NTNU

Innovation and Creativity

Analysis of Software Faults using Safety-techniques with Respect to the Software System DAIM

Jostein Dyre-Hansen

Master of Science in Computer Science
Submission date: June 2007
Supervisor: Reidar Conradi, IDI
Co-supervisor: Jon Arvid Børretzen, IDI

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

Problem Description

Analysis techniques from safety-critical development, such as PHA or HazOp, shall be used on DAIM, a system used at IME for starting, delivering and finishing master thesis. Documents to be analyzed are from the specification and design phase of the system development. The results obtained from using these techniques shall be compared with existing fault reports with actual faults as have been discovered in the system. These fault reports shall also be treated and analyzed.

Assignment given: 22. January 2007
Supervisor: Reidar Conradi, IDI

Abstract

In this master's thesis we have analyzed the software system DAIM, which is a web-based delivery system used at NTNU in connection with master's theses and master students, with respect to software faults. Based on the documentation from the design stage of the DAIM project we have performed a technique called Preliminary Hazard Analysis (PHA), which is an analysis technique from safety-critical development. The results from this analysis have been compared with existing fault reports containing actual faults discovered in the system. Some of the intention behind our work has been to find if hazards identified with PHA can be related to actual faults found in the fault reports. In [17] it is stated that correcting software faults in later phases of the software development is much more expensive than in earlier phases and we have performed the PHA to see if some of the faults could have been avoided. We found that there were some connections between some of the faults and hazards identified, but the results were not entirely as expected.

In our previous work [5] we did a similar kind of analysis as we have done in this work regarding the analysis of fault reports and we have compared the results from our previous work with some of the results that we have obtained from this work to see how the distribution of fault types varies between the projects. The results showed that there were several differences between the projects, but some similarities were also discovered.

Preface

This master's thesis was written in the period from January 2007 to June 2007 and is the result of my MSc degree at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU) in Trondheim.

First I would like to thank my supervisor Jon Arvid Børretzen for his contribution to this master's thesis in the shape of excellent guidance, advice and support. I would like to thank Professor Reidar Conradi for defining the master's thesis and helping me getting started. Kai Torgeir Dragland did a good job assisting me in questions regarding the DAIM system. I will also give thanks to Professor Tor Stålhane and members of the BUCS project for their contribution.

Trondheim, June 2007

Jostein Dyre-Hansen

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter covers the introduction to the master's thesis and it will include the context, definition, motivation, background and outline for the rest of the master's thesis.

1.1 Context

The context of this master's thesis is taken from the research project BUCS (BUsiness Critical Software) [2] which includes participants from IDI (Institute of Computer and Information Science) at NTNU (Norwegian University of Science and Technology) in Trondheim and some companies in the Norwegian IT-industry. BUCS is a research project in the area of computer science with focus on helping developers, users and customers developing safe software. The overall goal in BUCS has been to develop methods to improve support for analysis, development, operation and maintenance of business-critical systems and the project is financed by NFR (Research Council of Norway).

In accordance to developing safe software, a system called DAIM has been developed at NTNU for supporting several processes concerning master students and master's theses, and in this master's thesis we will use a safety analysis technique for analyzing this system. The method that we will use is called PHA (Preliminary Hazard Analysis) which is a method used in the early design phase with the goal of revealing potential dangers in early project development. The main reason for doing this analysis is first and foremost to improve the DAIM system, but in addition to this, a master's thesis was written last year with another approach when analyzing the DAIM system. In [11] the author was also concerned with the DAIM system, but the method used in the analysis process was FMEA which is a method that deals with analysis of robustness. As stated above we will now use another method for analyzing the DAIM system.

1.2 Definition

The main title of this master's thesis is stated as:

> *"Analysis of software faults using safety-techniques with respect to the software system DAIM."*

A more detailed description follows:

> *"Analysis techniques from safety-critical development, such as PHA or HazOp, shall be used on DAIM, a system used at IME for starting, delivering and finishing master's thesis. Documents to be analyzed are from the specification and design phase of the system development. The results obtained from using these techniques shall be compared with existing fault reports with actual faults as have been discovered in the system. These fault reports shall also be treated and analyzed."*

This master's thesis will consist of a literature study of what exists regarding different safety-techniques used today. As can be read above we have focus on software faults and we will use a hazard analysis technique called PHA when analyzing the software system DAIM. In addition to this we have received fault reports from the same system which we will classify and analyze. The results from the PHA and the analysis of the fault reports will then be compared and we will try find if there are any connections between the results from the PHA analysis and the fault report analysis. In our previous work [5] we did a similar kind of analysis regarding fault analysis and we will use the results from our previous work and compare them with the results that we will find in this work.

1.3 Motivation

There are several reasons for doing analysis on software faults. First of all removing faults in earlier phases in the software development is of interest of everybody because correcting and removing faults will become more and more expensive as the project evolves. According to [17], correcting software faults in later phases of the software development is more expensive than in earlier phases and the quote below comes from an analysis concerning the fault correction process in a large-scale SDL production model:

> *"Faults undetected within the originating phase took approximately eight times more effort to correct."*

Having in mind that the hazard analysis technique PHA typically is performed in earlier phases of the system development, this have motivated us to try out the PHA technique to see if PHA can be used to reveal faults earlier in the system development process of a given system. In this thesis we will also analyze fault reports from the same system and by doing this we will get the chance to compare the results from the PHA and the analysis of fault reports to see if some faults could have been removed earlier.

Aside from the economical aspect there is also a safety aspect related to analysis regarding software faults. The hazard analysis technique that we will use in this master's thesis resides from safety-critical development where safety is of highest priority. As more and more of the tasks today are handled by computer systems, there is a need for safe and robust systems which can handle unforeseen situations. Safety-critical systems such as flight controller systems are systems that need a high degree of fault prevention, because of the consequences a fault in these kinds of systems can cause to the environment. A plane crash can be catastrophal for people, buildings and the environment. As the aircraft industry becomes more and more dependant on new technology such as software, the importance of making technology with as few faults as possible must be the goal to pursue.

Faults occur in most systems and will probably always occur, but if the faults can be revealed in earlier phases of the development, then both the economical and safety aspect will be fulfilled in a better way. The expression ”look before you leap, a stitch in time saves nine” may be suitable in this connection, because safety-critical analysis is much about reducing the risk for anything unwanted to happen before it actually happens.

1.4 Background

This master's thesis is mainly devoted to fault analysis in the area of system development within computer science. Our previous work [5] was also concerned with fault analysis of system development within computer science, but in a different setting and with different goals. In our previous work we cooperated with an external actor, namely a Norwegian software company in the banking and financial sector and we received fault reports from different projects. The fault reports from the projects were categorized according to a classification scheme and then analyzed with respect to different criterions such as what are the most common faults, what faults are considered most severe and if there were trends between the different fault reports from the different projects etc. In addition to this it was a literature study regarding what is being done within fault analysis within computer science with focus on the terminology used; we found that the terminology used depends on who has written it and sometimes it is difficult to get a generic understanding of the different concepts because different people see things differently. For example the terms ”mistake”, ”fault”, ”error”, ”failure”, ”anomaly” occurred several times, but according to the literature the definitions could be different. Another important issue was that of finding a classification scheme to use for the process of categorizing the different faults. Several classification schemes exist and they all have weaknesses and strengths. In this master's thesis we have taken into account the experiences we gained from our previous work and we have tried to use some of what we learned back then.

1.5 Outline

Below we will give the outline of this master's thesis with information of what can be found in the different chapters within this master's thesis:

Chapter 2 - Prestudy In this chapter we will look into different concepts concerning safety-critical systems. We will give definitions of different terms often used in this context and we will present different safety-critical techniques in the area of hazard analysis.

Chapter 3 - DAIM In this chapter we will introduce DAIM which is the basis for our research study. A brief overview of the system and the development history will be given in addition to key information that we will need in the rest our work with this master's thesis.

Chapter 4 - Research Questions and Method This chapter introduces the research context and process in the report. We will present the material to be analyzed and how the material will be analyzed. The challenges we have met will be presented as well as the choices that we had to make.

Chapter 5 - Hazard Analysis This chapter contains the results from the hazard analysis that we have performed in our work. The reasons for using the particular method that we have used will also be given in addition to some information regarding the process.

Chapter 6 - Fault Classification and Analysis In this chapter we will present our results from the classification of the fault reports from DAIM and we will analyze our findings. We will also compare the results obtained from the analysis of DAIM with the results from previous work which also included fault analysis but with different systems in focus.

Chapter 7 - Results In this chapter we will present the results from our work and we will point out the most important findings that we have made.

Chapter 8 - Discussion of Validity and Own Contribution This chapter contains the discussion part of the master's thesis along with own contributions. The discussion part is closely related to the threats to the validity of our work.

Chapter 9 - Conclusion and Further Work This chapter will present the conclusions that we have made in addition to some comments about what could be interesting to study in further work.

Chapter 2

Prestudy

In this chapter we will look into different concepts concerning safety-critical systems. We will give definitions of different terms often used in this context and we will present different safety-critical techniques in the area of hazard analysis.

2.1 Introduction

In Section 2.2 we will focus on safety-critical systems and safety in general and introduce different concepts and definitions. Section 2.3 will describe some concepts and terms often used in safety-critical analysis such as definitions of the term "fault" and when to apply different techniques for managing faults. In our previous work [5] we were concerned with fault analysis and since this master's thesis also will contain fault analysis, Section 2.4 will briefly look into this. Section 2.5 takes into account the different requirements that are often referred to in system development as well as safety requirements. In Section 2.6 we will introduce the concept of hazard analysis and different methods concerning hazard analysis will be presented.

2.2 Safety-Critical Systems

When discussing safety-critical systems it will be natural to define the term **safety** first. According to page 2 in [13], safety is defined as:

> ***Safety*** *is a property of a system that it will not endanger human life or the environment.*

The terminology concerning safety varies a lot and the definitions of various concepts are defined differently in the literature. In [10] a more general definition of safety is given as:

> ***Safety*** *is freedom from accidents or losses.*

A **safety-related system** is defined as [13]:

> *A **safety-related system** is one by which the safety of equipment or plant is assured.*

Further it is stated that a safety-related system is a synonym for the term safety-critical system.

The definitions mentioned above are very general, but they contain the most important issues when dealing with safety-critical systems. Safety-critical systems are systems that may cause a lot of harm to people and/or the environment. The main goal in safety-critical systems is to reduce the risk of anything wrong happening and the impact if something wrong does happen. Some examples of safety-critical systems can be aircraft flight control systems or signal systems for trains. If these systems fail, both people and the environment will probably suffer dependant of how serious the failure is. Making systems without failures is very difficult and probably not realistic, but it is important to produce software and hardware that can handle unforeseen problems that appear and try to reduce the impact of these problems by doing for example thorough analysis before and during software development. In the example mentioned above with aircraft flight control systems, the complexity of the system is quite huge and flight accidents may have catastrophal consequences. The terms faults, errors and failures are terms that often are used in this context and in Section 2.3.1 we will look at some definitions.

According to [10], page 244, there is a lack of software safety engineers with good knowledge of both hardware and software although a few companies have established system safety engineering positions. Further it is stated that the software safety engineer should have overall responsibility for analysis concerning software safety, participating in software design reviews and also contribute in the process of identifying software hazards and establish a connection between the system development and the hazard analysis process. Establishing an effective safety program in a company is demanding and it requires systematic methods and organizational controls. In [10] the chapter 11.1.4 is devoted to issues concerning system safety in organizations and includes issues such as duties, responsibilities, personnel qualifications, subcontracting etc. For example it is proposed that the software safety management and personnel should be included in the safety program from the start and there should be frequent meeting activity among all the participants involved to ensure that the different hazards are handled in the best possible way.

Having employees just focusing on safety issues is desirable, but for different companies this leads to challenges. Small companies may not afford having persons just focusing on safety issues while big companies may have the possibility to do that. The key is to have a balance between a proper degree of safety and the related costs.

There exist several methods for analyzing software systems and the different methods have their advantages and disadvantages. Some of the methods are best suited in certain situations while other methods can be used in many situations and in Section 2.3 we will describe different approaches for handling faults. Techniques referred to as hazard analysis are often used earlier in the system development process and are used for trying to avoid unwanted situations to happen. In Section 2.6 we will go into detail about hazard analysis and describe different methods often used; PHA and FMEA are examples of techniques referred to as hazard analysis. An example of a technique that is used in later stages of the system development process including when the system is operative, is inspection of fault reports either from the system itself, such as system logs, or fault reports created by the users of the system when they have discovered that something is wrong. This technique differs from hazard analysis in the way that it is a reactive way of handling faults while the hazard analysis tries to find treats/hazards before they occur and is of a more proactive nature. In Section 2.4 we will say more about classification and analysis of fault reports.

2.3 Different Approaches for Managing Faults

Different approaches for managing faults in a software system exist, and what approach to use depends on several factors such as how far one has got in the development process and how critical the system is. In the literature one refers to different techniques concerning management of faults and from page 2 in [13] four groups of techniques are described:

Fault avoidance These techniques aim to prevent faults from entering the system during the design stage. Resources should be allocated in the design stage and have focus on avoiding faults.

Fault removal This approach deals with methods for finding faults within the system before it enters service. Software and hardware testing techniques should be involved in this process.

Fault detection These techniques focus on systems that are operational. The aim here is to detect faults while the system is running and try to minimize the effects of the different faults.

Fault tolerance The goal is to make systems that operate correctly even though faults do occur.

2.3.1 Faults, Errors and Failures

The terminology within the area of computer science varies a lot. In our previous work [5] we did a study of the differences concerning the terminology

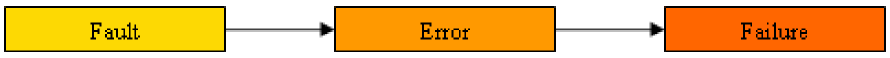


Figure 2.1: Different phases.

used and we could conclude with that there are several variances referring to the terminology used in the literature. Sometimes different authors define similar concepts differently which makes it harder to get a common understanding of the different concepts. In this report we will use some of the same definitions as we used in our previous work and Figure 2.1 describes the transitions between the terms fault, error and failure.

On page 2 in [13] some definitions are given and we will use them as a basis in this report. The definitions:

> ***A fault*** *is a defect within the system.*
>
> ***An error*** *is a deviation from the required operation of the system or subsystem.*
>
> ***A system failure*** *occurs when the system fails to perform its required function.*

Based on Figure 2.1, one can see that a fault leads to an latent error where a fault can be a mistake done by the programmer. When the error is triggered, it produces wrong and erroneous data and this affects the delivered service and a failure will occur. One of the goals in safety-critical development is that latent errors should not result in system failure. There are several techniques for not letting that happen such as the one obtained by interconnecting several modules in such way that if one module fails other modules are ready to take over and system failure has been avoided.

2.4 Fault Analysis

In our previous work [5] we did research on fault reports from online-systems and we used different techniques when analyzing and classifying the different fault reports. This section will include some elements from this work as we will use the same way of categorizing faults as we did in [5]. In Section 2.4.1 we will present the classification scheme that we will use in our work.

2.4.1 Orthogonal Defect Classification

The Orthogonal Defect Classification scheme (ODC) was developed by Ram Chillarege in 1990 at IBM Research and it is an example of a scheme for classifying faults, or defects, which is the term used in this scheme. The article [4] says that ”The goal is to provide an in-process measurement paradigm to extracting key information from defects and enable the metering of cause-effect relationships. According to [3], ODC gathers and converts data and information in software defects to valuable measurements in the software development process and/or the product. These measurements are then used for a variety of management and technical purposes. The reasons for using ODC are among other factors cost reduction, quality improvement, schedule management, process diagnostics.

With ODC the developer/tester categorizes a defect into classes that point to the part of the development process which needs attention [4], see Figure 2.3 [1] for an illustration. In the process of software development the activities are mainly divided into design, code and test, but there exist variations. It is of importance that the classification schemes must have consistency between the different stages considering that the different process stages sometimes may overlap or that the different releases are being developed in parallel. In a huge project the different process stages may be developed by different people and/or different organizations. When the classification schemes offer consistency between the different stages, it is possible to look at trends among the various stages. In Figure 2.2 [2] one can clearly see that there is a connection between the different defect types found in the various stages. As the development project evolves, one can see that the number of function defects decreases with the various stages taken into consideration that the design stage is performed before the coding and testing stage. Another interesting aspect according to [4] is that the classification scheme should be independent of the specifics of a product or organization. If these two conditions are fulfilled, the classification schemes may provide relationships and models that are very useful.

In the ODC taxonomy the term attribute is used for categorizing and an example of such an attribute is the attribute ”defect type”. The user that corrects the defect chooses the defect type. The different types of defects should be very obvious and leave no room for confusion. There is made a distinction between something ”missing” or ”incorrect”. The number of defect types was initially five, but has been extended to eight and the different defect types used in ODC are illustrated in Table 2.1 [4].

It should also be mentioned that there exist variants of the ODC classification scheme. In [7] the ODC classification scheme is extended with 3 more defect types and in this approach we have a total of 11 defect types. In addition to the fact that 3 more defect types have been added, some of the original defect types have also been replaced by new defect types. Table 2.2 illustrates the defect types used in [7].

[1]http://www.chillarege.com/odc/articles/odcconcept/node3.html 17.06.2007 15:00

[2]http://www.chillarege.com/odc/articles/asqc/node8.html 17.06.2007 15:00

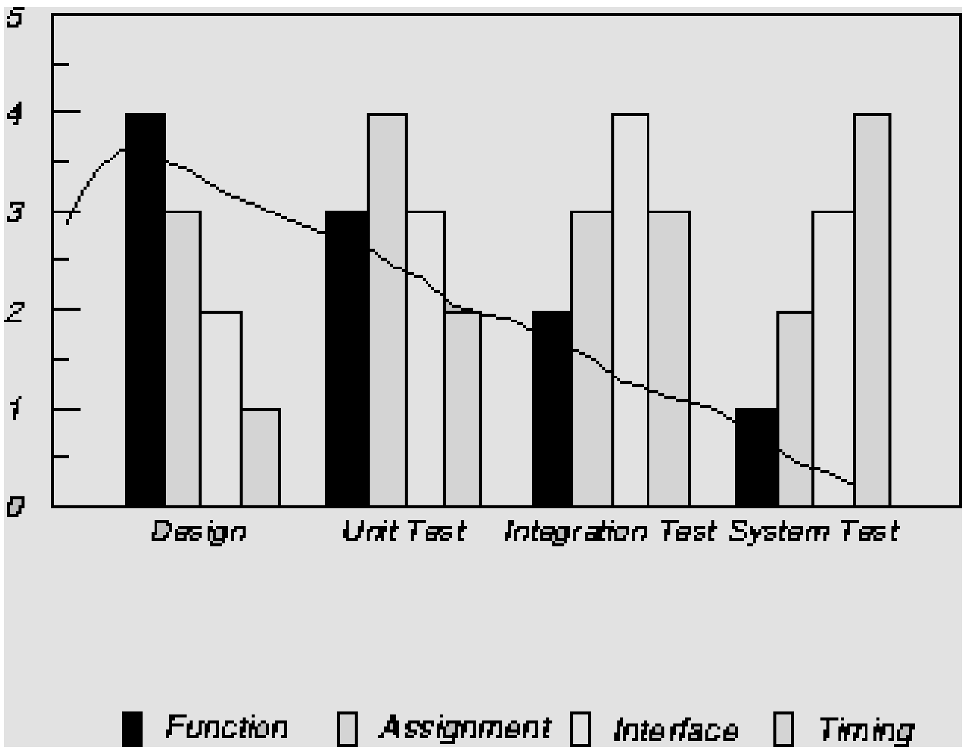
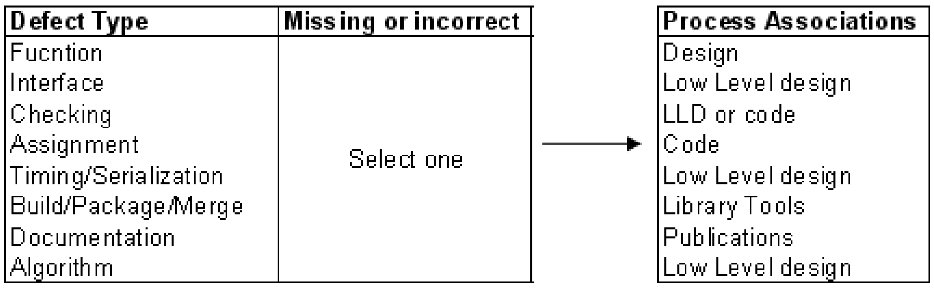


Figure 2.2: Number of faults in the different stages in the development process.



Defect Type	Missing or incorrect
Fucntion	
Interface	
Checking	
Assignment	Select one
Timing/Serialization	
Build/Package/Merge	
Documentation	
Algorithm	

Process Associations
Design
Low Level design
LLD or code
Code
Low Level design
Library Tools
Publications
Low Level design

Figure 2.3: The figure illustrates the fault types and associations to the different process stages.

ODC defect types
Function
Assignment
Interface
Checking
Timing/Serialization
Build/Package/Merge
Documentation
Algorithm

Table 2.1: Defect types in ODC.

Variant of ODC defect types
Function
Assignment
Interface
Checking
Build/Package
Documentation
Data
Memory
Environment
Naming conventions
Understandability

Table 2.2: A variant of ODC defect types.

2.5 Requirements

In system development projects there are often requirements connected to the projects which have to be fulfilled for the customer to be satisfied. Typically one refers to customer requirements which is further divided into functional requirements and non-functional requirements. In systems where safety is important there is often attached a document concerning the safety requirements, which is supposed to stake out what is required from the system concerning safety issues. These documents state what the system should do and should not do for maintaining the wanted level of safety. In Section 2.5.1 we will look closer into some system requirements often referred to in system development while Section 2.5.2 briefly will introduce requirements concerning safety.

2.5.1 System Requirements

System requirements may vary a lot from system to system. It all depends on what kind of system one is dealing with, such as the complexity of the system, the size of the system and the need for safety etc. The purpose of the system should absolutely be taken into consideration when deciding on the system requirements. Some systems have more focus on safety than others while other systems demand for a high degree of up-time. If these system requirements are stated early in the process, it might be easier to fulfill the requirements because they are in focus from the start. There exist different terms for describing the various general system requirements and in this section we will base our terminology on the pages 20 to 25 in [13].

The first system requirement that we will take into account is **reliability**, which is defined as:

> ***Reliability*** *is the probability of a component, or system, functioning correctly over a given period of time under a given set of operating conditions.*

The requirement reliability may vary a lot from system to system, because the different systems may have different needs. Some systems are expected to operate for a long time, maybe for several years and maintenance may be impossible due to difficult conditions etc. Therefore it is necessary to have this in mind from the start of the development and take this into consideration. It is also a matter of money, because systems with a high degree of reliability are probably more expensive to develop than others.

Availability is another requirement that has to be considered and it is defined as:

> *The* ***Availability*** *of a system is the probability that the system will be functioning correctly at any given time.*

The two terms availability and reliability are often mentioned in the same context, but the two terms have quite different meanings. When speaking of reliability one refers to a period of time while availability refers to a particular point in time. Reliability does not include any actions for repairing a system that has been brought down. Said in other words reliability can be thought of as the time that it will take a component, part or system to fail while it is operating. Unlike availability, reliability does not take into consideration the time needed for repair to get the unit back into a working condition. According to [15], availability can be thought of as the probability that the system performs its required function whenever someone calls it given that the system has not been brought down or is being repaired. Availability is a function of reliability and maintainability. As an example one could think of a system that operates correctly 99 hours out of 100 hours, and then one can say that average availability during the period is 99/100, or 99 percent.

System integrity is another system requirement that needs to be mentioned in this context and the definition is given as:

> *The **integrity** of a system is its ability to detect faults in its own operation and to inform a human operator.*

In Section 2.3 we listed four different groups of techniques for managing faults and system integrity is related to the group named fault detection. System integrity is of highest importance in highly safety-critical systems, such as in railway signalling systems or aircraft controller systems where the consequences can be catastrophic if something happens that was not supposed to happen. One have to realize that faults do occur, but the important thing is to have systems that can handle faults in such a way that the user of the system is noticed about the fault and then can take actions based on the error message and make sure that the error is handled in a reasonable way. The damage the situation may cause, can then be avoided or at least be reduced to a minimum. Preferably one should create systems that can handle different faults itself without any human intervention, but this is easier said than done. There are situations where human attention might be necessary such as in situations where there are subjective value judgements involved.

Another important system requirement which deserves some attention is **data integrity** and according to [6] data integrity is often defined as:

> ***Data integrity** refers to the prevention of unauthorized and improper data modification.*

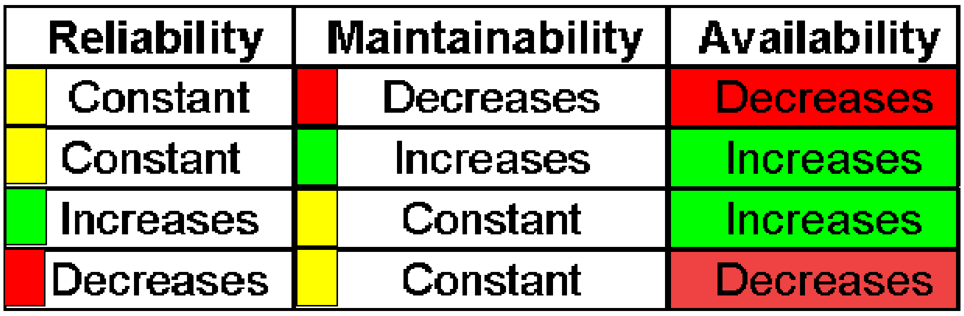
Probably with most of the computer systems in use it is a wish that there is a high degree of data integrity. The degree of how much integrity the different systems need varies a lot from system to system. Some systems are quite critical and need a high degree of integrity while other systems are not that critical demanding a lesser degree of integrity. Even though a high degree of integrity is preferred, there is also a matter of economy. As stated earlier in this chapter, it is more expensive to develop a system with a higher degree of reliability and the same is truth for the requirement of integrity. It is important to have a reasonable balance between the costs and the degree of fulfilling the system requirements. Some systems may be too expensive to develop if all requirements should be too extensively implemented and companies may prefer to take a higher risk instead of too high development costs. Then if something happens it may be cheaper to correct the problems afterwards instead of trying to make a 100 percent fault proof system which might not even be possible. There are several examples on systems that need a high degree of integrity such as systems in the area of banking and finance, military systems and telecom. The consequences of faults related to integrity can be catastrophal in these systems.

Maintainability is given as another system requirement and there is given a definition of **maintainability** and **maintenance**:

> ***Maintenance*** *is the action taken to retain a system in, or return a system to, its designed operating condition.*
>
> ***Maintainability*** *is the ability of a system to be maintained.*

With most computer systems, maintenance is of high priority for different reasons. As new technology comes and requirements change there is a need for maintenance for keeping the systems up-to-date. Problems arise in shape of for example failures and maintenance is necessary for correcting the different faults which have caused failures to the systems. The time needed for maintenance of different systems varies from system to system and depends on several factors. Systems that are well implemented in the first place would probably not need that much attention compared to systems not that well planned and implemented, but this may not be true in all situations. Other factors such as complexity and size can alone be reasons enough for maintaining a high level of maintainability for different companies.

When speaking of maintainability it is also appropriate to include the two terms reliability and availability because the terms are closely related. From Figure 2.4 [15] we can see that if the reliability is constant and the maintainability is decreasing, it will result in decreasing availability. If the maintainability is increasing, the result will be increasing availability. In the case where the maintainability is constant, increasing reliability will lead to the availability increases, while decreasing reliability will result in decreasing availability. As one understands, the system does not only depend on how frequent the system is out of order, but also the time needed to restore it. Having this in mind makes it easier to understand that the different system requirements are closely related and need to be seen and treated as a whole.



Reliability	Maintainability	Availability
🟨 Constant	🟥 Decreases	🟥 Decreases
🟨 Constant	🟩 Increases	🟩 Increases
🟩 Increases	🟨 Constant	🟩 Increases
🟥 Decreases	🟨 Constant	🟥 Decreases

Figure 2.4: The relationship between availability, reliability and maintainability.

There are several ways to measure the different system requirements and both qualitative and quantitative approaches exist. With respect to the maintainability there exist a well known measure, namely **Mean Time To Repair** abbreviated MTTR. According to [1] we have that:

> *The **Mean Time To Repair** is the arithmetic mean of the elapsed time between the first status of Incident Reported and the corresponding status Defect Resolved for the set of all resolved defects.*

As stated in the article [1] one should be careful with comparing measures like these between companies because there are different viewpoints from company to company. A random user will probably think of the time to repair from where he/she reports the failure until it is solved while a company will measure the MTTR from the moment they receive the fault report from their own helpdesk.

The term **dependability** is defined in [9] and we present it here:

> *Computer system **dependability** is the quality of the delivered service such that reliance can justifiably be placed on this service.*

In [9] reliability, availability, maintainability and safety are said to be the measures of dependability. For a system to be dependable, a combination of the measures just mentioned has to be included and then it is easier to quantify the dependability. Having mentioned service it will be appropriate to define it [9]:

> *The **service** delivered by a system is the system behavior as it is perceived by another special system(s) interacting with the considered system: its user(s).*

According to [9] the life of a system can be perceived by its users as being in one of two possible states:

- Service accomplishment, where the service is delivered as specified.
- Service interruption where the delivered service is different from the specified service.

In this section we have defined the requirements reliability, availability, system integrity, data integrity, maintainability and dependability. These requirements are often referred to when speaking of requirements concerning computer systems and should be closely considered when developing computer systems. If possible all of the requirements should be fulfilled to a highest possible degree, but this might not always be possible. Making a system that fulfills all requirements 100 percent costs a lot of money and the companies involved may be forced to reduce the degree of fulfilment because of this, resulting in a tradeoff between the cost and the degree of fulfilment.

2.5.2 Safety Requirements

In safety-critical development there is also a need for safety requirements in addition to the system requirements mentioned in Section 2.5.1. As discussed in Section 2.2 safety requirements are about avoiding incidents that can result in accidents which can cause harm to people, property or the environment. In the context of a computer system where the computer system deals with safety-critical issues, there might exist a lot of components controlled by the computer system, that could cause harm to the surroundings. These undesirable incidents are often termed **hazards** and the term hazard will be defined in the next Section 2.6.

One of the main issues when dealing with safety is to handle these hazards in the best possible way. There are several stages in the process of handling hazards and the main stages defined in [13], page 25-26, are:

- Identification of the hazards associated with the system.
- Classification of these hazards.
- Determination of methods for dealing with the hazards.
- Assignment of appropriate reliability and availability requirements.
- Determination of an appropriate safety integrity level.
- Specification of development methods appropriate to this integrity level.

2.6 Hazard Analysis

There are several definitions of the term **hazard**, but in our work we will use the definition found in [14]:

> *A **hazard** is a state or set of conditions of a system or an object that, together with other conditions in the environment of the system or object, will lead to an accident.*

Hazards are considered as threats to people and the environment and some examples of hazards can be a car accident, stroke of lightening, plane crash, brake failure. These incidents may result in catastrophic consequences like people being hurt or even killed; buildings and means of transport may be damaged or destroyed. With all the hazards there is a risk associated and [14] defines risk as:

> ***Risk** is defined as the product of an event's consequence and its probability of occurence or as its hazard level (severity and likelihood of an occurence) combinded with 1) the likelihood of the hazard leading to an accident and 2) hazard exposure or duration.*

Having introduced the term hazard it will be natural to introduce the compounded term **hazard analysis** which is, as the name implies, an analysis of hazards. It is important to accept the fact that hazards will always exist and hazard analysis is about reducing the risk for the hazards to happen. There are several steps involved in the process of analyzing hazards and there exist different approaches. In advance of the hazard analysis it is important to establish goals or purposes, because the steps to be taken depend on this. Having defined different goals or purposes, makes it easier to choose the most appropriate steps in order to make the analysis more reasonable.

There are different analytical techniques used in the area of hazard analysis. Each technique provides different viewpoints and some techniques may be better suited for some kind of tasks than other tasks and therefore it is important to find the best suitable technique dependant on what kind of task one shall handle.

According to [10] the techniques most widely used are:

- Failure Modes and Effects Analysis (FMEA)
- Failure Modes, Effects and Criticality Analysis (FMECA)
- Hazard and OPerability studies (HAZOP)
- Event Tree Analysis (ETA)
- Fault Tree Analysis (FTA)

- Preliminary Hazard Analysis (PHA)

Section 2.6.1 to Section 2.6.5 will describe these techniques with more details.

2.6.1 Failure Modes and Effects Analysis

Failure Modes and Effects Analysis was one of the first systematic methods for analyzing faults in technical systems and is often abbreviated **FMEA**. The method was developed by reliability engineers in the late 1950s with the goal of predicting equipment reliability mainly in military systems. In its original form it is a variant of reliability analysis that place importance on successful functioning rather than hazards and risk. Its main goal is to calculate the overall probability for a product to operate without failure for a given length of time. Over the years the method has been further developed and has been used in several areas, but the main content has been the same with just minor variations dependant on what tasks it was intended to solve [12] [10].

The FMEA technique is usually performed in the beginning of a system development project, often in the design phase. The main objective is to identify parts or properties of a system that need to be improved in order to satisfy the reliability and safety requirements which are expected. In [12] there are defined several intermediate goals which we will give in the list below:

1. Identify every possible failure state for every component in the system.

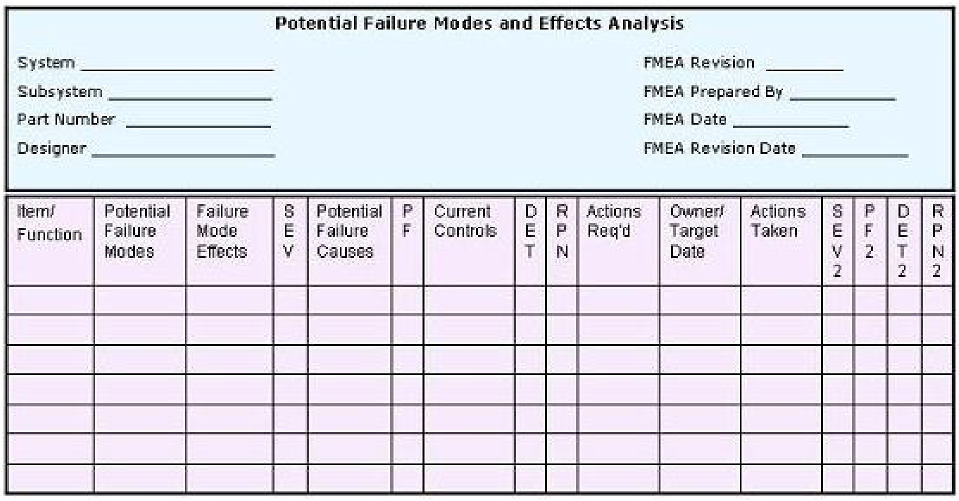
2. Determine the causes of the failure states.

3. Determine the failure states impact on the system as a whole.

4. Determine the severity of the different fault effects.

The FMEA technique is mainly a qualitative analysis, but could also be of a quantitative nature when assigning probability and frequency to the different failure modes. The technique is quite easy and does not involve a special technique or algorithm, but it requires that the participants have thorough knowledge of the system [12]. When doing the FMEA analysis a FMEA scheme for filling out the details concerning the different failures has to be completed. There exist different kinds of FMEA schemes and an example[3] is given in Figure 2.5.

[3]www.siliconfareast.com 23.03.2007 15:00


Potential Failure Modes and Effects Analysis

System ____________________ FMEA Revision ________

Subsystem __________________ FMEA Prepared By ___________

Part Number ________________ FMEA Date ____________

Designer ___________________ FMEA Revision Date ___________

Item/ Function	Potential Failure Modes	Failure Mode Effects	S E V	Potential Failure Causes	P F	Current Controls	D E T	R P N	Actions Req'd	Owner/ Target Date	Actions Taken	S E V 2	P F 2	D E T 2	R P N 2

Figure 2.5: An example of a FMEA scheme.

Failure Modes, Effects and Criticality Analysis

Failure Modes, Effects and Criticality Analysis, abbreviated **FMECA**, is actually an extension of FMEA [12] [13] [10]. The main difference between FMEA and FMECA is that FMECA contains a more detailed approach concerning the criticality of the failures which involves a description or ranking of the failures according to criticality.

2.6.2 Hazard and OPerability Analysis

Hazard and OPerability Analysis (**HAZOP**) is a hazard analysis technique used in several areas. The technique, which originates from the chemical industry, was developed in the early 1960s in England and later improved upon and published by the Chemical Industries Association (CIA).

According to [10] HAZOP is based on a system theory model of accidents that assumes that accidents are caused by deviations from the design or operating intentions and an example could be that when expecting a forward flow, a backward flow or no flow at all appears. The technique demands for creativity in discovering all possible hazards and operating problems that can occur. HAZOP is of a qualitative nature where the participants systematically try to identify all possible deviations from the design's expected operation and all hazards associated with these deviations. A difference between HAZOP and the other techniques mentioned in this chapter is that the other techniques require that hazards are identified before the analysis.

In Figure 2.6 [4] an example of a scheme used in connection with the

[4]http://www.frascati.enea.it 12.04.2007 14:00

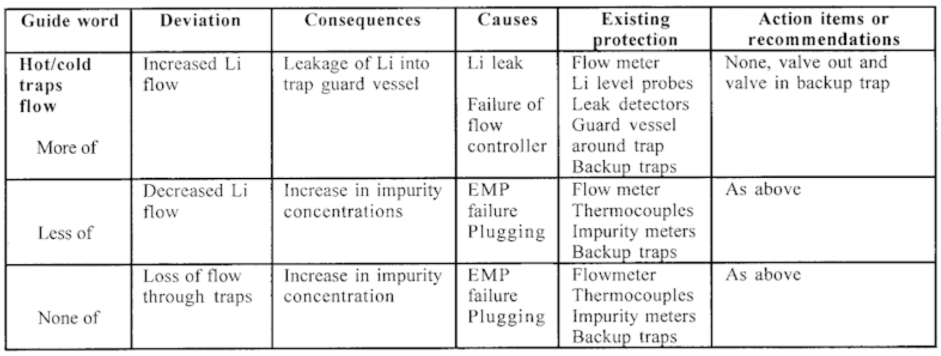
Guide words	Meaning
NO, NOT, NONE	The intended result is not achieved, but nothing else happens (such as no forward flow when there should be).
MORE	More of any relevant physical property than there should be (such as higher pressure, higher temperature, higher flow, or higher viscosity).
LESS	Less of a relevant physical property than there should be.
AS WELL AS	An additivity occurs in addition to what was intended, or more components are present in the system than there should be (such as extra vapors or solids or impurities, including air, water, acids, corrosive products).
PART OF	Only some of the design intentions are achieved (such as only one of two components in a mixture).
REVERSE	The logical opposite of what was intended occurs (such as backflow instead of forward flow).
OTHER THAN	No part of the intended result is achieved, and something completely different happens (such as the flow of the wrong material).

Table 2.3: Guide words for HAZOP

HAZOP process is illustrated. The leftmost column contains the **guide words** used in the process. There exist different approaches concerning the guide words and in [10] a list of guide words is given which we have presented in Table 2.3. In the column **Deviation**, the deviation from the requirements specification is given and an example from the table is ”Increased Li flow”. Given the deviation, it is easier to see what the consequences are and column **Consequences** deals with this. The causes for the deviations are described in the column **Causes**. The column **Existing protection** describes what kind of protection mechanisms that exist if the deviation occurs. The column **Action items or recommendations** describes the action to be performed if a deviation is being present.

The strength of HAZOP is its simplicity and ease of application. In addition to finding failures HAZOP has the potential to find more complex types of hazardous events and causes. By using HAZOP there has been documented a reduction of the number of hazards and problems encountered in operation [10].

When performing the HAZOP, normally 4-6 persons should be present in addition to the HAZOP leader and a secretary and the time needed to perform the analysis depends on the size and complexity of the system to be analyzed [12]. Normally HAZOP is performed at the point in time where



Guide word	Deviation	Consequences	Causes	Existing protection	Action items or recommendations
Hot/cold traps flow More of	Increased Li flow	Leakage of Li into trap guard vessel	Li leak Failure of flow controller	Flow meter Li level probes Leak detectors Guard vessel around trap Backup traps	None, valve out and valve in backup trap
Less of	Decreased Li flow	Increase in impurity concentrations	EMP failure Plugging	Flow meter Thermocouples Impurity meters Backup traps	As above
None of	Loss of flow through traps	Increase in impurity concentration	EMP failure Plugging	Flowmeter Thermocouples Impurity meters Backup traps	As above

Figure 2.6: An example of a scheme used in the HAZOP process.

the design is finished and after, but there exist variants of HAZOP such as Preliminary Hazard Analysis (PHA), see Section 2.6.5, which may be used in earlier phases for revealing what dangers and problems that can arise.

2.6.3 Event Tree Analysis

Event tree analysis (**ETA**) is a well known technique used in risk analyzes and this technique has been used in several huge projects such as the American nuclear program WASH-1400 in the 1970s. Event tree analysis may resemble fault tree analysis in some aspects such as the use of a tree structure, but the two techniques have a big difference in the way the trees are constructed; fault trees starts with identifying the top event first while event trees start with finding initiating events that lead to possible outcomes, "top events". Section 2.6.4 presents fault tree analysis (FTA).

ETA is concerned with finding chains of events that follow from a particular initiating event which in turn can result in accidents of different kinds. As can be seen from Figure 2.7, each event has a probability attached to it which makes it possible to calculate the likelihood of the different events to happen.

According to [12] an ETA is normally performed in six steps:

1. **Identification of initiating event** The initiating event can be a fault of technical or human nature and it is important that the initiating event could give rise to a chain of possible events.

2. **Identification of safety functions** The safety functions are the systems' defence and are supposed to hinder or reduce the effect of a initiation event. The people involved in this process will have to identify all safety functions which can change the result of the initiating event with respect to what order the events are activated in. Examples

can be automatic shutdown systems, alarms in case of fire, procedures to be followed in emergencies etc.

3. **Construction of the event tree** There are different standards for drawing the event tree, but normally the event tree is drawn chronologically from left to right, where the leftmost event is the initiating event which triggers successor events. For each safety function that is being activated there are two branches, either "success" or "fail". When the tree is constructed, one can trace a path through it by choosing a branch under each successive safety function and a probability for different accidents can be calculated because each branch has got a probability assigned to it. Figure 2.7 [5] gives an example of an event tree.

4. **Description of the resulting chain of events** This is a qualitative part of the analysis where the analyst is supposed to describe the resulting chain of events. Some of the chains of events show that the safety functions work as planned and result in situations where the system continues operation or has to be shut down in a safe way. The analyst should also rank the consequences according to their severity and this should be used when assigning resources for improving different parts of the system; some parts need more attention than other parts of the system etc.

5. **Calculation of the probability of the identified consequences** If there exist reliability data for the initiating event and the safety functions, a quantitative analysis of the system can be performed. Concerning the initiating event we need information about the number of events per unit of time and for the safety functions we need a measure of probability stating that they work satisfactorily when activated.

6. **Presentation of the results** At the end of the analysis a presentation of the event tree has to be done.

In accordance with FTA, ETA may also be performed by one analyst, but preferably it should be performed in a group of two to four persons in a brainstorming session. At least one of the persons involved should have experience with ETA analysis and the rest of participants should have good knowledge to the system. For achieving good quantitative analyzes relevant reliable data should be present [12]. The ETA process should be introduced after most of the design of the system is complete [10].

ETA gives a clear picture of the chain of events that follows after a fault in the system which makes it a suitable technique for improving different parts of the system as for example safety. As stated above FTA differs from ETA

[5]http://www.ece.cmu.edu/~koopman/des_s99/safety_critical/Event_tree.jpg 15.05.07 13:55

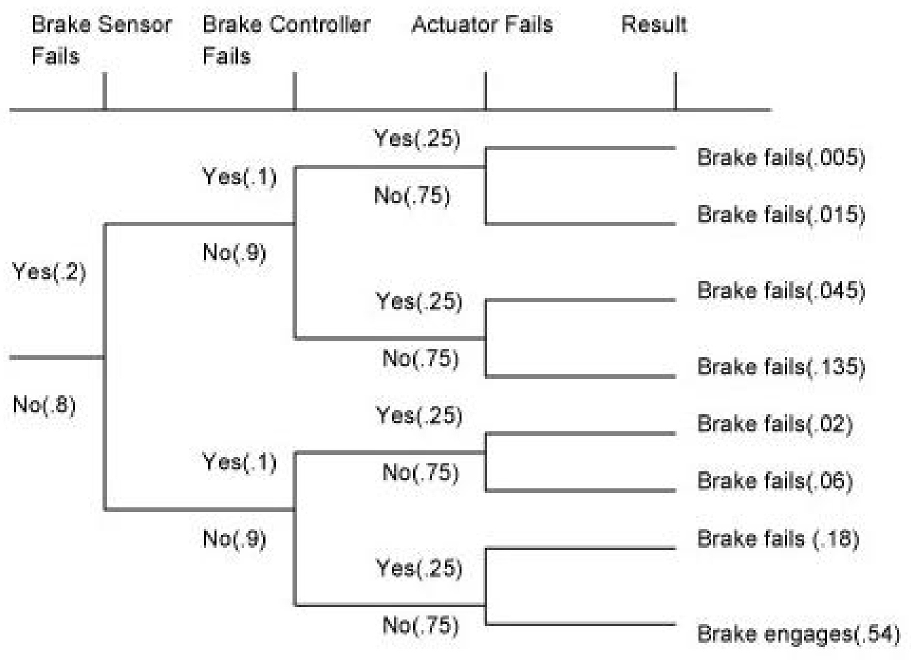


Figure 2.7: An example of an event tree.

primarily in the way relationships between events are handled. While FTA offers a snapshot of the system state pointing out the relationship between events, ETA focuses on the relationship between sequences of events that are linked and assigned a probability. In theory event trees are better suited when working with notions of continuity (logical, temporal and physical) while fault trees have an advantage when identifying and simplifying event scenarios. ETA and FTA are together the most used analysis methods in risk analyzes because of their simplicity and that they both are well documented.

2.6.4 Fault Tree Analysis

Fault Tree Analysis (FTA) was developed by Bell Telephone Laboratories in the early 1960s in connection with the evaluation of the Minuteman Launch Control System for missile launch. The aerospace company Boeing adopted this analysis technique and has developed the technique further [12].

According to [10] FTA is primarily used for analyzing causes of hazards and not for identifying hazards. FTA is a top-down search event and the top event of the fault tree has to be identified by other techniques. In contrast to ETA, discussed in Section 2.6.3, FTA starts with all identified hazards and works backwards to determine potential causes while ETA do the opposite namely starting with all possible events and based on these determine the outcome [13]. Normally FTA is performed in four stages [10] and a short summary follows:

1. **System definition** This stage includes the decisions of determining the top event, initial conditions, existing events and impermissible events.

2. **Fault Tree Construction** When the stage above is finished, the construction of the fault tree begins. Based on the top events, the analyst use his/her knowledge to find the causal events which leads to the top event by using logic symbols for describing the causal relations. Several logical operators exist, for example the "AND" and "OR" operators, but there are others too.

3. **Qualitative analysis** The main task here is to reduce the tree to a logically equivalent form that shows the specific combinations of basic events sufficient to cause the top event. The main goal is to find the *minimal cut set.*

4. **Quantitative analysis** The quantitative analysis uses the minimal cut sets from the qualitative analysis to calculate the probability of occurrence of the top event from the probability of the occurrence of the basic events. The probability of the top event will be the sum of the probabilities of all the cut sets if they are all statistically independent.

The FTA technique is based on boolean logic and a diagram with boolean symbols is used to illustrate the relationship between an undesired incident and its causes. From Figure 2.8 [6] which contains a simple fault tree for a brake system, it can easily be seen that "Brake Fails" is the top event which may be the outcome if all of the underlying events occur. As one can see from Figure 2.8 there are two boolean operators, one "or" operator and one "and" operator, and if the inputs from "Brake Sensor Fails", "Brake Controller Fails" and "Brake Actuator Fails" are all true, this will result in that the top event "Brake Fails" will be true and the brakes will fail. The FTA analysis is best suited for discrete events rather than continual events because of the boolean nature of the technique. The fault tree is actually a snapshot of the system at one point in time and does not allow for many changes before the analysis will be difficult to manage.

The FTA analysis can be done by one analyst, but preferably it should be done in a group of two to four people [12]. At least one of the members of the group should have experience with fault tree analysis and the rest of the group should have good knowledge to the system. When doing such an analysis it is demanded that one have a thorough understanding of the

[6]http://www.ece.cmu.edu/ 26.03.2007 17:00

system operations, the various failure modes of the different system components in addition to the effects that these failure modes have on the system. Such information can for example be achieved from the FMEA process, see Section 2.6.1, which is being performed in the earlier stages of the system development. One of the advantages with FTA analysis is that it is easy to explain to a third party which is not familiar with the analysis and this is way this method is one of the most used analysis methods in risk analysis. Another advantage one can get benefit from using this method is that the person doing the analysis is forced to understand the system very well, and this may result in that while doing the analysis it might be easier to find and correct latent faults.

2.6.5 Preliminary Hazard Analysis

Preliminary Hazard Analysis (**PHA**) is based on a technique developed by the U.S. Department of Defence (DoD). The technique which handles risk analysis is widely used and it has proven to be effective within safety related work done in DoD and safety analyzes of process plants [12].

PHA is normally carried out in an early phase of the development and the goal is to reveal potential hazards as early as possible in the project development. After having identified several hazards, actions can be taken to control the hazards either by eliminating them or reducing their impact on the system being developed. According to [12], the PHA is normally executed in three steps:

1. **Gather necessary information** Considering that the PHA is carried out in the early design phase, there is probably limited information regarding the system available and this should encourage the participants to find information concerning similar systems and experiences which might be useful to start off with.

2. **Execution of the analysis** In this phase the analysis of the different hazards should be executed and it starts with identification of hazards, critical events and other events that may result in unwanted consequences. The participants involved in the analysis process shall also identify design criterions or alternatives that can eliminate or reduce the dangers. After having identified several hazards the participants have to identify the different causes why the hazards may occur. Further the consequences of the different hazards should be registered and actions to be taken for removing or reducing the possibility of the hazards to happen have to be stated.

3. **Documentation of the results** The results from the steps above should be documented in some kind of a scheme with key information from the analysis.

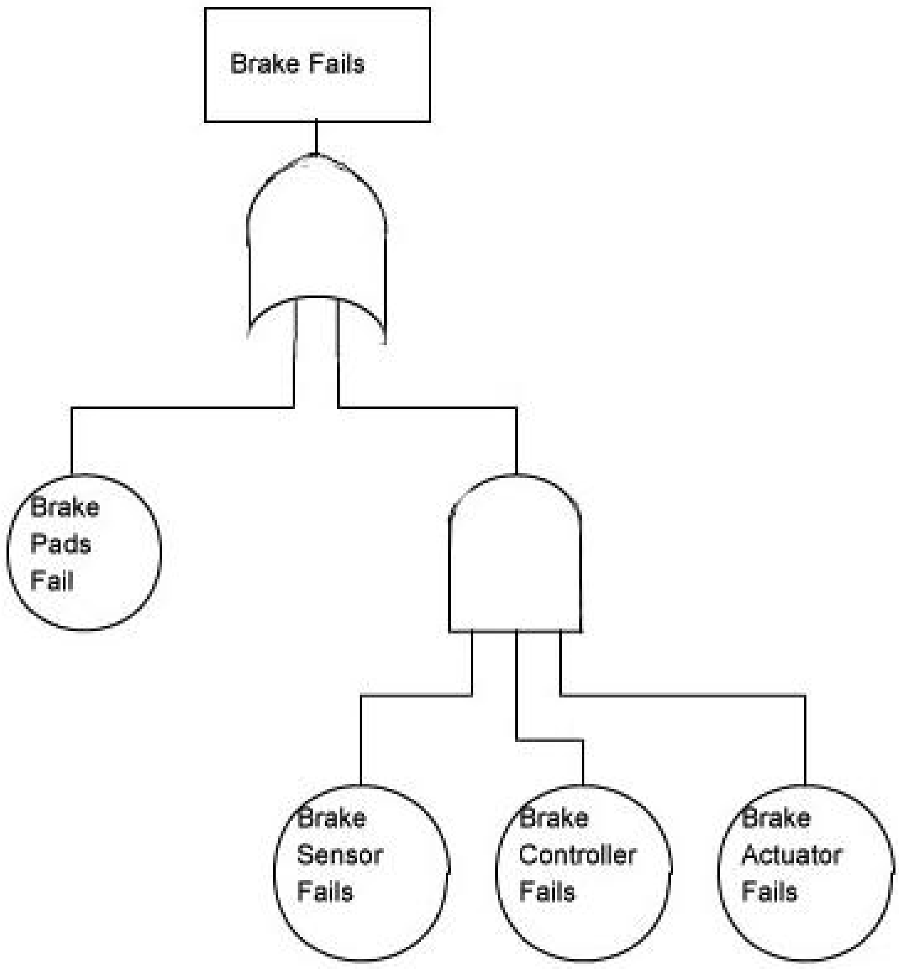


Figure 2.8: An example of a fault tree.

No.	Hazard	Cause	Level	Effect	Category
1	....	....	....	....	....
2	....	....	....	....	....

Table 2.4: Hazard identification form.

The PHA is a quick and easy analysis to do and should be executed early in the design phase of the system. According to [12] the analysis should be performed by one or two experienced engineers and as stated in item 3 in the list above, they should use a scheme for registering the information related to the different hazards. In Table 2.4 we give an example of such a scheme that can be used in the analysis process [10]. There exist several approaches when performing a PHA and what data to record varies from approach to approach. In [10] a more detailed scheme is proposed which includes information such as:

- System, subsystem, unit (equipment grouping where the potential hazard exists).
- Category (hazard level).
- Operational phase when hazardous.
- Organizations responsible for ensuring that safeguards are provided for the specific hazard.
- Verification methods (tests, demonstrations, analysis, inspection) to verify that the hazard is effectively controlled.
- Remarks and status of the hazard resolution process. The hazard is closed when it has been verified that the recommended actions have been implemented and are effective.

2.7 Summary

In this chapter we have looked at different issues that are important in this master's thesis. We have presented issues such as safety-critical systems, fault analysis, different approaches for managing faults and hazard analysis. Considering that the hazard analysis is one of the main tasks in our work, we have made a table that summarizes and compares the different hazard analysis techniques in Table 2.5 according to certain criterions. The comparison is mainly based on the information given in [12].

	FMEA	**HazOP**	**ETA**	**FTA**	**PHA**
Formalization	High	Moderate	Moderate	High	Low
Process Stage	Medium	Medium	Early	Late	Early
Costs	Low	Moderate	Low	Moderate	Low
System Info Requirements	Moderate	Moderate	High	High	Low
Complexity level	Low	Moderate	Moderate	Moderate	Low
Experience required	Moderate	Moderate	High	High	Moderate

Table 2.5: Summary of the hazard analysis techniques

Chapter 3

DAIM

In this chapter we will introduce DAIM which is the basis for our research study. A brief overview of the system and the development history will be given in addition to key information that we will need in the rest our work with this master's thesis.

3.1 Introduction

In Section 3.2 we will present the DAIM system which we will analysis in this master's thesis. We will give a brief development history and describe the key functionality of the system. Section 3.3 will describe the functional requirements of the DAIM system in addition to a diagram of use cases related to the system. This will be the basis for the analysis in Chapter 5.

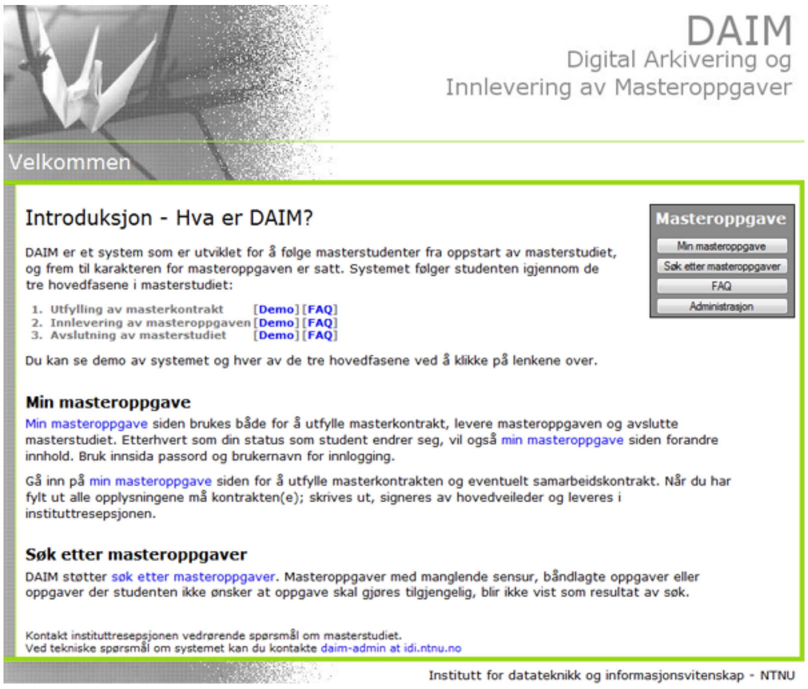
3.2 About DAIM

DAIM[1] is an abbreviation for "Digital Arkivering og Innlevering av Masteroppgaver" and is a system used at NTNU for dealing with the administration of master students and master theses. The system is supposed to support the master students in the three phases:

1. Complete the master's thesis contract
2. Delivery of the master's thesis
3. End the master study

In addition to support the master students when dealing with the administrative details concerning the master's thesis, DAIM also offers another functionality. This functionality makes it possible for the people that visits

[1]http://daim.idi.ntnu.no 10.04.2007 15:00



Introduksjon - Hva er DAIM?

DAIM er et system som er utviklet for å følge masterstudenter fra oppstart av masterstudiet, og frem til karakteren for masteroppgaven er satt. Systemet følger studenten igjennom de tre hovedfasene i masterstudiet:

1. Utfylling av masterkontrakt [Demo] [FAQ]
2. Innlevering av masteroppgaven [Demo] [FAQ]
3. Avslutning av masterstudiet [Demo] [FAQ]

Du kan se demo av systemet og hver av de tre hovedfasene ved å klikke på lenkene over.

Min masteroppgave

Min masteroppgave siden brukes både for å utfylle masterkontrakt, levere masteroppgaven og avslutte masterstudiet. Etterhvert som din status som student endrer seg, vil også min masteroppgave siden forandre innhold. Bruk innsida passord og brukernavn for innlogging.

Gå inn på min masteroppgave siden for å utfylle masterkontrakten og eventuelt samarbeidskontrakt. Når du har fylt ut alle opplysningene må kontrakten(e); skrives ut, signeres av hovedveileder og leveres i instituttresepsjonen.

Søk etter masteroppgaver

DAIM støtter søk etter masteroppgaver. Masteroppgaver med manglende sensur, båndlagte oppgaver eller oppgaver der studenten ikke ønsker at oppgave skal gjøres tilgjengelig, blir ikke vist som resultat av søk.

Kontakt instituttresepsjonen vedrørende spørsmål om masterstudiet.
Ved tekniske spørsmål om systemet kan du kontakte daim-admin at idi.ntnu.no

Institutt for datateknikk og informasjonsvitenskap - NTNU

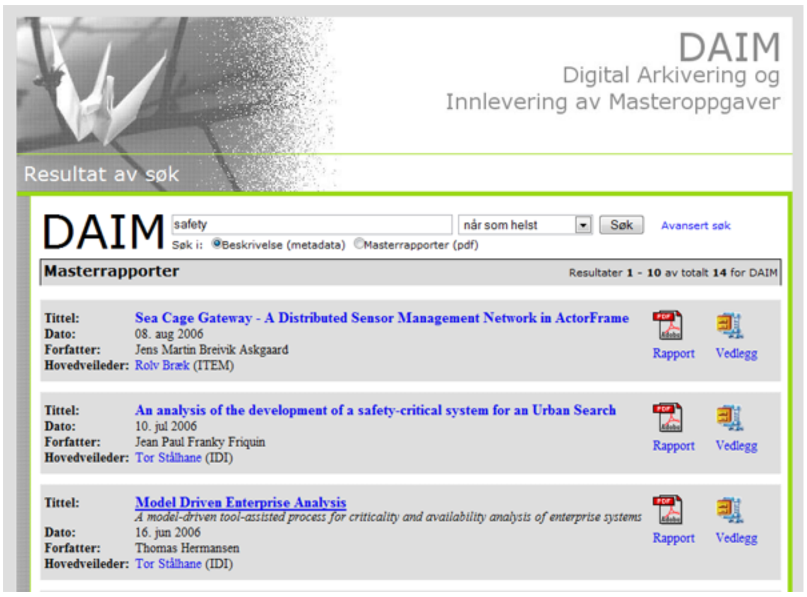
Figure 3.1: Screenshot of the DAIM webpage.

the DAIM webpage [1] to search for master's theses that have been delivered over the years. Figure 3.2 illustrates this search functionality where the term "safety" has been used to find master's theses that include this term. Master theses related to the term "safety" are then listed on the web page based on some kind of ranking algorithm that returns master's theses in an order according to relevance of the term. Master's theses that have not been censored or master's theses that are classified will not be included in the result set. Master students may also choose to not publish their master's theses.

3.2.1 A Brief Development History

The first version of DAIM was developed by Tapir Uttrykk [2] and the duration of the project was from late autumn 2004 to spring 2005. This version was much more simple than the DAIM version that we know today and it provided only limited functionality. The system provided an interface for uploading the master's theses in the format "PDF" which then was sent to the print shop for printing. Then a digital copy of the master's thesis in addition to some description data was sent to the Department of Computer and Information Science.

[2]http://www.uttrykk.no 21.05.2007 12:30


DAIM
Digital Arkivering og
Innlevering av Masteroppgaver

Resultat av søk

DAIM safety | når som helst | Søk | Avansert søk
Søk i: [✓ Beskrivelse (metadata)] Masterrapporter (pdf)

Masterrapporter — Resultater **1** - **10** av totalt **14** for DAIM

Tittel: Sea Cage Gateway - A Distributed Sensor Management Network in ActorFrame
Dato: 08. aug 2006
Forfatter: Jens Martin Breivik Askgaard
Hovedveileder: Rolv Bræk (ITEM)
Rapport Vedlegg

Tittel: An analysis of the development of a safety-critical system for an Urban Search
Dato: 10. jul 2006
Forfatter: Jean Paul Franky Friquin
Hovedveileder: Tor Stålhane (IDI)
Rapport Vedlegg

Tittel: Model Driven Enterprise Analysis
A model-driven tool-assisted process for criticality and availability analysis of enterprise systems
Dato: 16. jun 2006
Forfatter: Thomas Hermansen
Hovedveileder: Tor Stålhane (IDI)
Rapport Vedlegg

Figure 3.2: Screenshot of the search functionality on the DAIM webpage.

In the summer/autumn 2005 the Department of Computer and Information Science at NTNU wanted to take part in this project and upgrade the system, which resulted in that the responsibility was shared between Tapir Uttrykk and the Department of Computer and Information Science. The reasons for this were several and we have listed the most important ones below:

- DAIM was supposed to support the entire process from the start to the end concerning the master's thesis.
- Make it scalable such that other Departments at NTNU could be integrated in the DAIM system.
- Control the process of developing DAIM; meaning that Tapir Uttrykk should not alone be responsible for the expenses and decisions concerning DAIM.

In August 2005 the process of upgrading the DAIM system started with design of new functionality and the database for storing master theses. The implementation started in the end of November 2005 and was supposed to be finished in the middle of December 2005, but the project was delayed resulting in that the functionality required for withdrawal of master's theses was finished in early January 2006. The deadline for withdrawal of master's theses for master students was 15th of January 2006 making the system just in time to offer its services. The functionality regarding the delivery and administration of master's theses was developed in the period from January to February/March 2006 making the system able to handle the deliveries of master's theses in June.

3.2.2 User-Oriented Development

One important goal for the DAIM system was to make the system as user friendly as possible. Several methods were used for trying to achieve this and in the following we will present some of the methods used.

During the development students and people from the administration were gathered to perform paper prototyping. This was done to identify the users needs and expectations and for trying to avoid situations were the users would not understand how the system should be used. By doing paper prototyping this may have resulted in that the most obvious misunderstandings have been avoided. As a part of the paper prototyping several interviews were carried out and the parties involved were given the chance to come up with suggestions for improvement regarding the logical structure of the work flow and language used for text fields and buttons. Based on all the information that came out of this process, the graphical user interface and the database scheme were developed. Having this information available before the implementation started, made it easier for the programmer because he/she did not have to make that many decisions as is the case when the requirements specification is not properly elaborated. In the case where the requirements specification is poorly elaborated many of the decisions have to be taken by the programmer and having in mind that the programmer will never have the same understanding of the domain as the users of the system, this may lead to many problems and misunderstandings. User-oriented development is one of the main pillars when developing systems that works. A system can be flawless, but if no one uses the system because of poor usability the system will still be a flop.

3.3 Functional Requirements

In our work with the DAIM system we have had to take into account its functional requirements. Here we will present the functional requirements as use cases because the DAIM system is role-based and use cases are categorized by various roles. Figure 3.3 which describes the different use cases and the following textual use cases are taken from [11].

Student

- UC01: Log in. Every internal actor, such as student, administrator, system, administrator, supervisor and economy have to be logged in before they can do anything at all. External users who just want to search for master's theses at NTNU do not need to be logged in.

- UC02: Fill in contract. The required information about a master's thesis has to be filled in before deadline for the selection of the master's thesis.

- UC03: Create cooperation group. Two or maybe three students can cooperate in writing a master's thesis. The students cooperating have to be added when the contract for the master is filled in.

- UC04: Generate contract/schema. Students can generate contract/schema if they have filled in all information required and the administrator has approved the contract. This use case contain three user stories:
  - 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's thesis. Students can upload several files for their master's thesis. Two days after delivery, the paper versions of the master's thesis are delivered at the institute office. Extra copies can also be ordered during 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 information that is related to the paper version, such as total page numbers, page numbers that will be printed in color, comments that will be sent to Tapir and/or the institute. In addition, the decision whether the master's thesis should be made public or not is done at this stage.
  - UC05-04: Order extra copies of a student's master's thesis.

External users

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

Economy

- UC07: Handle payments. The economy role handles every payment for the master's thesis. When the actor logs in, the default page will show the following information:
  - UC07-01: A list of master's theses which still miss the payments for censoring.
  - UC07-02: Change status of master's theses which have been censored and the censors have got paid.
  - UC07-03: A list of master's theses with censors having been paid.
  - 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's thesis. The user can get an overview of delivered master's theses by choosing a combination of several parameters or by searching. This use case consists of the following stories:
  - UC08-01: The master's thesis can be chosen by several parameters, such as status of the master's thesis, or type of study. The student name, supervisor and the title will be shown.
  - UC08-02: The user can also sort the master's thesis found by student, supervisor, and title.
  - UC08-03: It is possible to search for the master's thesis by student name, supervisor, censor, title or the date that the master's thesis was delivered, censored date etc.
- UC09: Choose censor. Administrator, system administrator, and supervisor can choose a censor for one specific master's thesis. The censor can be found in a list. The supervisors can only choose a censor for the master's thesis that they have supervised.
- UC10: Validate information. A master's thesis has different status, e.g. "Registered, but master contract is not ready", "Writing the master's thesis", "Finished, but the grade is not decided". Generally, there are different information that can be validated, dependent on the status of the master's thesis. When a student has filled in the required information in the master contract, the administrator/system administrator has to check if the information is correct and then approve the master contract. If the information is not filled in correctly, the administrator will have to contact the student and tell him/her what has to be changed. 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 information that is related to the master's 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 registered.
  - UC11-02: Create/Update a supervisor's account. Information of registered supervisor can be updated. A supervisor account can be created if it is not registered.
  - UC11-03: Create/Update a censor's account. A censor account is created if it does not exist. Information of registered supervisor 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's theses. This use case includes the following user stories:
  - UC12-01: Get an overview of important deadlines for censoring.
  - UC12-02: Get an overview of information about restricted master's theses.
  - UC12-03: Get an overview of information about secondary-supervisors and external supervisors.
  - UC12-04: Get an overview of information about delivered master's theses.

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 administrators can import data from a local database for students and censors. The data must be saved in 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.

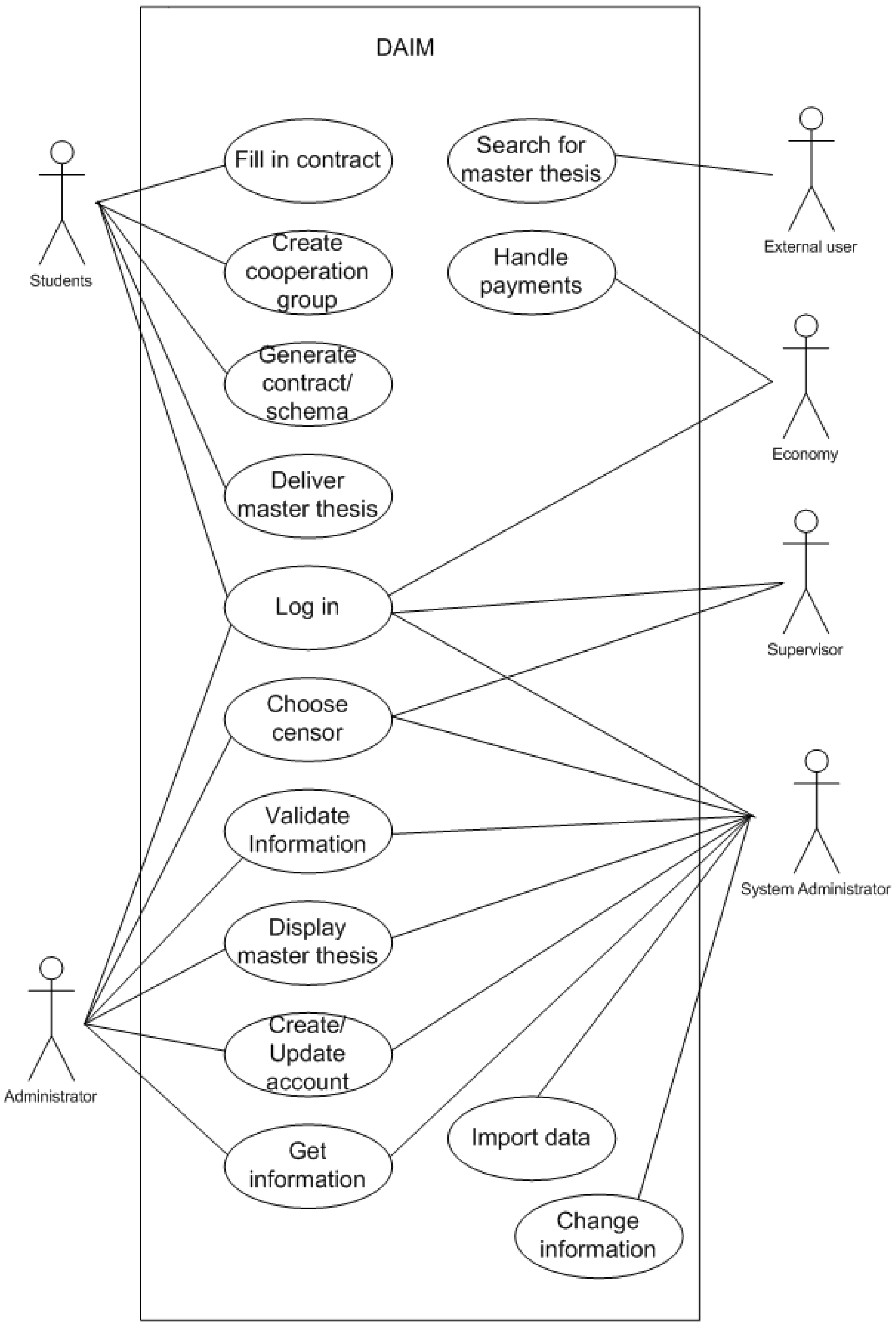


Figure 3.3: Use case diagram from the DAIM system.

3.4 Summary

This chapter contains information concerning the IT-system DAIM, which we are about to analyze in the following chapters. We have presented key information such as what kind of system DAIM is, a short development history and the functional requirements which we will base some of the analysis in the following chapters on.

Chapter 4

Research Questions and Method

This chapter introduces the research context and process in the report. We will present the material to be analyzed and how the material will be analyzed. The challenges we have met will be presented as well as the choices that we had to make.

4.1 Introduction

Section 4.2 presents the context of our work while Section 4.3 states the research questions that we have decided on. In Section 4.4 we introduce the research method and process that we have used in our work. Information about how we collected our data will be given in Section 4.5. Much of our work has been concerned with classification of faults and hazards and Section 4.6 has been devoted to the classification scheme that we have used in our work. We were faced with some challenges in our worked and they are stated in Section 4.7.

4.2 Context

Our work is mainly concerned with analysis of software development projects and the use of techniques for identifying faults and hazards. There is a difference between the concepts "hazard" and "fault" and in this master's thesis we will try to find if we can see if there are any connections between the two concepts. From Section 2.6 we have that an hazard can be seen as a treat to a system and if the developers do not take this treat into consideration when designing the system, sooner or later it may result in a system failure. In connection with IT systems there are often reports generated based on the behavior of the system such as fault reports or the users/developers of the system may report that something is wrong. Faults retrieved from fault reports manifest themselves in the system as failures seen from the users or developers point of view and have to be corrected after they have appeared. The main difference between hazard analysis and fault analysis is that the former analysis is supposed to foresee problems before they appear while the latter analysis handle the problems after they have appeared. Section 2.3.1 and Section 2.4 contain more information concerning faults and fault analysis.

The software development project that we have chosen to investigate is called DAIM and it is a system used at NTNU in connection with master students and master's theses. More detailed information concerning DAIM can be found in Chapter 3. One of the goals with this master's thesis is to perform a hazard analysis technique called PHA on the DAIM system to see if this method can be of value with respect to fault analysis and fault prevention. Even though DAIM has already been implemented and is up running, we have received specification documents from earlier phases of the development and the PHA analysis will be performed based on the design documentation from the system. In addition to the PHA, we have received fault reports from the same system which we will classify and analyze. Compared to our previous work [5] this work is not of such practical nature and a company would not get the same benefit as they did in our previous work, because here we compare different methods and results to see if the methods lead to the same results, or if some methods discover faults that other methods do not discover.

4.3 Research Questions

The research questions that we have come up with in this master thesis are stated below and some of the research questions concern the fault reports from DAIM and our previous work while some of the research questions are related to the PHA. RQ4 involves both the fault reports and the PHA.

RQ1: How is the distribution of fault types from the fault reports analyzed in DAIM?

RQ2: How does the distribution of fault types from the fault reports analyzed in DAIM differ from the fault reports in our previous work?

RQ3: What kind of fault types does PHA indicate as possibilities?

RQ4: Does the hazard analysis technique PHA applied to the DAIM specification documents reveal faults that have actually appeared in the system?

The research questions will be answered in the chapters that follow.

4.4 Research Method and Process

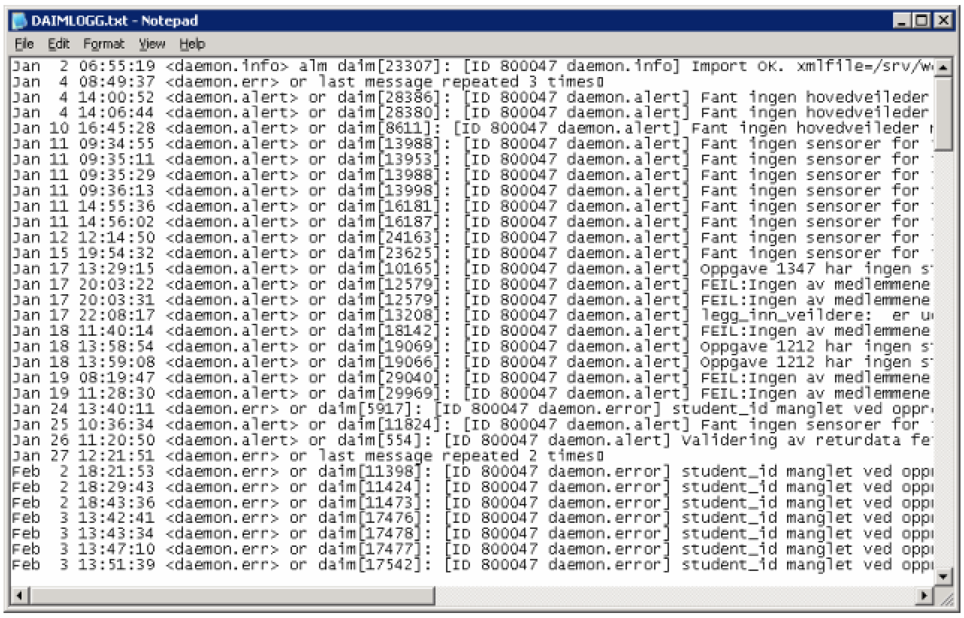
In general our task has been divided into several parts where one of the parts involved analysis of historical data. We received several fault reports from DAIM which we were supposed to classify and analyze in the same way as we did in our previous work [5] but with minor changes. Section 4.6 describes the classification scheme that we used. In Section 4.5.1 we will take a closer look at the fault reports that we received from DAIM. We were also supposed to compare the results from the fault analysis from our previous work with the results from the fault analysis of DAIM to see if the fault distributions differed. The next part of our task was to use a fault analysis technique named PHA on the DAIM system. The PHA technique, which is described in Section 2.6.5, is a variant of hazard analysis and we have used this technique with respect to several use cases given in Figure 3.3. The final part of our task consisted in comparing and analyzing the results from the PHA process and the analysis of fault reports from DAIM to see how the methods worked and hopefully point out possible connections between the methods. In connection with the comparison we thought it might be useful to classify the different hazards making it easier to compare corresponding hazards and fault reports. Therefore we used the same classification scheme, see Section 4.6, concerning the classification of hazards as we did with the classification of fault reports. There were some challenges related to the classification of hazards according to this classification scheme and Section 4.7.3 will be devoted to this.

4.5 Data Collection

Considering that we received data from different sources, we found it appropriate to assign Section 4.5.1 for the collection of fault reports and Section 4.5.2 for the collection of data related to the preliminary hazard analysis.

4.5.1 Fault Reports

We received two different kinds of collections of fault reports from the system developers connected to the DAIM system. The first collection which is illustrated in Figure 4.1, is a system log that we received in ".txt" format. As one can see from the illustration this system log is computer generated and does not contain too many details. The second collection of fault reports that we received, illustrated in Figure 4.2, included more information concerning each fault instance, but the numbers of fault instances were actually quite few. This collection was received in the format ".doc". We then converted the ".doc" and ".txt" files into the format used in Microsoft Excel, namely ".xls", because of the possibilities that Microsoft Excel provides concerning statistical analysis. After the conversion we put each collection into



```
Jan  2 06:55:19 <daemon.info> alm daim[23307]: [ID 800047 daemon.info] Import OK. xmlfile=/srv/w
Jan  4 08:49:37 <daemon.err> or last message repeated 3 times
Jan  4 14:00:52 <daemon.alert> or daim[28386]: [ID 800047 daemon.alert] Fant ingen hovedveileder
Jan  4 14:06:44 <daemon.alert> or daim[28380]: [ID 800047 daemon.alert] Fant ingen hovedveileder
Jan 10 16:45:28 <daemon.alert> or daim[8611]: [ID 800047 daemon.alert] Fant ingen hovedveileder
Jan 11 09:34:55 <daemon.alert> or daim[13988]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 11 09:35:11 <daemon.alert> or daim[13953]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 11 09:35:29 <daemon.alert> or daim[13988]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 11 09:36:13 <daemon.alert> or daim[13998]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 11 14:55:36 <daemon.alert> or daim[16181]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 11 14:56:02 <daemon.alert> or daim[16187]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 12 12:14:50 <daemon.alert> or daim[24163]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 15 19:54:32 <daemon.alert> or daim[23625]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 17 13:29:15 <daemon.alert> or daim[10165]: [ID 800047 daemon.alert] Oppgave 1347 har ingen s
Jan 17 20:03:22 <daemon.alert> or daim[12579]: [ID 800047 daemon.alert] FEIL:Ingen av medlemmene
Jan 17 20:03:31 <daemon.alert> or daim[12579]: [ID 800047 daemon.alert] FEIL:Ingen av medlemmene
Jan 17 22:08:17 <daemon.alert> or daim[13208]: [ID 800047 daemon.alert] legg_inn_veildere:  er u
Jan 18 11:40:14 <daemon.alert> or daim[18142]: [ID 800047 daemon.alert] FEIL:Ingen av medlemmene
Jan 18 13:58:54 <daemon.alert> or daim[19069]: [ID 800047 daemon.alert] Oppgave 1212 har ingen s
Jan 18 13:59:08 <daemon.alert> or daim[19066]: [ID 800047 daemon.alert] Oppgave 1212 har ingen s
Jan 19 08:19:47 <daemon.alert> or daim[29040]: [ID 800047 daemon.alert] FEIL:Ingen av medlemmene
Jan 19 11:28:30 <daemon.alert> or daim[29969]: [ID 800047 daemon.alert] FEIL:Ingen av medlemmene
Jan 24 13:40:11 <daemon.err> or daim[5917]: [ID 800047 daemon.error] student_id manglet ved oppr
Jan 25 10:36:34 <daemon.alert> or daim[11824]: [ID 800047 daemon.alert] Fant ingen sensorer for
Jan 26 11:20:50 <daemon.alert> or daim[554]: [ID 800047 daemon.alert] Validering av returdata fe
Jan 27 12:21:51 <daemon.err> or last message repeated 2 times
Feb  2 18:21:53 <daemon.err> or daim[11398]: [ID 800047 daemon.error] student_id manglet ved opp
Feb  2 18:29:43 <daemon.err> or daim[11424]: [ID 800047 daemon.error] student_id manglet ved opp
Feb  2 18:43:36 <daemon.err> or daim[11473]: [ID 800047 daemon.error] student_id manglet ved opp
Feb  3 13:42:41 <daemon.err> or daim[17476]: [ID 800047 daemon.error] student_id manglet ved opp
Feb  3 13:43:34 <daemon.err> or daim[17478]: [ID 800047 daemon.error] student_id manglet ved opp
Feb  3 13:47:10 <daemon.err> or daim[17477]: [ID 800047 daemon.error] student_id manglet ved opp
Feb  3 13:51:39 <daemon.err> or daim[17542]: [ID 800047 daemon.error] student_id manglet ved opp
```

Figure 4.1: An excerpt from the DAIM system log.

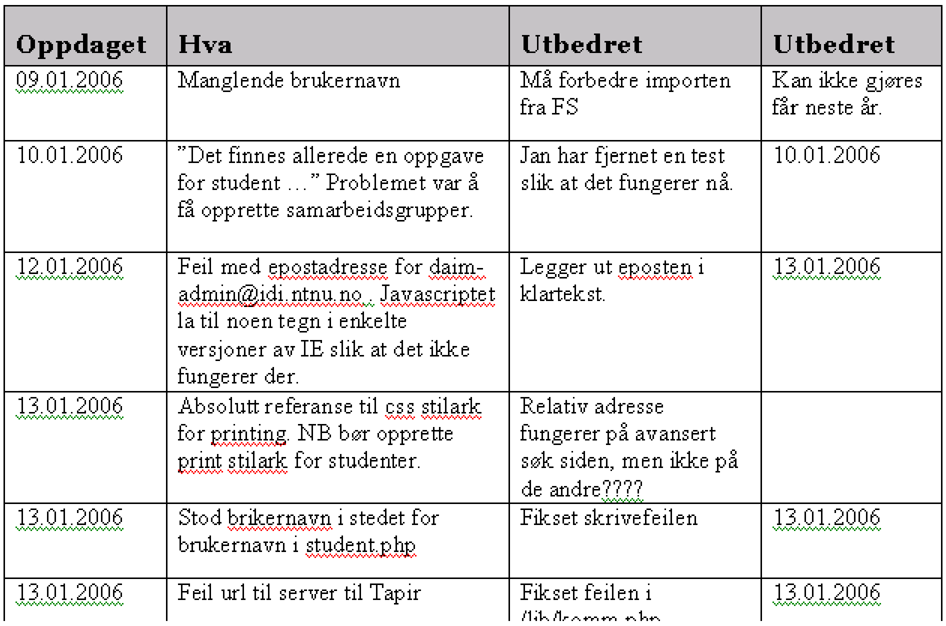
a separate spreadsheet. Each row in the spreadsheets contained one fault report and we made a column for each fault report where we categorized the fault report according to the fault types we were supposed to use. The fault types that we used in the classification process is given in Section 4.6.1.

4.5.2 Preliminary Hazard Analysis

In Section 5.3 we describe how the sessions of PHA were performed. From these sessions we made a lot of notes and hazard tables which had to be reviewed and converted into a digital version. As can be seen from the hazard tables in Chapter 5, we also added a column that was used in the classification of the different hazards as an identifier for the different hazards identified. As the PHA technique normally is performed in the early stages of the system development, we based our approach of the PHA on the several use cases in Figure 3.3 which reside from the design stage in the development of DAIM.

4.6 Classification of Faults and Hazards

We have chosen to use the same classification scheme in this work as we did in our previous work [5] namely a modified ODC scheme, see Section 2.4.1. The original ODC scheme was developed in 1992 [4] and has a suitable number of fault categories which makes it easier to work with compared to



Oppdaget	Hva	Utbedret	Utbedret
09.01.2006	Manglende brukernavn	Må forbedre importen fra FS	Kan ikke gjøres før neste år.
10.01.2006	"Det finnes allerede en oppgave for student ..." Problemet var å få opprette samarbeidsgrupper.	Jan har fjernet en test slik at det fungerer nå.	10.01.2006
12.01.2006	Feil med epostadresse for daim-admin@idi.ntnu.no . Javascriptet la til noen tegn i enkelte versjoner av IE slik at det ikke fungerer der.	Legger ut eposten i klartekst.	13.01.2006
13.01.2006	Absolutt referanse til css stilark for printing. NB bør opprette print stilark for studenter.	Relativ adresse fungerer på avansert søk siden, men ikke på de andre????	
13.01.2006	Stod brikernavn i stedet for brukernavn i student.php	Fikset skrivefeilen	13.01.2006
13.01.2006	Feil url til server til Tapir	Fikset feilen i	13.01.2006

Figure 4.2: An excerpt from the manually made DAIM fault reports.

other classification schemes such as the Hewlett-Packard scheme or the IEEE 1044-1993 Anomaly Classification scheme. Considering that we had good experience with the ODC scheme in our previous work we wanted to continue using the ODC scheme, but with smaller modifications. In Section 4.6.1 we will present the modified ODC fault categories used in the classification process.

4.6.1 The Modified ODC Categories

This section will introduce the fault categories that we have used in the classification process and they are based on [4] and [8]. The ODC based fault categories that we will present below coincide with the fault categories that we used in our previous work [5] and Table 4.1 summarizes the fault types. In [7] the researchers also added additional fault types, because they felt it was necessary and in the classification scheme that we will use, we have also added new fault types. In Table 4.1 the additional fault types are marked with a "*".

Function A function error is one that effects significant capability, end-user interfaces, product interface, interface with hardware architecture, or global datastructure(s) and should require a formal design change. Examples:

1. A function is missing when doing a specific operation.

Modified ODC fault types
Function
Assignment
Interface
Checking
Timing/serialization
Relationship *
Documentation
Algorithm
Data *
GUI *
Environment *
Duplicate *
Not fault *
Unknown *

Table 4.1: Modified fault types in ODC.

2. A function produces unexpected and wrong results.

Assignment An assignment error indicates a few lines of code, such as the initialization of control blocks or datastructures. Value(s) assigned incorrectly or not assigned at all; but note that a fix involving multiple assignment corrections may be of type algorithm. Examples:

1. Internal variable or variable within a control block did not have correct value, or did not have any value at all.
2. Initialization of parameters.
3. Resetting a variable's value.
4. The instance variable capturing a characteristic of an object (e.g. the color of a car) is omitted.
5. The instance variables that capture the state of an object are not correctly initialized.

Interface Interface corresponds to errors in interacting with other components, modules or device drivers via macros, call statements, control blocks or parameter lists. Examples:

1. A database implements both insertion and deletion functions, but the deletion interface was not made callable.
2. The OO-message incorrectly specifies the name of a service.
3. The number and/or types of parameters of the OO-message do not conform with the signature of the requested service.

Checking Checking addresses program logic which has failed to properly validate data and values before they are used. Examples:

1. Value greater than 100 is not valid, but the check to make sure that the value was less than 100 was missing.
2. The conditional loop should have stopped on the ninth iteration. But it kept looping while the counter was less or equal than 100.

Timing/Serialization Timing/Serialization errors are those which are corrected by improved management of shared and real-time resources. Examples:

1. Serialization is missing when making updates to a shared control block.

Relationship Relationship describe errors that occur due to mistakes in library systems, management of changes, or version control.

Documentation Documentation errors can affect both publications and maintenance notes. Examples:

1. When the implementation is done correctly according to the design document derived from the requirements specification, but it turns out that there is something wrong with the design document meaning that the requirements specification is misunderstood or wrong.

Algorithm Algorithm errors include efficiency or correctness problems that affect the task and can be fixed by (re)implementing an algorithm or local datastructure without the need for requesting a design change. Examples:

1. The low-level design called for the use of an algorithm that improves throughput over link by delaying transmission of some messages, but the implementation transmitted all messages as soon as they arrived. The algorithm that delayed transmission was missing.
2. The algorithm for searching a chain of control blocks was corrected to use a linear-linked list instead of a circular-linked list.

Data Data used in the system is wrong or incorrect. Examples:

1. Missing information in reports generated, such as missing page numbers, duplicated data, such as a string or number repeated two times make up a data fault.

GUI This type of fault is connected to faults regarding the visual appearance. Examples:

1. If a button on a web page contains the name "Save", but is supposed to be named "Next" etc., this qualifies as a GUI fault.
2. The font type and size is wrong.

Environment Environment faults may include faults that occurs in back-end systems controlled by others. Examples:

1. The server did not respond because it had crashed. This resulted in that the system could not get the needed data and was not able to do its tasks.

Duplicate A duplicate error is an error that is reported earlier.

Not fault Sometimes faults may be reported, but due to certain circumstances, they do not qualify for being a fault. Examples:

1. When the test data is incorrect.
2. When something is done in compliance with the requirement specification and is totally correct, but the customer changes his/her mind and wants a change of some kind.
3. When something is reported as a fault, but the fault can not be recreated.

Unknown This category is applied to a fault that is difficult to categorize. Examples:

1. Faults with little or no information is difficult to categorize.

4.7 Challenges in Our Work

We met several challenges in our work and Section 4.7.1 to Section 4.7.3 describe some of the challenges we were forced to deal with.

4.7.1 Little Experience with PHA

We did not have much knowledge about PHA before we started our work and therefore we had to spend some time learning this technique properly. Considering that PHA is a technique that requires some degree of creativity, it may take some time to learn it properly and one has to go back and forth in the process because one may discover new things at later stages and previous work has to be updated.

4.7.2 Inconsistency in the Way Faults are Reported

The fault reports that we received were of two different kinds; one of the collections of fault reports were generated automatically from the system (system log) while the other collection was human made. There were several differences in the way these collections presented the different fault reports. The auto-generated fault reports were shorter and more ”cryptic” and they were harder to understand while the descriptions from the human made fault reports were easier to understand because they were longer and explained the faults in a better way. Luckily we got a lot of help from our contact person in DAIM when we needed it and this prevented us from guessing.

4.7.3 Classification of the Hazards According to Classification Scheme

We met some challenges when we classified the different hazards according to the chosen classification scheme, see Section 4.6.1. In contrast to the fault reports the hazards were described less detailed and were of a more general nature and this made it difficult for us to classify some of the hazards. On the other side some of the hazards contained elements from several fault types and we ended up classifying some of the hazards as belonging to several fault types.

4.8 Summary

In this chapter we have introduced the context of our work in addition to the research questions that we are about to answer in later chapters. The research method and process has also be been given some attention while a section has been devoted to the challenges we were forced to deal with. The classification scheme that we will use throughout this master's thesis has also been presented.

Chapter 5

Hazard Analysis

This chapter contains the results from the hazard analysis that we have performed in our work. The reasons for using the particular method that we have used will also be given in addition to some information regarding the process.

5.1 Introduction

In this chapter we will present the work that we have done concerning the hazard analysis of the DAIM system. Section 5.2 gives the reasons why we ended up using the hazard analysis method PHA and Section 5.3 describes how we carried out the analysis. The results from the analysis will be presented in Section 5.4.

5.2 Our Choice of Hazard Analysis Technique

Our choice of hazard analysis technique fell on PHA and there were several reasons for this. Concerning FMEA this technique was performed on DAIM last year in the master's thesis [11] and therefore it was no point of doing this analysis again. Considering that we received documentation from the design phase of DAIM we had to choose a technique that was best suited for analysis early in the system development. By looking at Table 2.5 we see that FTA normally is used later in the development process and since we are supposed to analyze design documentation this excludes this option. Then we were left with three options: HazOP, ETA or PHA. The criteria "System Info Requirement" says something about how well one should know the system before doing the analysis and from Table 2.5 it is stated that if one wants to use the ETA technique one needs a high degree of knowledge to the system before one can do the analysis and we did not have much knowledge to DAIM before we started. It would probably have taken too long time to obtain this knowledge of DAIM considering that this master's thesis is limited to a time span of 20 weeks. The technique HazOP has "moderate" assigned to almost every criteria and considering that the "process stage" has medium assigned while PHA has "early" assigned to it in addition to that the knowledge required for doing an HazOP analysis is higher than with the PHA, we ended up with PHA. It shall also be stated that PHA is the most light-weighted safety analysis method among all the methods mentioned and since the application that we are about to analyze is not a safety-critical application, this may justify our choice even more.

5.3 PHA Sessions

During our work we had several PHA sessions where we analyzed the different hazards connected to the DAIM system. The number of participants was not constant, but three of the participants were present at every session. The group consisted of one project leader, four research fellows, one of the system developers from the DAIM system and a master degree student. The professor was the session leader and was in charge of the meetings by using the blackboard and making questions which stimulated the participants to discuss what hazards that could occur. After having discussed several possible hazards we came up with hazards that everyone agreed on and the session leader wrote the different suggestions on the blackboard. The schema that we used when analyzing the hazards has the column headers described as:

Hazard This term describes the hazard or treat that have pointed out to be a potential problem.

Cause Here we try to find reasonable causes dependant on the hazard identified.

Consequence We try to figure out what the consequences of the hazard are.

Action Here we make suggestions about what can be done to prevent the hazards from happening.

H This column is used as an identifier for the different hazards. Considering that a hazard can have several causes attached to it, we decided to split the hazards with respect to this such that for example the hazards H1, H2 and H3 actually come from the same hazard, but they have been split into three hazards.

We based our PHA sessions on the use case diagram in Figure 3.3 describing the DAIM system, where we treated each actor and their use cases subsequently. The Tables from Table 5.1 to Table 5.8 describe the use cases

Hazard	**Cause**	**Consequence**	**Action**	**H**
Unauthorized access	Illegal username in DB	Destroyed data	System feedback	H1
	User is missing because of gap in export from FS	Cannot do what she/he wants	Manual control routines	H2
Cannot get access	Too strict network policy	Cannot do what he/she wants	Use another login system (increase complexity)	H3

Table 5.1: Hazard table for the use case ”Log in” which is common for all users except from the ”external user”.

of each actor and each actor has got its own table. One use case, ”Log in”, was common for almost all actors and this use case is placed in its own table, namely Table 5.1.

5.4 Results from the PHA Sessions

In this section we will present our findings from the PHA sessions. Each actor in the use case diagram will be given a subsection where we include the corresponding hazard analysis matrix for that actor and some comments concerning the matrix. As said in Section 5.3 the use case ”Log in” was common for almost all actors and we will place this use case in a separate subsection. Our results will be given from Section 5.4.1 to Section 5.4.8.

5.4.1 ”Log in” Use Case

This use case is common for the actors **Student**, **Administrator**, **Economy**, **Supervisor** and **System administrator** while the actor **External user** does not have login functionality. Table 5.1 shows the use case ”Log in”. The hazards related to the ”Log in” use case are several and the most critical one is related to unauthorized access. If a user logs on to the system with administrator permissions and the user is not supposed to have administrator permissions this can result in unfortunate situations. Data can be destroyed or changed which will reduce the integrity of the system. The causes for this may be that someone has hacked into the database or that an System administrator erroneous has given a user administrator rights. As a precaution mechanism the system should give some kind of system feedback. The idea here was that the user who was logged in should have the possibility to see his/her username somewhere on the web page such that the user would notice if he/she was logged in with the wrong username. This feedback would of course not help in the situation where a hacker was involved, but here it might be a solution to have some kind of a logging functionality where every action done by the administrator users are logged with for example date, IP addresses and corresponding actions such that it might be possible to rollback the actions performed by an illegal administrator. If a hacker is involved the IP address might be a good start to find out who is behind the illegal actions and maybe deny incoming requests from this IP address for a start etc. Dependant on the criticality of the system, actions have to be done in accordance to this, meaning that this system is not that critical as a banking system etc. and one may not want to use too much resources on finding the hacker, but one want to make sure that this kind of hacking does not occur again by improving the security in the system.

If an authorized administrator does not have access to the system, this is classified as a hazard. The reason for this can be that the user is missing in the database because of a gap in the export from FS (Felles Studentsystem). FS is a system taking care of study administrative data concerning the students at NTNU [1]. Another reason can be that the local user's network policy is too strict especially when for example logging in from a remote location because in some networks the policy is that redirecting is not legal such as the case is with the DAIM system where DAIM is redirecting to another system for retrieving necessary information and this will then cause trouble for all actors that need to log on to the system. The consequences of the hazard will be that a correctly authorized administrator would not get access to the system and he/she would not be able to do what he/she wants to do. A solution to this problem could be to use another login system, but this may also lead to higher complexity which might not be very desirable.

5.4.2 Actor: Administrator

The hazard analysis table for the **Administrator** actor is shown in Table 5.2. One of the hazards that we identified was that of choosing wrong external examiner from the list of external examiners when assigning a external examiner to a master's thesis. The reason can be that the administrator incorrectly chooses a wrong external examiner for a specific master thesis because of human error. If for example there is a list with many external examiners and two external examiners have got the same surnames, there is a chance for choosing wrong external examiner if one is not focused enough (human mistake). The result will be that wrong external examiner is assigned for wrong master's thesis which is unfortunate. An action for reducing the possibility of this to happen can be that the actor performing the wrong action gets some kind of feedback from the system such as a confirmation question; ex. "Is this information correct? Yes or no". If the damage has already happened it should be possible to change the data in retrospect. Another solution can be to add limitations and logic to the system such that external examiners only are allowed to be assigned to master's theses in a certain field of interest and by doing this one will reduce the

[1] http://www.ntnu.no/sa/sfs/studsys/fellessys 05.05.07: 15:25

Hazard	**Cause**	**Consequence**	**Action**	**H**
Choose wrong external examiner from list	Human error, ”miss” the right choice	Wrong result (external examiner)	System feedback External examiner theme info Possibility for changing the external examiner.	H4
Ignore deficiency or inconsistence	Human error	Save mistakes or inconsistent data	Check the content: Spell check Control of consistency (dates)	H5
Missing approval	Human error	Cannot deliver	Manual control ”Timeout” (E-mail to the student/admin)	H6
Cannot find contract/thesis	Severely delayed Missing info Wrong search criteria	Missing approval	Change proposal, search if we find 0. Help to the search criteria	H7
Person with wrong permissions/Aliasing Overwrite user	Creative reuse ”Inherit”	Wrong data Destroyed data	Training System feedback Split functionality (new/change)	H8
User with wrong permissions Role with wrong permissions	......	......	......	......
Generation of inconsequent reports [borderline between system fault and programming fault]	Inconsistent SQL use (criteria)	Wrong/Misleading info that not necessarily will be detected	System test? Gather similar SQL-querries	H9
Wrong update of sencor/teacher	Creative reuse	Destroyed data		H10

Table 5.2: Hazard table for the actor”Administrator user”

possibility of choosing a ”completely wrong” external examiner for a master’s thesis. Making such a solution will increase the complexity of the system, but it maybe will be worth doing it if this is a huge problem. In the example above with the two external examiners with the same surnames, one of the external examiners may be available for master’s theses in the field ”System development” while the other external examiner is only allowed to censor master’s theses in the area of ”digital image processing”. Then if wrong external examiner was chosen for the wrong master’s thesis, the system would have noticed this and could give the actor feedback and ask him/her to correct the information. The solution just discussed may be ideal in theory, but difficult to implement in practice.

Another hazard that we found during the sessions was the one related to displaying master’s theses. A big problem would be if the master thesis resides in the system, but the Administrator cannot find it.

If a master contract is missing approval from the supervisor, this can can lead to problems because there is strict policies regarding deadlines in connection with master’s theses. The reason why the master contract may be missing approval could be human error such as the supervisor has forgotten it or is not available because of illness etc. The consequence will be that the student will not get to deliver his/her master’s thesis and may be delayed half a year or worse. A way to avoid this to happen is to introduce manual controls where a person is dedicated to go through the list of students and check their status concerning approval of master’s thesis. If the time is running out, the person dedicated to do the manual control has to contact the student or supervisor to inform them that the master’s thesis needs to be approved soon. Today one person is assigned to this task and it has worked fine so far. A suggestion for improvement can be to include information concerning the status of the supervisors. If one knows that a supervisor will stay a year abroad or for some reason will be away for a period of time, it may be a possibility to include functionality regarding this into the system. Another possibility is to include some kind of a function to the system, where the system itself checks the date and the list of master’s theses that are not approved and then if approvals are missing generate a reminder email to the professors, students, administrators etc. saying that the master’s theses have to be approved soon.

Another problem can be if the administrator does not find the master contract because of a critical delay or that the contract is incomplete and therefore has not been approved and has to be returned to the student for review. Another reason can be wrong search criteria such as the parameter ”date” is wrong. This will result in missing approval which can be very unfortunate for the master student. A solution to this might be a useful search function if the administrator actor does not find anything. Further it might be an idea to have some kind of a help description for the search function describing all the parameters the system supports for improving the search functionality and one might get the desired results.

The next hazard we will present is related to the process of creating and updating a user/role account. If an administrator by mistake gives wrong permissions to the wrong people this can lead to several problems. On one side if users with malicious intentions get too many rights they can get access to data that they were not supposed to have and they can manipulate data such as unauthorized deletion or modification. This will reduce the integrity of the system and can have serious consequences. On the other side users that get too few rights will not be able to do their work because of this. This hazard is mainly caused by human error and it is difficult to remove. One suggestion of a preventive action can be to have some kind of review done by another administrator. Then at least two persons have controlled the information and it might be safer. This solution may demand for more work from the administrators and it is not given that they have got time for this. There has to be a tradeoff between the safety level needed and cost. It might show that the risk for this to happen is so small that if it actually happens it would be worth correcting it later.

One hazard that we identified was related to creating and/or updating a user account. If a user is selected from the list, the information connected to the user will appear in text fields (editable) below the list. If the Administrator actor does some changes in these fields, the information concerning the user chosen in the list will be changed. The threat that we identified was if the Administrator wants to create a new user and erases the information of another user resulting in that the information related to the first user is overwritten. On way to avoid this could be to provide system feedback saying that for example ”You are about to change the name of user X. Do you want to continue?” and give the Administrator actor properly training. Another way to avoid this hazard is to split the functionality concerning creating/updating user such that creation of a new user is performed in one place while updating is performed in another place.

We identified a hazard related to the generation of inconsequent reports because some reports in the DAIM system are made in advance. If someone makes a query that is a little different from ”normal” this may result in wrong and misleading information when the information is retrieved. One easy example describing this could be to make a query ”NAME DATE” which might give a different result than the query ”DATE NAME”. This kind of error is related to databases and from SQL one knows that a join function between two tables can lead to different results dependant on the order the tables are joined in; A x B may give a different result than B x A. A way to prevent this may be to gather all SQL queries in one place and make them as similar as possible. It may also be reasonable to make some scripts that handle this kind of problems meaning that in the case where a table A should be joined with table B, the script would instruct the computer to join A x B in this order and not vice versa. It should be said that this problem may be seen as a coding error and the only way to avoid this problem from happening is by testing it thoroughly like with any other coding errors.

5.4.3 Actor: System Administrator

The hazard analysis table for the **System Administrator** actor is shown in Table 5.3. The two actors **System Administrator** and **Administrator** have several use cases in common, but in the following paragraph we will only present the use cases that are exclusive for the System Administrator actor. We will underline that all the use cases and corresponding hazards mentioned in Section 5.4.2 and Table 5.2 for the Administrator actor also applies for the System Administrator.

The actor System Administrator has one use case named ”Import data” which we found interesting considering the possibility of potential hazards. From Table 5.3 we have identified the hazards ”Wrong data from FS” and ”Inconsequent data” which may occur when importing data from the FS system. One cause that we have found concerns inconsequent data that DAIM receives from the FS system meaning that data might be registered in a wrong format. When the DAIM system receives the data it will produce an error message saying that something went wrong with the validation of the data. The line of code that contains the error will not be executed and the entire file has to be corrected and executed again. For preventing this to happen it might be necessary to register the data manually, but this will be less effective. Another possibility is to look at the insertion routines in FS and try to change them if it is possible. The last action to perform that we can think of is the possibility to ”clean data”, making a script that removes the unwanted data, but this can be difficult to do. One possibility could be to first validate the data, then insert it. Today all these operations are performed as a whole.

There is always a risk that wrong information will be inserted into the system because of human error. The result will be that the system contains wrong data which may lead to problems and extra work. One suggestion for avoiding this can be to involve another person to review the changes/insertions done by the System Administrator actor such that the data at least is controlled by two persons.

5.4.4 Actor: Supervisor

The hazard analysis table for the actor **Supervisor** is given in Table 5.4. From Figure 3.3 it can easily be seen that the use case ”Choose censor” for the actor Supervisor also is present for the actors System Administrator and Administrator and we will therefore refer to Section 5.4.2 concerning the hazards connected to this use case. The second use case related to the Supervisor actor is ”Log In” which is thoroughly investigated in Section 5.4.1.

Hazard	Cause	Consequence	Action	H
Choose wrong external examiner from list	Human error, "miss" the right choice	Wrong result (external examiner)	System feedback External examiner theme info Possibility for changing the external examiner	
Ignore deficiency or inconsistence	Human error	Save mistakes or inconsistent data	Check the content: Spell check Control of consistency (dates)	
Missing approval	Human error	Cannot deliver	Manual control "Timeout" (E-mail to the student/admin)	
Cannot find contract/thesis	Strongly delayed Missing info Wrong search criteria	Missing approval	Change proposal, search if we find 0. Help to the search criteria	
Person with wrong permissions/Aliasing Overwrite user	Creative reuse "Inherit"	Wrong data Destroyed data	Training System feedback Split functionality (new/change)	
User with wrong permissions Role with wrong permissions	......	......	......	
Generating inconsequent reports [borderline between system fault and programming fault]	Inconsistent use of SQL (criteria)	Wrong/misleading info that not necessarily is discovered.	System test? Gather similar SQL-queries	
Wrong update of external examiner/professor	Creative reuse	Destroyed data		
"Wrong" data from FS Inconsequent data	Inconsequent data input Fault registered Wrong format	System gives error message The code line is not executed; have to execute the file again.	Manual registering Change "inserting" routines in FS "Clean data"	H11
Wrong user permissions given	Manual routine	Permissions to wrong person, safety threat	insert one and one Possibilites for removing permissions	H12
Wrong information inserted	Manual fault	Wrong data	External validation source	H13

Table 5.3: Hazard table for the actor "System administrator user"

Hazard	Cause	Consequence	Action	H
Choose wrong external examiner from list	Human error, "miss" the right choice	Wrong result (external examiner)	System feedback External examiner theme info Possibility for changing the external examiner	

Table 5.4: Hazard table for the actor "Supervisor user"

Hazard	**Cause**	**Consequence**	**Action**	**H**
Wrong registration	Wrong registration and missing undo possibilities	Wrong presentation of information Censor not being paid	Undo possibilities	H14

Table 5.5: Hazard table for the actor ”Economy user”

5.4.5 Actor: Economy

The hazard analysis table for the actor **Economy** can be found in Table 5.5. We will refer to Section 5.4.1 regarding the hazards connected to the use case ”Log in”. Besides from the use case ”Log in”, the actor Economy has got one use case named ”Handle payments” which introduces a hazard. There is a risk that the Economy actor makes a mistake when registering the different payments. The consequences from this hazard may be that the system will contain incorrect information and the external examiners may not receive the payment they are entitled to or they may receive a lesser amount. Unless they themselves discover the error it will probably never be corrected and the external examiners will suffer from this. We found that this may happen because of human error done by the Economy actor and that the system does not provide some kind of ”undo” functionality. An action for preventing this situation may be that the system should provide some kind of a ”undo” functionality which helps the Economy actor to correct the problem.

5.4.6 Actor: Student

The hazard analysis table for the actor **Student** can be found in Table 5.6. As previous stated the ”Log in” use case is common among all actors except from the actor External user and we will therefore refer to Section 5.4.1 regarding the hazards connected to that use case.

One hazard that we found with respect to the use case ”Fill in contract” concerns the scenario where the Student actor inserts incorrect values in the master contract. This may happen because of sloppiness from the student actor. Another reason may be that the slots where the user is supposed to insert data does not properly explain what and how the data should be inserted. Simple examples can be uncertainties concerning the date format etc. The consequences will be that wrong information will reside in the system which can cause problems for the parties involved. In the example where we have the date ”020407” it can be confusing to know how this date shall be interpreted if the date format is not specified explicitly. Based on the date it can be of both the formats 'ddmmyy' or 'yymmdd'. An action for preventing this to happen is to explicitly give an example close to where the actor is supposed to insert his/her dates such as ”The date format should be of the format ddmmyy”. In general the system should provide the user with the possibility to read through all the data that the user has inserted

Hazard	**Cause**	**Consequence**	**Action**	**H**
Wrong value in contract	Sloppiness MMI Touch wrong key on keyboard	Wrong info	Read through and approve info Rules of consistency	H15
-datafault / typefault	Validation fault Characterset fault	Wrong data in DB	Rules of consistency	H16
Missing input	Failed to see the field	Missing data	Rules of consistency Check that the fields are correctly filled in	H17
Different ways to type the name of a person	Not pay attention	Wrong info	Use of usernames not allowed to change names	H18
Wrong info in contract - change name fault - email - phonenumber - study program - branch of study	Unaware Misunderstanding	Wrong info	Get as much as possible info from FS Shows contract for approval	H19
	Interface, programming error (assignement)		Testing	H20
Cooperation contract fault	Wrong input sloppiness	Wrong info	Check agains study program, usernames etc. Check another group belonging	H21
Mess in the group composition	People have difficulties when deciding	Inconsistency within the DB appliacation hangs	”Correct sequence in DB-access/writing” Usermanual Check groups where number of members = 1	H22
Change in contract after having signed		Wrong info	Only administration has access Status check on files	H23
Cannot deliver master’s thesis	Wrong file type		Check in system (php)	H24
	Wrong version/file	Wrong master’s thesis	User has to take care of oneself	H25
	Wrong pdf-type	print shop problems	User has to take care of oneself, print shop make contact (phone)	H26
Cannot deliver corrected master’s thesis	Rules		The administration needs to fix it	H27
One person in the group delivers master’s thesis without consent	Different views in the group	Wrong delivery	The administration needs to fix it Function that	H28

Table 5.6: Hazard table for the actor ”Student user”

and give him/her a question such as ”Is this information correct” where the user has to respond with ”confirmed” or ”cancel” etc.

Another hazard related to the use case ”Fill in contract” is if one or more fields in the contract have been left out. The reason may be that the actor has overlooked these fields or did not know what to write. This scenario will result in missing data. An action for preventing the absence of important data can be to implement functions that check if some fields are not filled with sufficient information and make the user aware of this. In the situation where the user fails to fill out some required fields in the schema, a help text should be provided by the system such that the user at least get a clue of what is expected to be registered. Today a solution exist and the user receives a message at the bottom of the web page stating that some fields are not correctly filled, but according to one of the developers of the DAIM system, new and improved ways of informing the user of this have been thought of. A solution could be to put a color on the text fields that are not filled in correctly making it more visible.

We have identified a hazard related to the inconsequent typing of person names. In computer systems where a database is used to keep track of people it is not unusual that users are registered more than one time because of inconsequence in the way users write their names. One record may contain the name, while other records contain variances of the same name for the same person. As examples the middle name may not be present in a record such that it seems that there are two persons. Someone uses a hyphen in the name, but sometimes this hyphen may be forgotten for some reason etc. The cause that we found is based on inattentiveness from the user. The consequences can be that there exist wrong information in the system, a person can be named differently such that it seems that it is two persons instead of one. As a barrier one can use the usernames that the actors use for logging into the system such that the system ”translate” the username into the original name.

Another important hazard concerns wrong information in the master contract such as wrong name, email, telephone number, programme of study or wrong branch of study. This can be caused by inattentiveness from the actor or misunderstandings. The consequence will be that the system contain incorrect information. A way to prevent this hazard from happening can be to get as much as possible of the information concerning the student actor from the FS system. This system is of administrative character having administrative data for students and their studies and importing data from this system may reduce the possibilities for errors connected to wrong information regarding the students.

If a master's thesis is carried out by more than one student, this has to be taken into account in the master contract and this may lead to a hazard. The hazard may appear because the student actor introduce incorrect data because of sloppiness. This results in incorrect information in the system and extra work has to be done to fix the errors. As a way to reduce the possibility of the hazard to happen the system should provide functions that check the programme of study and the username of the actor. The system should also include program logic putting restrictions such that a person only can belong to one cooperation group. If the person wants to change his/her cooperation group, then the person has to resign from the group which he/she is related to for the moment and after this join the new group. It is important that the user does this in the correct order just mentioned.

Another hazard that we found in connection to cooperation groups was complications in the composition of the groups. Sometimes people may have difficulties when deciding to work in a group or not and that there can be changes in the composition of the groups because people change their minds from time to time. This may result in inconsistent databases and the system may be incorrectly updated. An action to prevent hazards in group relations can first of all be to make a guide or manual stating how to correctly make changes concerning the composition of the groups. When there are several changes concerning the composition of the groups it is important to handle the writing and accessing sequences in the database correctly. In addition there should be implemented at function that checks the groups where "no_of_members < 2" etc.

If the master contract needs to be changed after a period of time, such as a change in the problem description etc., this may lead to a potential problem or an hazard. The causes for this may be that there have been some changes in the description of the master's thesis for some reasons. The consequences may be that wrong information will reside in the system and for preventing problems of this kind it may be wise to only let the administrator actor in agreement with the supervisor, have privileges for changing information such as the problem description etc. Another way for preventing this hazard to happen is to implement a status check on the files such that one can see when and what is changed etc. and then it is easier to "roll-back" the changes.

When the time has come to deliver the master's thesis we have identified one hazard which may occur. According to Figure 3.3 we have a use case "Deliver master thesis" which may result in the situation that the student actor may not get to deliver the master's thesis of various reasons. One reason could be that the file type submitted is in a wrong format and this will result in the situation where the master's thesis does not get delivered in the worst case. Today the system is implemented in such way that if this hazard would happen, the user will be informed on the web page giving the actor another attempt in trying to upload the file. The system should have implemented a check function that makes sure that the user gets an error message with an explanation of what is wrong such that the user can correct this himself/herself. The student actor has a responsibility to make sure that he/she delivers the correct version/file of the master's thesis. The consequences of delivering for example a previous version of the master's thesis will be that wrong master's thesis is delivered and if it is an earlier version it may result in a poor grade. As a preventive action the system should remind the user of this, but the responsibility lies on the student actor. Another reason why the master's thesis might not be delivered is because of wrong pdf-type. This may lead to problems for the print shop, but the print shop will try to get in touch with the student and inform him/her of the problem giving the student a second chance to solve the problem.

We identified a hazard related to delivering the master's thesis in a co-operation group and this hazard may occur when a student actor in the co-operation group delivers the master's thesis without consent from the other student actors in the group. The reasons may be that there are different opinions in the group or misunderstandings of different kinds. The result will be that the delivery is wrong and if this happens the administration has to be contacted for making a solution to this problem. A solution could be to implement a functionality that makes sure that a person in a cooperation group does not get deliver the master's thesis without consent from the other group members. When a member of a cooperation group submits the master's thesis, a function should be implemented such that an an email is sent to the other members of the group with a request saying that they have to log on to the DAIM system and accept or deny the submission. In the situation where one of the group members does not have the ability to respond to this request, it might also be useful to implement a function that gives the actor the possibility to confirm this in advance, said in other words an auto-confirm possibility.

5.4.7 Actor: External user

The hazard analysis table for the actor **External user** can be found in Table 5.7. According to Figure 3.3 the actor External user has one use case namely "Search for master's thesis" and we have found several hazards connected to this use case.

The first hazard that we identified deals with publishing master theses that are confidential. Neither the actor External user nor anybody except from the Administrator actor should have the possibility to find confidential master's theses. The reason why this may happen can be that the Administrator actor has forgot to mark the checkbox "confidential" when registering the information concerning the respective master's thesis. As a consequence sensitive information may come out. A way to prevent this from happening is first and foremost to give the people responsible for publishing the master's theses properly training in advance. If the damage has happened a message should be sent to the person responsible such that the person can correct the error. In the opposite situation where non-confidential master's

Hazard	Cause	Consequence	Action	H
Confidential master's thesis is published	Missing registration concerning confidentiality	Leaking sensitive information	Message to the person who has registered the master's thesis Training	H29
Non-confidential master's thesis hidden	Wrong registration concerning confidentiality	Information unavailable	Message to the person who has registered the master's thesis Training	
Missing/Wrong information when searching	SQL -> coding error Wrong presentation of data	Cannot find what one is looking for Group info missing	Bug fix [dette var jo en reell feil]	H30
Misleading ranking Missing searchresult	Bad searching algorithm	Risk to not get wanted information Less effective	improve searching algorithm	H31
Illegal access to database by using esc-sequences to do SQL actions	"Hacking"	Could delete or modify data	Strip esc-sequence, "wash" input	H32

Table 5.7: Hazard table for the actor "External user"

theses by mistake are not published, the cause for this is probably the same as for the one in the previous hazard namely an error in connection with checking the wrong alternatives concerning "confidential/no confidential". As with the previous hazard the actions to be done is to provide training programs for the users that deal with this and if the damage has occurred it is necessary to send a message to the person responsible.

When an external actor is searching for master's theses, we identified a hazard related to missing and misleading information concerning the master's theses that are retrieved. If a master's thesis is authored by several people in a cooperation group and the information that the external actors retrieves while searching does not contain this information this may confuse the external actor. The causes for this may be SQL coding errors or wrong presentation of data. Another cause can be that too little information is appended to the master's theses while registering master's theses. The consequences can be that the external actor does not find what he/she is looking for. This hazard was actually a real fault and it was fixed. For the future it might be useful to revise what information should be attached to the master's theses, because the amount of master's theses increase every year and new useful parameters may be introduced for limiting the result set that are retrieved when searching for master's theses.

When searching for a master's thesis it may happen that the ranking of the results is poor and the search results in general is not very good. This may happen because of a poor search algorithm and the actor External user may not find what he/she wants or at least it is taking longer time than expected to find it. This leads to loss of efficiency and a better search and ranking algorithm should be implemented. We have identified poor ranking and missing search results as a hazard.

Hazard	Cause	Consequence	Action	H
Change/Delivery of wrong data	System administrator does more than what he/she is allowed to	Destroy data	Improved log to trace actions	H33

Table 5.8: Hazard table for the actor ”Simulated user”

A hazard related to hacking is based on SQL injections which may be a potential risk in the DAIM system. As said SQL injections appear because of hacking and the consequences may be that data can be deleted or modified. As a preventive action the system should have implemented functions for stripping escape sequences and ”wash” the input.

5.4.8 Actor: Simulated user

The actor **Simulated user** is not shown in the use case diagram in Figure 3.3 because this actor can take every role in the use diagram. The actor **System administrator** can log in as every actor and this is known as the Simulated user. If an actor has some kind of trouble, the System administrator can log in as this actor and he/she will get the same permissions as this actor and could see what the problem is. The hazard that we identified with this actor is the delivery/change of wrong data which may be caused by misuse from the System administrator. This can result in destroyed or manipulated data and the user that was simulated may get all the trouble even though the System administrator was the one doing the wrong actions. For preventing these things to happen there should be implemented a log for tracing actions making it is easier to find the actor responsible.

5.5 Categorization of the Hazards

In our work we were supposed to compare the results from the analysis of the fault reports against the results from the hazard analysis. In accordance to the analysis of the fault reports, we used the classification scheme from Section 4.6.1 and classified each hazard according to these categories. Some of the hazards were assigned several fault types, because they had several causes related to them in addition to the fact that the descriptions of the hazards in general were less detailed than the descriptions of the faults from the fault reports. Some of the hazards in the hazard analysis tables in Section 5.4 are common for several actors, for example System Administrator and Administrator, and if this is the case we have only categorized one of the instances. Table 5.9 shows the results from the process of categorizing the hazards according to fault type.

In Section 4.3 we stated the research question RQ3 which we will make an effort to answer in the following. RQ3 is given as:

Fault type	**Hazards**
Algorithm	H9 H21 H22 H30 H31
Assignment	H20
Checking	H5 H7 H14 H16 H17 H22 H24
Data	H1 H2 H2
Documentation	
Duplicate	
Environment	H11 H16
Function	H3 H5 H7 H8 H10 H13 H14 H18 H19 H22 H28 H29 H32
GUI	H4 H13 H15 H29
Interface	H2
Not fault	H12 H23 H25 H26 H27 H33
Relationship	
Timing/Serialization	H22
Unknown	

Table 5.9: Categorizing of hazards according to fault types.

Fault type	**Number of hazards**	**Number of hazards(%)**
Function	13	30,2%
Checking	7	16,3%
Not fault	6	14,0%
Algorithm	5	11,6%
GUI	4	9,3%
Data	3	7,0%
Environment	2	4,7%
Assignment	1	2,3%
Interface	1	2,3%
Timing/Serialization	1	2,3%
Documentation	0	0,0%
Duplicate	0	0,0%
Relationship	0	0,0%
Unknown	0	0,0%
Total	**43**	**100,0%**

Table 5.10: Number of hazards assigned to the different fault types.

RQ3: What kind of fault types does PHA indicate as possibilities?

From Table 5.10 we see that 13 of the hazards have been assigned to the fault type **Function**, something which is almost twice as much as the second most used fault type used in the categorization process of the hazards. The fault type **Checking** has a number of 7 hazards related to it, while the fault type **Not fault** has 6 hazards connected it. The fault types **Algorithm** and **GUI** then follows with 5 and 4 hazards related. Fault types with no hazards related are **Documentation**, **Duplicate**, **Relationship** and **Unknown**.

The fault type **Function** has 30% of the hazard instances assigned to it and according to Figure 2.2 we can see that faults related to the fault type Function has most occurrences in the design phase compared to the other fault types. This seems to be the case in DAIM as well. If we look

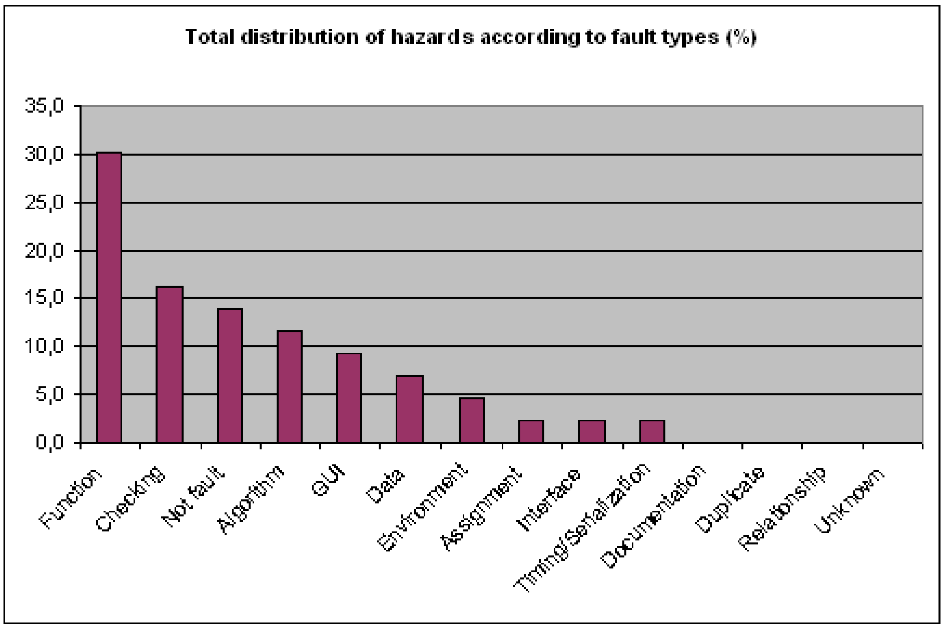


Figure 5.1: Total distribution of hazards with respect to fault types.

at the distribution of the hazards related to the fault types **Checking** and **Assignment**, we see that they make up a total of approximately 20%. From Figure 2.2 we see that the number of instances regarding the fault type Assignment is quite high in the phases Design, Unit Test and Integration and from the same figure we have that Function faults in the design phase are more frequent than Assignment and Checking faults which is also the case in DAIM, see Figure 5.1.

The four fault types **Environment, Interface, Timing/Serialization, Relationship** have together a total of approximately 10% of the instances. This may not be that strange considering that the DAIM system is a quite independent system that does not communicate with many other systems. The only system DAIM depends on is Innsida which provides DAIM with authentication of users.

Concerning the fault types **Unknown** and **Duplicate** they both end up with 0% of the hazards assigned to them. This is not that strange considering that Unknown is used when we do not have enough information for the classification and considering that we had a representant from the DAIM system we had all the information that we needed. With respect to the fault type Duplicate we did not use this at all even though some of the actors from the PHA had the same hazards connected to them; the common hazards were considered as one hazard. This category was mainly introduced with respect to the fault analysis in Chapter 6, because the fault analysis from our previous work [5] had several instances assigned to it.

There have been none hazards related to the fault type **Documentation** and this is reasonable because Documentation fault instances concern faults in later phases; from Section 4.6.1 we have that Documentation faults concern faults where the implementation differs from the specification. Considering that the PHA come from the design phase it is impossible to find those kind of faults here.

Together the fault types **GUI**, **Data** and **Algorithm** represent a total of 30% of the hazard instances. Considering that DAIM is a web-based system it is no surprise that some of the hazards have been assigned to the fault type GUI. The fault types Algorithm and Data will also be present in this kind of system.

The fault type **Not fault** concern instances of hazards that are not actually faults. It can be that users do things what they are not supposed to do or ”human error”.

5.6 Summary

This chapter has consisted of one of our main tasks in this master thesis, namely the PHA of the DAIM system. We have stated our reasons for choosing PHA a long with how the analysis was performed. The results from the PHA was interesting and has been given in several paragraphs.

Chapter 6

Fault Classification and Analysis

In this chapter we will present our results from the classification of the fault reports from DAIM and we will analyze our findings. We will also compare the results obtained from the analysis of DAIM with the results from previous work which also included fault analysis but with different systems in focus.

6.1 Introduction

Section 6.2 presents the results from the categorization process of the fault reports connected to DAIM. In our previous work we did the same kind of analysis as we have done in this master's thesis and therefore we have included our findings from back then in Section 6.3. We then compare the results from the analysis of DAIM and our previous work and this can be found in Section 6.4.

6.2 Results from the Fault Analysis of DAIM

We received two different collections of fault reports from our contact person representing the DAIM system. One of the collections of fault reports, named "Collection 1" from now on, was a simple system log which provided limited information concerning the different faults and the respective descriptions while the other collection, "Collection 2" contained more detailed information regarding each fault. In Section 6.2.1 we will present the results from the analysis of Collection 1 and Section 6.2.2 provides the analysis of Collection 2 while Section 6.2.3 joins the results from the two sections containing the different collections of fault reports just mentioned. In Section 4.3 we stated several research questions and research question RQ1 will be answered here. RQ1 was stated as:

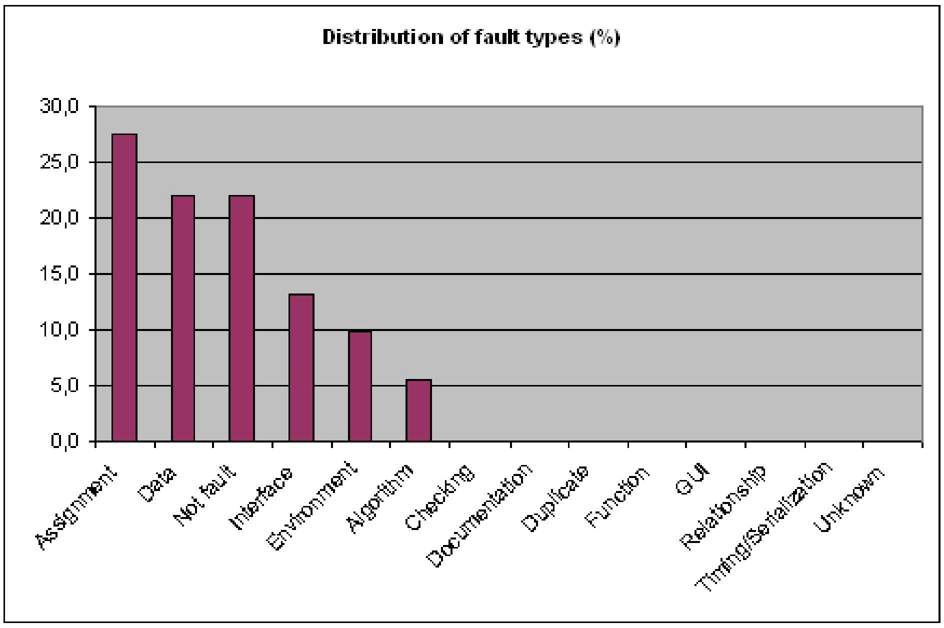


Figure 6.1: Distribution of fault types within Collection 1.

RQ1: How is the distribution of fault types from the fault reports analyzed in DAIM?

6.2.1 Analysis of Fault Reports from Collection 1

In Figure 6.1 and Table 6.1 we present the fault distribution resulting from the fault analysis of Collection 1. This collection of fault reports is based on a system log from the DAIM system and contains system error messages in a short format. During the analysis sessions we received help from one of the representatives of the DAIM system, because some of the error messages were very hard to classify considering that we have not been involved with the development of the system. From the system log we identified a total number of 91 fault instances and from Table 6.1 we can easily see that the fault type **Assignment** is the fault type most often referred to when reporting a fault. 25 of a total of 91 fault instances have been assigned to this fault type and this gives the percentage 27,5%. The fault types **Data** and **Not fault** follow with 20 fault instances each which is approximately 22 % each. **Interface**, **Environment** and **Algorithm** are represented in 13,2%, 9,9% and 5,5% of the occurrences. **Checking**, **Documentation**, **Duplicate**, **Function**, **GUI**, **Relationship**, **Timing/Serialization** and **Unknown** are not used in the classification process at all and have a total of 0% occurrences.

Fault type	**Number of faults**	**Number of faults (%)**
Assignment	25	27,5 %
Data	20	22,0 %
Not fault	20	22,0 %
Interface	12	13,2 %
Environment	9	9,9 %
Algorithm	5	5,5 %
Checking	0	0,0 %
Documentation	0	0,0 %
Duplicate	0	0,0 %
Function	0	0,0 %
GUI	0	0,0 %
Relationship	0	0,0 %
Timing/Serialization	0	0,0 %
Unknown	0	0,0 %
Total	**91**	**100,0%**

Table 6.1: Distribution of fault types within Collection 1.

The most obvious reason why so many fault types have been left out in the classification process is that the fault reports from Collection 1 originate from a system log that is automatically computer generated. Even though the system is developed by people, the system will only do what the developers have programmed the system to do. As time goes by, new errors occur and situations not thought of in advance will show up and most likely the system will not be able to handle these new situations. One of the main differences between a system log and a human generated fault report is that that the system does not have the ability to think for itself and it does exactly what it is told to do while people have the ability to interpret the data in a subjective matter such that more useful and relevant information could be added. In Section 4.6.1 faults referred to as GUI faults are those faults regarding the visual appearance such as the font type or font size is wrong etc. These kinds of faults would hardly ever occur in a system log, because it is a question of appearance from the users point of view and is of a subjective nature. This may be the most obvious reason why GUI related faults are completely left out in the classification process. The same is true for the fault type Documentation which has none fault instances attached it. We find this very reasonable considering that the system log does not represent any assessment of the documentation. Considering that the DAIM system is a quite small system with few developers it might not be that surprising that the fault type Function has no occurrences. From Section 4.6.1 we have that "A Function error is one that effects significant capability, end-user interfaces, product interface, interface with hardware architecture, or global data structure(s) and should require a formal design change". Having in mind that the DAIM system has been developed by very few engineers, it has probably been easier to get the total overview and communication between the participants has been easier. In large system developments projects with many people involved there is a huge need for administration and communication something which takes a lot of time. The fact that the DAIM system is quite a small system with few engineers involved may be one of the reasons why Function faults are not mentioned in the fault reports in Collection 1 considering that Function faults are more severe faults that can be a result of misunderstandings etc. We shall also add the fact that since we have received the fault reports from a system log there are several limitations of what can be registered concerning faults that require for example a formal design change as stated above. The system will not be able to detect this itself, but the engineers will have to detect this when evaluating the system.

It is easier to find the reason why the fault type Relationship has a share of 0% in the analysis. As DAIM only depends on one external system, namely Innsida[1] which is the intranet on NTNU, the probability that faults would occur is less than in systems where many systems depend on each other. Timing/Serialization is related to the fault type Relationship and almost the same answer can be given here as with the fault type Relationship; namely the lack of dependability from a wide range of other external systems.

The fault category Unknown was useful in our previous work [5] when we analyzed other projects, but we did not have the same opportunity to communicate with the developers as we have had in this work. Some of the fault instances from this work was classified as Unknown because the fault reports were poorly described, but now we have received help from one of the representative from the DAIM system such that this fault category has a share of 0%. Faults of the fault type Duplicate also ends up with 0 fault instances which might not be that strange considering that we in our previous work analyzed projects of a bigger size where more people were involved and sometimes the same faults were pointed out by different people, but in the DAIM system which is quite smaller every engineer have had their own area of responsibility.

6.2.2 Analysis of Fault Reports from Collection 2

In Figure 6.2 and Table 6.2 we present the fault distribution resulting from the fault analysis of Collection 2. Collection 2 differs from Collection 1 in the way that the former contains fault reports that are automatically computer generated, namely a system log from the DAIM system, while the latter contains manually created fault reports made by the system developers. The fault reports in Collection 2 have a total number of 26 fault instances. The fault types **Assignment** and **Not fault** are the fault types

[1]https://innsida.ntnu.no 11.05.07 16:57

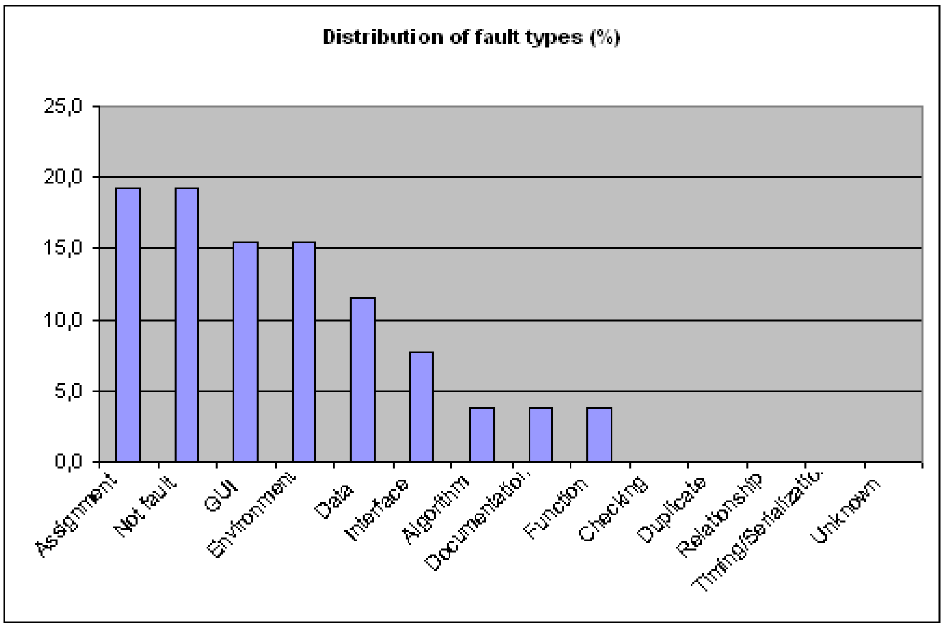


Figure 6.2: Distribution of fault types within Collection 2.

most referenced to with a percentage share of 19,2% each. Further the fault types **GUI** and **Environment** follow with 15,4% each, **Data** has 11,5% and **Interface** has a share of 7,7% while the three fault types **Algorithm**, **Documentation** and **Function** are represented in 3,8% occasions each. The remaining fault types which include **Checking**, **Duplicate**, **Relationship**, **Timing/Serialization** and **Unknown** have no fault instances.

In this collection of fault reports, Collection 2, more of the fault types have been used when classifying the different fault instances compared to the classification process of Collection 1. As stated above, 5 of the fault types have not been referenced to in the classification process while the rest have been referenced to in a lesser or greater degree. This is a little increase from Collection 1, described in Section 6.2.1, where 8 of a total number of 14 fault types were not used at all in the classification process. We have found it reasonable to believe that this may have been a result of more human intervention. As stated previous, the fault reports in Collection 1 are based on a system log while the fault reports in Collection 2 are produced manually by the system developers in the DAIM project. We see from Figure 6.2 that several GUI faults were encountered from the classification process in Collection 2 which was not the case for Collection 1. The fault type Documentation is also represented in Collection 1 with one instance. These two fault types, GUI and Documentation, cannot very easily be revealed from a system log, such as in Collection 1, but in Collection 2 where the

Fault type	**Number of faults**	**Number of faults (%)**
Assignment	5	19,2 %
Not fault	5	19,2 %
GUI	4	15,4 %
Environment	4	15,4 %
Data	3	11,5 %
Interface	2	7,7 %
Algorithm	1	3,8 %
Documentation	1	3,8 %
Function	1	3,8 %
Checking	0	0,0 %
Duplicate	0	0,0 %
Relationship	0	0,0 %
Timing/Serialization	0	0,0 %
Unknown	0	0,0 %
Total	**26**	**100,0%**

Table 6.2: Distribution of fault types within Collection 2.

system developers themselves manually have described the faults that have occurred, it has been easier to point out such faults. This collection of fault reports contains more information concerning each fault compared to Collection 1 which made it easier for us to classify each fault instance into correct fault category. One important aspect of this collection to have in mind is the small number of faults, namely 26 fault instances, which is very little and this makes it more difficult for us to generalize the results. Another weakness with this collection is the short period of time which the fault reports spans over. Collection 2 is based on a two month period while Collection 1 spans over a period of approximately 15 months. Considering the short period of time and the small amount of data makes it harder for us to conclude with anything, but we think that we have enough data to present some trends which can be useful.

6.2.3 Analysis of Collection 1 and Collection 2

In Figure 6.3 and Table 6.3 we present the total fault distribution of Collection 1 and Collection 2. We see that the fault type **Assignment** has the greatest number of occurrences with 25,6%. Fault instances classified as **Not fault** give rise to 21,4% of the fault instances while **Data** faults follow closely with a share of 19,7%. The fault types **Interface** and **Environment** are represented in 12,0% and 11,1% of the fault occurrences. The 5 fault types **Checking**, **Duplicate**, **Relationship**, **Timing/Serialization** and **Unknown** have no occurrences while **Documentation** and **Function** have a share of 0,9% each.

As stated above we observe from Figure 6.3, which contains a graph

Fault type	**Number of faults**	**Number of faults (%)**
Assignment	30	25,6 %
Not fault	25	21,4 %
Data	23	19,7 %
Interface	14	12,0 %
Environment	13	11,1 %
Algorithm	6	5,1 %
GUI	4	3,4 %
Documentation	1	0,9 %
Function	1	0,9 %
Checking	0	0,0 %
Duplicate	0	0,0 %
Relationship	0	0,0 %
Timing/Serialization	0	0,0 %
Unknown	0	0,0 %
Total	**117**	**100,0%**

Table 6.3: Total distribution of fault types after joining Collection 1 and Collection 2.

with the total distribution of faults from Collection 1 and Collection 2, that the fault type Timing/Serialization has no fault instances and we did not find this very surprising. In our previous work [5] when we analyzed more complex systems in the banking and financial sector more interaction between different applications and larger amounts of data were interchanged. Considering that DAIM is not a very complex system, see use case diagram in Figure 3.3, and the degree of interaction with other application is minimal this resulted in null instances of Timing/Serialization faults.

Even though DAIM is a relatively newly developed system, we did not receive as many fault reports as we thought we would in advance. According to our contact person in the DAIM system, the main part of the testing has been performed during development and unit testing has been widely used. Having in mind that there have been quite few persons involved with the development of the system, it has been easier to organize the development and there has been a smaller need for formal communication.

In contrast to our previous work [5] where we classified the different faults also according to severity, the fault reports that we received from DAIM did not contain any information regarding severity. This hindered us from producing statistics concerning the severity.

Faults Not Registered in The Fault Reports

Concerning web browser compatibility DAIM is supposed to work properly with Internet Explorer, Opera and Firefox and the system is continuously tested for those web browsers. As we know that web browsers as well as most technology applications today change from time to time, this results

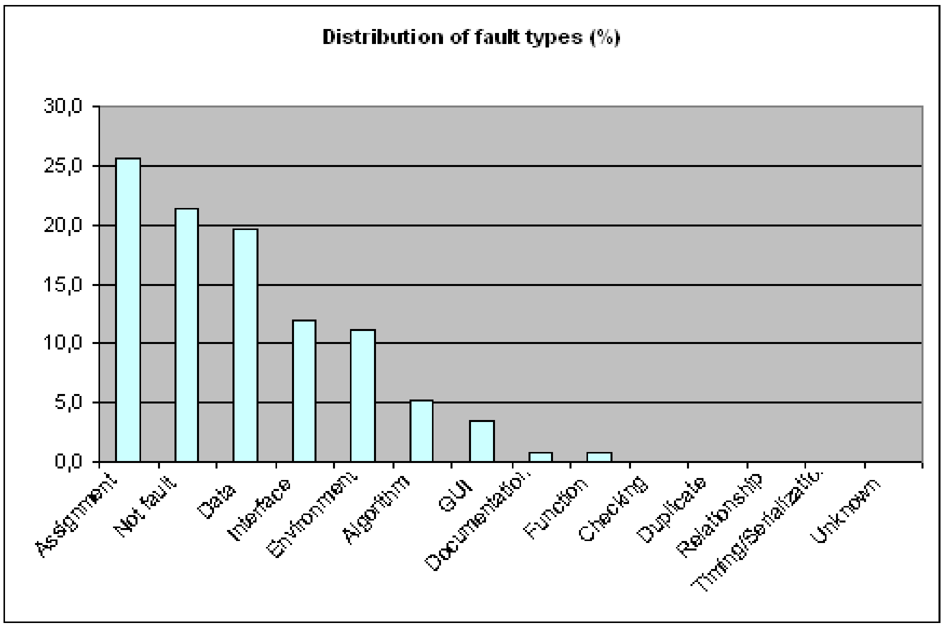


Figure 6.3: Total distribution of fault types after joining Collection 1 and Collection 2.

in challenges for system developers in general and in our case the system developers connected to DAIM. Newer versions of web browsers show up once in a while and they are updated enabling new functionality and use of new standards while some functionality are disabled and removed. The web browsers do also give the users choices concerning what functionality to be enabled and disabled such that the user environment may differ a lot from user to user which makes it more difficult to make applications that work no matter how the user environment is. Examples from DAIM related to this topic is the use of Javascript which is required for the DAIM web page to work properly and according to one of the developers in the DAIM project there was also a problem concerning activation and deactivation of pop-up windows in different web browsers resulting in that the use pop-up windows was removed from the web page. Even though DAIM is tested with the three browsers mentioned above, there is a chance that some users of the system may have extraordinary settings enabled in their respective user environment causing the DAIM web page not to function properly. As time goes by some people update their web browsers to newer versions while some stick to their old versions and this may result in problems in the long run. The problems that we have discussed in this paragraph are some of the problems that are not registered in the fault reports received from the DAIM system.

Fault type	**Number of faults**	**Number of faults (%)**
Function	191	21,2 %
Not fault	158	17,5 %
GUI	138	15,3 %
Unknown	87	9,7 %
Assignment	75	8,3 %
Checking	58	6,4 %
Data	46	5,1 %
Algorithm	37	4,1 %
Duplicate	36	4,0 %
Environment	36	4,0 %
Interface	11	1,2 %
Timing/Serialization	11	1,2 %
Relationship	9	1,0 %
Documentation	8	0,9 %
Total	**901**	**100,00%**

Table 6.4: Total distribution of fault types from Collection O.

6.3 Results from the Fault Analysis from our Previous Work

In our previous work [5] we did the same kind of analysis as we have done in this master's thesis concerning the fault reports. We received 901 fault reports from 5 different projects from one company, from now on we name it Company O, that were developing software for the banking and financial sector and we classified the fault instances according to the same classification scheme as we have used in this work namely the ODC classification scheme, described in Section 4.6.1. The projects that we received were of quite similar nature and can be seen as more or less homogenous projects. When we will compare the results from these projects with the DAIM system, see Section 6.4, we will use the term Collection O which includes all the fault reports from the 5 projects as a whole.

In Figure 6.4 the distribution of fault types from Collection O is shown graphically while Table 6.4 describes the distribution more exactly. It can easily be seen that the fault type **Function** is the fault type with most fault instances assigned to it and it occurs in 21,2% of all fault occurrences. The fault type **Not fault** follows with 17,5% and is closely followed by the fault type **GUI** which has a share of 15,3% of all fault occurrences. The fault types **Interface**, **Timing/Serialization**, **Relationship** and **Documentation** were the fault types less used in the categorization process and each category was assigned approximately 1,0% of the fault instances each.

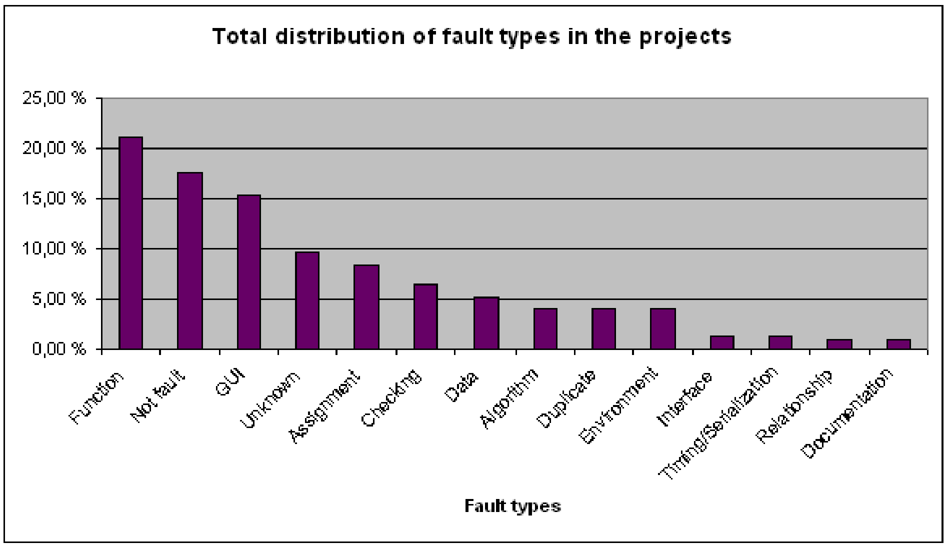


Figure 6.4: Total distribution of fault types from Collection O.

6.4 Comparison of Results from the Analysis of DAIM and Previous Work

In Section 4.3 we stated research question RQ2 and this question will be answered in this section. Research Question RQ2 was:

RQ2: How does the distribution of fault types from the fault reports analyzed in DAIM differ from the fault reports in our previous work?

The comparison of the fault distribution from the fault reports regarding DAIM and Collection O can be seen in Figure 6.5 and we see that the fault distributions differ in many ways, but having in mind that the fault reports from Collection O comes from 5 different but quite homogenous projects we could not expect to obtain that similar results. It is important to have in mind that the DAIM system differs from the projects that we analyzed from company O in several ways. First of all the fault reports from Collection O resides from a company in the banking and financial sector something which implies that different kind of tasks are performed by the system in comparison to the DAIM system which is a web-based system mainly built around a database. Another issue to have in mind is that we had several more fault reports to classify and analyze from company O; exactly 901 fault reports in contrast to 117 fault report from DAIM. The complexity of the DAIM project is not that high as most of the projects analyzed from company O and this may be a reason why the results from the comparison

Fault Category	**DAIM (%)**	**Collection O (%)**	**Absolute difference (%)**
Documentation	0,9	0,9	0,0
Relationship	0,0	1,0	1,0
Algorithm	5,1	4,1	1,0
Timing/Serialization	0,0	1,2	1,2
Not fault	21,4	17,5	3,9
Duplicate	0,0	4,0	4,0
Checking	0,0	6,4	6,4
Environment	11,1	4,0	7,1
Unknown	0,0	9,7	9,7
Interface	12,0	1,2	10,8
GUI	3,4	15,3	11,9
Data	19,7	5,1	14,6
Assignment	25,6	8,3	17,3
Function	0,9	21,2	20,3

Table 6.5: Comparison of the fault distribution from DAIM and Collection O.

of the fault reports between the two collections may vary a lot.

From Figure 6.5 we see that the distribution of fault types varies a lot between DAIM and Collection O and in Table 6.5 the exact values are presented numerically. The most striking difference seems to be related to the fault type **Function** where DAIM has a percentage share of 0,9% in contrast to 21,2% from the fault reports from Collection O and it is a difference of 20,3%. We think that some of the reasons for this may be related to the fact that the DAIM project is not that complex as the projects in Collection O and there are less people involved with the DAIM system. Fault instances of the fault type Function tend to be more serious faults requiring a formal design change and when more people are involved in the development process misunderstandings would normally occur more often. Some of the projects from Company O were quite complex with several modules working together and this may have resulted in higher density of Function faults. The fault type **Assignment** has a difference of 17,3% between DAIM and Collection O where DAIM has a share of 25,6% and Collection O a share of 8,3%. We suspect that this difference may be related to some of the same reasons mentioned with Function faults. DAIM is a less complex system with a few numbers of modules and the probability that some big changes are needed is smaller than in more complex and big systems. Assignment faults are often faults considered as less severe compared to Function faults and considering that the DAIM system is not very complex in addition to the fact the there are few people involved in the development, it is easier to obtain the overall overview with respect to the implementation of the system and delegation of tasks and there is a minor need for formal communication. We think that all the reasons that we have mentioned may explain some of the big difference

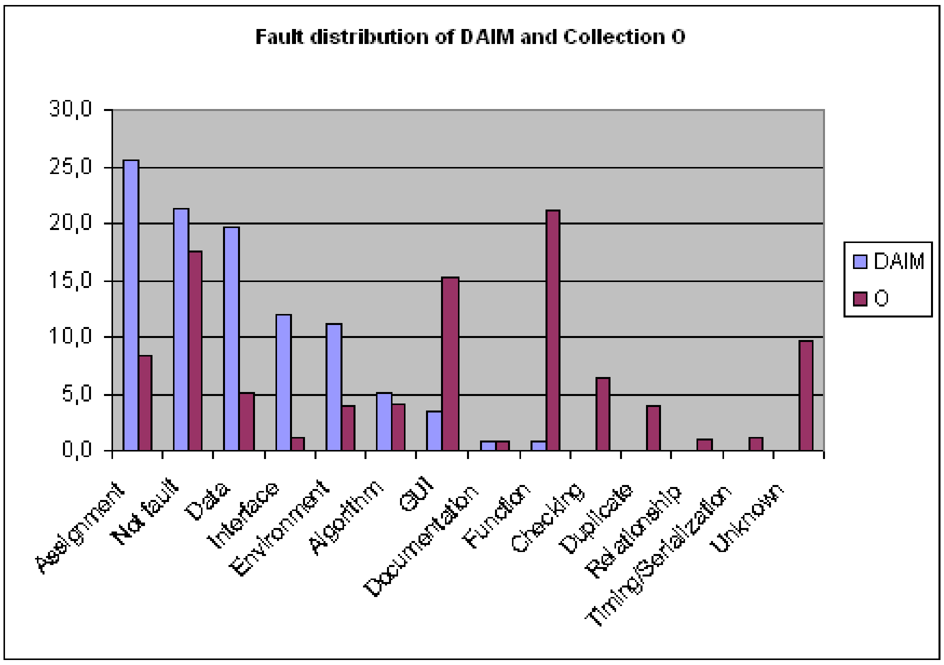


Figure 6.5: Comparison of the total distribution of fault types from DAIM and Collection O.

between the fault reports from DAIM and Collection O concerning the fault type Assignment.

With respect to the fault type **Data** there is a difference of 14,5% between the collections of fault reports from DAIM and Company O. According to our contact person in DAIM some of the fault instances referring to Data faults were probably related to some ”extra” testing done in connection with a master’s thesis from last year [11] concerning FMEA analysis. Considering that the number of fault reports received from DAIM was quite few, this may have affected the results some. In the fault reports from DAIM the fault type **GUI** occurs in 3,4% occasions while the fault reports from Company O has a share of 15,3% which makes a difference of 11,9%. As stated in Section 6.2.1 many of the fault reports from DAIM come from a system log and the system log will hardly be able to discover GUI faults. The fault instances categorized as **Unknown** is 0% with respect to the DAIM system while from Collection O we have that 9.7% of the fault instances have been assigned to this fault type. During the categorization process of DAIM we have had more contact with our contact person compared to when we analyzed the fault reports from Collection O. We did not have this possibility with Collection O such that sometimes we had to categorize the faults as Unknown because the fault reports were not very well described.

6.5 Summary

In this chapter we have performed the analysis of fault reports from DAIM and interesting findings were discovered. In addition to the analysis of DAIM, we also included the results from our previous work which also included analysis of fault reports and we compared these results with the results that we obtained from the analysis of DAIM.

Chapter 7

Results

In this chapter we will present the results from our work and we will point out the most important findings that we have made.

7.1 Introduction

Section 7.2 gives some comments concerning how the process of comparing the hazards and faults has been. In Section 7.3 we give the results from the comparison of the hazards and faults while Section 7.4 contains a brief summary of the results. In Section 7.5 we present some trends that were discovered with respect to faults and hazards in the different phases of the development.

7.2 The Process of Comparing Hazards and Faults

In the process of comparing the hazards with the faults from the fault reports we met some challenges. Having in mind that the hazards are of a quite general nature and do not contain details in the same degree as the fault reports this had to be taken into account when comparing. It was difficult to find exact matches between the hazards and the fault reports. The descriptions of the hazards are of a more oral nature compared to the fault reports which are more specific in the way they are written. We tried to do the process of comparing the information in the best possible way and we started out by going through one and one hazard at the time. In Chapter 5 where the hazard analysis tables are given, the rightmost column contains an identifier of the form ”H??” and we started with the hazard connected to H1. By looking at Table 5.5 it can easily be determined what fault category this hazard has been assigned to and based on this we have gone through the fault instances from the fault reports focusing on fault instances related to this fault category to see if the hazard have similarities with the different fault instances. As H1 is categorized as a ”Data” fault, we went through all the fault reports categorized as ”Data” faults and tried to find if there were some connections between the hazard and the fault instances.

7.3 The Results from the Comparison

We came up with a total number of 33 hazards where 7 of these were assigned ”Not Fault”, something which imply that these hazards were not related to the system but involved user faults. In the following subsections, Section 7.3.1 to Section 7.3.9, we will present the results from the comparison according to fault category. Some of the information given in the following subsections are based on Table 6.3, Table 5.10 and Table 5.9. In Section 4.3 we stated a research question RQ4 that involves finding if there is a connection between hazards from the PHA process and the fault reports and this research question will be answered in the following subsections.

RQ4: Does the hazard analysis technique PHA applied to the DAIM specification documents reveal faults that have actually appeared in the system?

7.3.1 Algorithm Faults

According to Table 5.9 we classified 5 of the hazards to the fault category ”Algorithm”. From the fault reports, see Table 6.3, we have that 6 of the fault instances was classified as ”Algorithm” faults. We did not find any connections between the fault instances from the fault reports categorized as ”Algorithm” faults and the hazards categorized as ”Algorithm” faults.

7.3.2 Assignment Faults

One of the hazards was classified as a ”Assignment” fault while in the fault reports we classified 30 fault instances to the category ”Assignment” faults. We did not find any relationship between the hazards and the fault reports. Having in mind that ”Assignment” faults are normally more detailed and often closely connected to coding errors, this may be one reason why the number of ”Assignment” faults is that high for the fault reports, but low for the hazards. Hazards are of a more general nature and do not include such level of details.

7.3.3 Checking Faults

The number of hazards categorized as ”Checking” faults was 7 and the number of fault instances categorized as ”Checking” faults was 0. This means that there was nothing to compare such that no relation between hazards categorized as ”Checking” faults and fault reports categorized as ”Checking” could be proved.

7.3.4 Data Faults

From the fault reports we categorized 23 of the fault instances as "Data" faults while from the hazard analysis we categorized 3 of the hazards as belonging to "Data" faults. The results from the comparison of the fault reports and hazards categorized as "Data" faults gave us some interesting findings. One of the fault reports (F1) has the description *Missing username* and in the column improvement it is stated that *Need to improve the import from FS.* From the hazard analysis we can see that hazard H2 from Table 5.1 concerns the hazard *Unauthorized access* and its cause is said to be *User is missing because of gap in export from FS.* From these descriptions we have reason to believe that there is connection between the hazard and the fault report. We also found that hazard H11 was related to the fault instance described above (F1). From Table 5.3 we see that H11 is described with the hazards *"Wrong" data from FS* and *Inconsequent data.* It is stated that the causes are *Inconsequent data input*, *Fault registered* and *Wrong format.* Considering that the fault description from the fault report is *Missing username* it can be the case that the username is wrongly registered in FS meaning that a character can be misplaced by another etc.

7.3.5 Environment Faults

The number of fault instances from the fault reports categorized as "Environment" faults was 13 while from the hazard analysis we identified 2 hazards categorized as "Environment" faults. From Table 5.6 we see that the hazard H16 is described as *Wrong value in contract - datafault/typefault* and the causes are *Validation fault* and *Characterset fault.* From the fault reports we have one fault report (F2) that is described as *ÆØÅ in email* and in the improvement column it is stated that *Changing of the character set regarding MIME.* We found this partly relevant to the hazard just mentioned, because it deals with character sets and in a system where one uses two languages with different character sets this may cause trouble for the system.

7.3.6 Function Faults

We categorized 13 of the hazards to the category "Function" faults and from the fault reports we had that one of the fault reports was assigned to the category "Function" faults. This fault report instance (F3) has been related to two of the hazards, namely H22 and H28. The fault description of F3 is *Email was not sent to other members of the group when the cooperation group was established.* Hazard H22, which states *Mess in the group composition* and the cause for this *People have difficulties when deciding*, can be related to the fault instance F3 which deals with problems regarding cooperation groups. Hazard H28 can also be related to F3, because it deals with problems regarding cooperation groups. H28 is described as *One person in the group* *delivers master's thesis without consent* and the cause is given as *Different view in the group.* This hazard may not be directly linked to F3, but we wanted to add this because it deals with issues concerning cooperation in groups.

7.3.7 GUI Faults

We registered a total of 4 "GUI" faults related to the hazards and from the fault reports we had a total of 4 as well. We came up with one weak connection between the hazard H15 and the fault report F4. H15 is described as *Wrong value in contract -datafault/typefault* and the cause is stated *Sloppiness MMI Touch wrong key on keyboard.* The fault instance F4 is given as *The student did not find the textfield where he/she was supposed to fill in delivery date* and this resulted in wrong values in the contract as stated in the description of hazard H15.

7.3.8 Interface Faults

We categorized one of the hazards as the fault type "Interface". Among the fault reports there have been identified 14 fault instances related to the fault type "Interface". We did not succeed in finding any connections between the hazards and the fault instances from the fault reports.

7.3.9 Timing/Serialization Faults

The number of "Timing/Serialization" faults from the fault reports was 0 and from the hazard analysis we had a number of 1 hazard assigned to the category "Timing/Serialization". Even though we had no instances from the fault reports categorized as "Timing/Seralization" we succeeded in finding another fault fault instance, F16 which is categorized as an "Assignment" fault, that can be related to the hazard H22 which is categorized in the group "Timing/Serialization" faults. H22 is described as *Mess in the group composition* with the cause *People have difficulties when deciding* while F16 is described as *No group members assigned to a master's thesis - Master's thesis 1212 has no students registered.*

7.4 A Brief Summary of the Results from the Comparison

From the preceding sections, Section 7.3.1 to Section 7.3.9, we did some findings which we will briefly summarize here. In general we did not find as many connections as we were hoping for before we started our work. Some of the reasons may be that DAIM is a newly developed and small system and we did not have that much material to work on, the numbers of fault reports were few etc.

From the comparison of fault reports and hazard instances we did find some connections or tendencies and they were related to the fault types **Data**, **Environment**, **Function**, **GUI** and **Timing/Serialization**. None of the connections we found were very strong, but as we use the term connections when referring to the findings there were something that related some of the hazard instances to the fault instances. There were some indications saying that some of the faults could have been avoided based on the results from the comparison that we have just done.

7.5 Trends

In this Section we will look at some trends that we have found interesting concerning faults and hazards identified in different phases.

According to Figure 2.2 and Figure 2.3 we have that fault instances related to the fault type Function faults occur more often in earlier phases of the development rather than in later phases; Function faults have most occurrences in the design phase while the occurrences decrease as the project evolves and enters later phases. Considering that the PHA analysis was performed based on documentation from the design phase of the development of DAIM, wee see from Table 5.10 that hazards assigned to the fault type Function actually have the highest frequency compared to the rest of the fault types in DAIM and this shows that the trend from Figure 2.2 actually coincides with the results from the PHA concerning the fault type Function. By looking at Table 6.3, which shows the distribution of fault types from the fault analysis of DAIM, we see that the fault type Function has a share of approximately 1% of the fault instances and this indicates that the number of fault instances related to Function faults have decreased as the development of the DAIM system has evolved. The fault reports from DAIM was received after the system has been in operation for a while.

Figure 2.3 states that fault instances of the fault type Assignment belong to the process of coding. By looking at Figure 6.3 we see that fault instances classified as Assignment faults have the highest number of occurrences and having in mind that DAIM has been in service for some time now and that the system is under continually development, this seems very reasonable. From Figure 5.1, which originates from the hazard analysis of DAIM, we can easily see that the number of hazards categorized as Assignment faults is quite low and considering that Assignment faults usually are more detailed (and less severe) than Function faults, it seems reasonable that Assignment faults have that little share of faults in early phases of the development.

Another trend that we discovered concerns the fault type **Interface**. From Figure 2.2 we see that the fault type Interface seems to increase in number of occurrences as the project evolves. This tendency can also be spotted from our different analyzes concerning DAIM; Table 5.10, which resides from the PHA, shows that hazard instances identified as Interface faults have a share of 2,3%, while Table 6.3 which comes from the analysis of fault reports, shows that fault instances referred to as Interface faults have the share 12,0%. This is an increase of approximately 10%.

Hazards classified as **Algorithm** faults have been reported in 11,6% of the occurrences of the different hazards, see Table 5.10, while from Table 6.3 we have that fault instances from the fault reports identified as Algorithm faults have a share of 5,1%. According to Figure 2.3 we see that Algorithm faults are related to the process stage ”Low level design” and this seems to be the case for the DAIM because of the decrease of occurrences as the project has evolved.

7.6 Summary

In this chapter we have performed the comparison of the results from the PHA analysis and fault analysis with respect to the DAIM system. We found some interesting results from the comparison, but not as many as we had expected. In addition to the comparison we also included a section about the trends that we found concerning faults and hazards with respect to different development phases.

Chapter 8

Discussion of Validity and Own Contribution

This chapter contains the discussion part of the master's thesis along with own contributions. The discussion part is closely related to the threats to the validity of our work.

8.1 Discussion of Validity

The question of how valid the results are from scientific work, is a very important and fundamental issue and needs attention. In [16] four different kinds of validity types are described and in the following subsections, Section 8.1.1 to Section 8.1.4, we will look into the different threats to validity that the results from this master's thesis may be suffering from. The four different validity types are:

- **Conclusion Validity**
- **Internal Validity**
- **Construct Validity**
- **External Validity**

8.1.1 Conclusion Validity

Concerning conclusion validity it is stated in [16] that: "The basic principle is that when you measure a phenomenon twice, the outcome shall be the same". Considering that this master's thesis has involved several tasks we will briefly look at each task with respect to the conclusion validity. Concerning the hazard analysis PHA we know that this kind of analysis includes some elements of creativity and experience. Regarding the creativity this is very person-dependant, but as there were several people with experience from this kind of analysis present during the different sessions, we claim to have reduced the treat of conclusion validity some, having in mind that the experienced people may have stimulated the other participants to think more creatively. The task related to classification and analysis of fault reports does also need some attention, because fault analysis also include some elements of subjectivity. We felt that we reduced this treat some by having defined the different fault categories properly in addition to state good examples of the different categories. In our previous work [5] we experienced that some of the fault instances were hard to classify because of insufficient descriptions, but in this work we were lucky to have more contact with the people involved with DAIM such that the number of fault instances characterized as "Unknown" became 0%. In addition to this we did some of the parts in the process of categorization together with our contact person from DAIM such that we had to agree upon the different fault instances. With respect to the third part of our task concerning the comparison of the results from the fault analysis and hazard analysis, this part will also have to be considered as partly subjective, but as we did some of the comparison together with our supervisor this led to a common understanding and this have probably reduced the treat of conclusion validity.

Another treat we have identified regards the issue of "fishing" which deals with the situation where one is "fishing" for a specific result. As we were to compare the results from the PHA and the fault analysis, we were hoping to find some connections between the results and this may have lead us to find "weak" connections which others would not have found. We think we reduced this treat some by doing some of the comparison together with or supervisor who has experience with this kind of work.

8.1.2 Internal Validity

Concerning the internal validity we have found one treat that is partly related to one of the treats identified in Section 8.1.3 concerning construct validity and it is named "maturation". This treat concerns the effect that the subjects react differently as time goes by. In our previous work [5] we did the same kind of fault analysis including classification of faults as we have done in this work and we might have seen things differently in this work compared to our previous work concerning the process of classification of the fault instances. We claim to have reduced some of this treat by having the same material present as we had back then; the material concerning the fault categories consisted of good descriptions of the fault categories in addition to good examples for illustrating what kind of faults this category should contain.

8.1.3 Construct Validity

From [16] we have a treat described as "Inadequate preoperational explication of constructs" which makes a point of the importance of having defined the correct goals in advance. There is given an example where one is going to compare two different inspection methods and it is important to have a clear idea of how to measure this; "What method is best? The method with the fewest faults?" etc. In this master's thesis this could have been a treat for us as well, but since we actually compare the results obtained by using two different inspection techniques, it cannot be considered as a treat of this kind. PHA is a method used earlier in the process of system development, such as in the design phase, while fault analysis normally is performed at later stages, after the system has been operable for a while. With this master's thesis we wanted to see if we could find connections between the results from the inspection techniques, but we did not actually know what we would find or if we would find anything at all. Our point is that we did not know exactly what we were looking for. The results that we have come up with has to be seen in a subjective matter, because other people might have found other connections than we did. For reducing subjective opinions we did some of the work concerning the comparison of the results from the inspection techniques together with our supervisor.

8.1.4 External Validity

External validity concerns the possibility of generalizing the results obtained from a project outside the project setting. In our work we have analyzed data from one project and this may conflict some with the ability to generalize the results. Considering that various companies have their one way of doing things and that projects differ from project to project, it can be that the results would have been different if we worked with another project. To our advantage it shall be stated that the system that we worked with was of a quite ordinary nature, namely a typical Web-based IT-system. Having this in mind we think that the results would not necessary vary that much if another system had been analyzed; at least we think we have reduced the treat some because of this fact.

Another treat related to external validity and difficulty of generalizing results that we have found concern the size of the system that we have worked with. DAIM is a small system with respect to complexity and number of developers, and we received a few number of fault reports. This may have affected the distribution of faults, such as less fault instances of the fault type Function than normal etc. Considering that the system is quite young it might have been a suggestion to do this kind of analysis later on. PHA is supposed to be performed during the early stages of a project, but if the fault analysis had been performed later on it would hopefully have existed more fault reports, but it shall be stated that this would not necessary have been the case. Performing the fault analysis later on would not necessarily guarantee more fault reports or big changes in the distribution of fault types. Luckily we had a very good collaboration with our contact person from DAIM and the fault reports that we received were of a good qualitative nature. As stated in Section 8.1.1 the number of fault instances classified as the fault type **Unknown** was 0% and this shows that all of the fault instances were classified even though the number of fault instances was small. We believe that this may have reduced the treat some.

8.2 Own Contribution

During our work with the DAIM system, we have found some suggestions for improving the system. One of the suggestions consists of making some kind of a feedback functionality or a forum on the DAIM web page. If the users of the system have questions or want to report errors on the web page, a way to report this should be supported on the web page. Today the web page displays an email address where the users can give the developers feedback, but for example in the situation where many users are wondering about the same questions it might have been a good idea to have some kind of a forum. Then the developers would avoid receiving many emails concerning the same issues and they could spend their time on other tasks. In a forum other users might also comment on other user's questions such that the developers do not need to be involved in every occasion, but they could have an administrator role and correct the forum posts in the case where a user has given wrong information etc. There exist a web page for frequently asked questions (FAQ), but as time goes by new issues do occur. For the moment DAIM only includes master students from the IME faculty, but according to one of the developers in the DAIM system, it is possible that DAIM may be rolled out for all master students at NTNU. If this happens, the number of users will increase greatly and there might be changes in the system because the different faculties might have their own wishes regarding new functionality etc.

On the FAQ web page all supported web browser for the DAIM web page are listed, but the web browser version is not mentioned. As we know that newer versions of the web browsers occur once in a while and new functionality is added and some functionality is removed the FAQ web page should include this kind of information.

Chapter 9

Conclusion and Further Work

This chapter will present the conclusions that we have made in addition to some comments about what could be interesting to study in further work.

9.1 Conclusion

In this master's thesis we have done several analyzes with respect to the IT system called DAIM. We performed a hazard analysis technique called PHA on DAIM and we categorized the hazards according to the classification scheme named ODC. This classification was performed in order to compare the results from the PHA with the results from our fault analysis of the DAIM system. The results from the fault analysis of DAIM were also compared with results from our previous work [5], which also concerned fault analysis, but these fault reports came from other systems.

We decided on four research questions in our work and in the following paragraphs we will give brief answers to those. Concerning RQ1 we were supposed to look at the distribution of fault types in DAIM and we found that fault instances related to the fault type **Assignment** had most occurrences (25,6%) closely followed by fault instances assigned to the fault types **Not fault** (21,4%) and **Data** (19,7%). The three fault types just mentioned had a total of approximately 65% of all the fault instances. The five fault types **Checking**, **Duplicate**, **Relationship**, **Timing/Serialization** and **Unknown** had 0% fault instances assigned to them.

In RQ2 we compared the distribution of fault reports assigned to different fault types from DAIM with our previous work and we found that were several differences. We have from Table 6.5 that the fault type **Function** is the fault type with the greatest difference (20,3%) concerning fault occurrences between DAIM and our previous work ; the fault types **Assignment** and **Data** follow with a difference of 17,3% and 14,6% between DAIM and our previous work. On the other side of the scale regarding the fault types with the least difference among the systems we have the fault types **Timing/Serialization** (1,2%), **Algorithm**(1,0%), **Relationship** (1,0%) and **Documentation** (0,0%).

RQ3 concerns the question of what fault types PHA indicates as possibilities and we found some clear tendencies. Hazard instances categorized as **Function** faults occurs in 30,2% of the occasions while hazard instances categorized as **Checking**, ranked as number two, has a share of 16,3% of all occurrences. Next we find hazard instances categorized as **Not fault** and **Algorithm** with a percentage share of 14,0% and 11,6%. The fault types **Documentation**, **Duplicate**, **Relationship** and **Unknown** have no hazard instances connected to them.

With respect to RQ4 where we were wondering if PHA actually reveal faults that appear, we did not find any definitive yes or no's. From Section 7.3 we have that some of the fault instances from the fault reports might be considered as related to some of the hazard instances. We found some connections/tendencies between the hazard and fault instances which might be interpreted in a way saying that some of the faults could have been avoided if one had focused more on the results from the hazard analysis. But as we know, the hazard analysis has been performed after the system has been in service for a while.

We will conclude with that PHA is a good technique for identifying treats and if we had used this technique on another system with different material to analyze, the results might have been different. DAIM is a small and simple system and faults from earlier phases have probably not been reported as faults at all, because the developers have handled these faults as they have appeared.

As we see it doing such an analysis before starting the implementation is by no means meaningless, because it may stimulate the developers to be more aware of what kind of faults that can occur and by having this in mind this itself might be of help reducing some of the faults/treats identified before the process of implementing the system.

9.2 Further Work

Considering that we have found some connections in our work regarding faults and hazards, it would have been very interesting to perform a PHA on another system than DAIM; it might have been a good idea to analyze a more complex system that contained different data material to use in the process. Since we have performed PHA in this master's thesis, it might have been interesting to perform another hazard analysis technique such as HazOP to see if this technique might be better suited.

Bibliography

[1] United Kingdom Software Metrics Association. Quality standards defect measurement manual. 2000.

[2] J. A. Børretzen, T. Stålhane, T. Lauritsen, and P. T. Myhrer. Safety activities during early software project phases. *Proceedings, Norwegian Informatics Conference*, 2004.

[3] Chillarege. What is odc?
`http://www.chillarege.com/odc/odcbackground.html [17.06.07 15:00]`.

[4] R. Chillarege, I. S. Bhandardi, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M. Wong. Orthogonal defect classification - a concept for in-process measurements. *IEEE Transactions on Software Engineering.* Volume 18, Issue 11, Nov. 1992.

[5] J. Dyre-Hansen. Analysis of fault reports from online-systems. Department of Computer and Information Science, NTNU, 2006.

[6] IEEE R. Sandhu E. Bertino, Fellow. Database security - concepts, approaches and challenges. 2(1), january-march 2005.

[7] K. El Emam and I. Wieczorek. The repeatability of code defect classifications. *Proceedings, The Ninth International Symposium on Software Reliability Engineering*, 1998.

[8] IBM. Details of odc v 5.11.
`http://www.research.ibm.com/softeng/ODC/DETODC.HTM [14.06.07 15:00]`.

[9] J-C. Laprie. Dependable computing and fault tolerance: Concepts and terminology. *15th Int. Symp. on Fault-Tolerant Computing*, 1985.

[10] N. G. Leveson. *Safeware: System Safety and Computers*, chapter 12, pages 253–254. Addison-Wesley, 1st edition edition, 1995.

[11] T. H. T. Pham. Websys- robustness assessment and testing. Master's thesis, Department of Computer and Information Science, NTNU, 2006.

[12] M. Rausand. *Risiko analyse.* Tapir Forlag, 1991.

[13] N. Storey. *Safety-Critical Computer Systems*, chapter 1. Addison-Wesley, 1st edition edition, 1996.

[14] T. Stålhane and T. Lauritsen. Safety methods in software process improvement. 2005.

[15] weibull.com. Relationship between availability and reliability. `http://www.weibull.com/hotwire/issue26/relbasics26.htm [13.03.07]`.

[16] C. Wohlin, M. Höst P. Runeson, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation In Software Engineering - An Introduction.* Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[17] D. Zage and W. Zage. An analysis of the fault correction process in a large-scale sdl production model. *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, 2003.