masteroppgave(DAIM).*

[15] NTNU, http://www.idi.ntnu.no/grupper/su/websys.html. *WEB-based SYStem.*

[16] Jeff Offutt, Ye Wu, Xiaochen Du, and Hong Huang. Bypass testing of web applications. Technical report, Information of Software Engineering. George Mason University.

[17] Jiantao Pan. *Software testing.* Carnegie Mellon University, http://www.ece.cmu.edu/.

[18] Ian Sommverville. *Software Engineering.* Addison Wesley, fifth edition, 1999.

[19] T. Stålhane. *WEBSYS Memo. Robustness and FMEA.* NTNU, http://www.idi.ntnu.no/grupper/su/websys.html.

[20] Arne Sølberg and D.C. Kung. Structured analysis and design. Technical report, Springer Verlag, Berlin-Heidelberg, 1993.

[21] David Talby, Ori Nakar, Noam Shmueli, Eli Margolin, and Arie Keren. *A Process-Complete Automatic Acceptance Testing Framework.* Lund University, Computer Science, http://www.cs.huji.ac.il/ davidt/.

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

[23] Yanlong Zhang, Hong Zhu, Sue Greenwood, and Qingning Huo. Quality modelling for web-based information systems. Technical report, School of computing and Mathematical Science, 2001.

[24] Jianyun Zhou and Tor Stålhane. *Early Robustness Assessment for Web-based systems.* IEEE, 2004.

[25] Jianyun Zhou and Tor Stålhane. What is robustness? Technical report, NTNU, WebSys report number 0306-01.

[26] Hong Zhu, YanLong Zhang, Qingning Huo, and Sue Greenwood. Application of hazard analysis to software quality modelling. Technical report, Oxford Brookes University, 2002 IEEE.

[27] Trond Marius Øvstetun, Christian Schwarz, and Stein Kåre Skytteren. *Automatic Acceptance Testing of Web Applications(AutAT).* NTNU and Bekk, http://boss.bekk.no/boss/autat/.

[28] Trond Marius Øvstetun, Christian Schwarz, and Stein Kåre Skytteren. *Installation guide for AutAT.* NTNU and Bekk, http://boss.bekk.no/boss/autat/install.php.