NORGES TEKNISK-NATURVITENSKAPELIGE UNIVERSITET

FAKULTET FOR INFORMASJONSTEKNOLOGI, MATEMATIKK OG ELEKTROTEKNIKK

HOVEDOPPGAVE

Kandidatens navn: Andreas Knudsen, Kristian Marheim Abrahamsen

Fag: Datateknikk

Oppgavens tittel (engelsk): Reuse Of Experience In HazOp

Oppgavens tekst:

Some systems can have serious impact on their environment if they fail. These systems can be a threat to both the people and nature surrounding it. HazOp is a methodology for finding hazards in such systems. It is a knowledge intensive technique in which a group works through the system design representation to find all possible hazards. Previous studies have indicated that experienced HazOp personnel is crucial to the success of the method.

The aim of this thesis is to create a software tool to facilitate reuse of knowledge in connection to this method. The work will include an experiment to test the value of knowledge reuse in HazOp.

Oppgaven gitt:	19. januar 2004
Besvarelsen leveres innen:	14. juni 2004
Besvarelsen levert:	14. juni 2004
Utført ved:	Institutt for datateknikk og informasjonsvitenskap
Veileder:	Tor Stålhane

Trondheim, 14. Juni 2004

Tor Stålhane
Faglærer

Abstract

This report presents a study of the effect of reusing experience in the Hazards and Operability Analysis method (HazOp method) with regards to how the effectiveness of the method is affected. The study was conducted by first creating a software tool for experience reuse in HazOp, then testing that tool in a student experiment in which the participants used the tool when conducting a HazOp.

During the experiment it was found that students using the tool found 21% more hazards in the system under study than their counterparts. After conducting the experiment it was found that there was a 94% certainty that this improvement was not due to random effects.

Contents

List of Tables

List of Figures

Preface

This report is the result of work performed in a project at the Department of Computer and Information Science, Norwegian University of Science and Technology, during the spring of 2004. This project is the Masters Thesis of Andreas Knudsen and Kristian Marheim Abrahamsen, for the degree of Master of Technology: Computer Science

We would like to extend our gratitude to Professor Tor Stålhane at the Department of Computer and Information Science, NTNU, who has been our supervisor during the project and has provided us with invaluable guidance and feedback.

We would also like to thank Knut Carlsen at NTNU Drift for his help in securing technical resources for our experiment.

Andreas Knudsen Kristian Marheim Abrahamsen

1 Introduction

Some systems can have serious impact on their environment if they fail. If a control system for a nuclear power plant fails, the consequences for people and environment can be disastrous. According to B. Carter , *"Major disasters, particularly in the oil, nuclear and chemical spheres, have stimulated attention on risk reduction to protect the environment and human safety[4]."* These systems often contain software components. It is necessary to know how and why a system might fail to reduce the risk it poses to the environment. In this context we focus on system hazards. Hazards, accidents, and risks are subjects that are tightly coupled. We will use the C.Knutsons definition of hazard in this project:

"A hazard is a set of conditions, or a state, that could lead to an accident, given the right environmental trigger or set of events. An accident is the realization of the negative potential inherent in a hazard." [8]

Being aware of what hazards a system might present, and dealing with these before they turn into accidents is crucial. In order to become aware of hazards, one must analyse the systems in question.

Avoiding risks when designing systems is an important part of systems engineering. As software becomes an ever larger part of the systems that surround us, avoiding risks also increasingly falls under the domain of software engineering.

"(...) and so we learn to live with the inherent risks that surround us, because the cost of avoidance just seems simply too high. However, as technology becomes more and more ubiquitous, with more of that technology controlled by software, a greater portion of the risk we face is ultimately in the hands of software engineers." [8]

In this paper we create a software tool that provides access to previous experiences made doing such analyses with the HazOp methodology and run an experiment to test whether using such a tool can be helpful when doing HazOp analyses.

1.1 Hazards Identification and Methodology

There are several methodologies and techniques for identifying risks and hazards in systems. Some of the most common are What If?, Interaction Analysis, Zonal Analysis, Checklists, Fault Mode and Effect Analysis (FMEA) and HazOp. Which is the most appropriate technique will depend on the project at hand. A combination of them can give the best result [6].

1.2 HazOp

This paper will focus on the HazOp methodology. *"Hazard and operability study (HazOp) is perhaps the most powerful technique for the identification and analysis of hazards[6]."* HazOp was originally developed for the chemical industry but has been successfully employed in other industries [6]. HazOp is a group study where the concept is to review a system in a series of meetings, during which a multidisciplinary team methodically "brainstorms" the system design, following the structure provided by the guide words and the team leader's experience [3]. In the study they look for possible deviations from design intent which might have serious consequences. HazOp is a creative technique where a thorough exploration of the design is in focus.

No matter how good a methodology may be, it still has to be managed by people. The HazOp methodology is a tool for a team, to identify hazards. As for any tool, its power is for naught if the persons using it does not know how to use it. The outcome of an analysis is therefore a combination not only of the of the methodology applied on the case at hand, but also of the practitioners' skills and knowledge in utilising the methodology; their ability to put it into practice. *"A HazOp is carried out by a team and is successful only if the team is well composed and well led[6]."*

A problem with HazOp today is that all the experience about the methodology and the practice fully depends on the knowledge of the persons involved in the team, and does not take advantage of the knowledge available in the organization. The members of the HazOp team are usually a part of the organization, but the organization can contribute with information no individual member is in possession of. Taking advantage of experience in an organization, a HazOp team can make sure that many aspects are taken into consideration during the project.

1.3 Purpose

The purpose of this paper is to test what effect a software tool for reuse of knowledge can have on a HazOp study.

This project will continue the work of Kristian Marheim Abrahamsen, exploring how reuse of knowledge can be applied in HazOp projects to achieve better results. In his work he suggested that a software tool could support exchange of experience between projects [1]. To the best of the authors' knowledge, no such tool exists to date. Abrahamsen set out a framework in the form of a requirements specification which such a program should adhere to. In this paper, this requirements specification is expanded on and implemented into a fully functional tool for assisting the HazOp process.

To get empirical quantitative information on what effect such a program might have, an experiment will be executed and the results from this will be analysed. The experiment will consist of groups of students conducting HazOp on simplified cases taken from the real world, both with and without the implemented program. This provides statistical data that will either support or undermine the theory that using a software tool to assist the HazOp process leads to more efficient identification of hazards in systems.

2 HazOp

This chapter is an introduction to the HazOp methodology. To a large extent it is based on the book "System Safety, HAZOP and Software HAZOP" by Redmill et al. [6]. They claim that *"HazOp is recognized to be a powerful technique, and its power is based on teamwork and a methodical step-by-step procedure."* This means that the organization must ensure that they are carrying out the HazOp process the way in which it is intended. If not, there is a high likelihood that the study will achieve only poor results even if the cost is high *"The title HazOp has been accorded to almost any attempts to identify hazards, often when the most casual approaches were employed[6]."* The reason for this might be that the literature on the subject has been sparse and that the management of the study has been inadequate. As with any creative process, the result often depends on who is involved and how they work. The best and, in theory at least, easiest way to ensure the success of a HazOp is to select the right study leader and give that person access to all necessary resources. The resources needed for a HazOp are for the most part properly skilled people, with time enough to be a part of all the study meetings. The material resources will not contribute significantly to the overall cost of the study, but are just as necessary in order to ensure success.

The study has four sequential stages. These are:

- Initiating the study
- Planning the study
- Holding the study meetings
- Dealing with follow-up work

This chapter will take a closer look at each stage and see what is recommended practice.

2.1 Initiating the Study

Planning is an essential part of a HazOp; *"If a HazOp is to be carried out successfully, it needs to be planned in advance and someone in the organization needs to be made responsible for it [6]."* A study initiator should be selected. This person has the overall responsibility for the study. It is important that the study initiator has a good understanding of the process, and that he or she has the authority to allocate the necessary resources to the project. It can be hard to get the most qualified people in the organization to join the HazOp team as these persons often have other regular duties. The organization must prioritize the study by giving the study leader enough power to make sure that the quality of the team is good enough. The study initiator selects the study leader who is responsible for planning and managing the HazOp.

2.2 Planning the Study

HazOp is a study that identifies hazards and operability problems. *"A HazOp is carried out by a team and never by an individual [6]."* The quality of the HazOp depends on the performance of the selected team; hence the selection of appropriate members is crucial. The study leader is responsible for selecting appropriate members for the team. The members must have complementary skills. Together they must have the technical knowledge that is necessary to complete the study. Because HazOp is a creative technique, the members and their team work are of vital importance to the result of the process. Group dynamics are of great importance for the quality of the study. *"A group which has complementary personalities may work better than a group which has been selected solely on technical ability [17]."* The members must be able to understand the design papers. A member can have more than one role. Typically, the roles are classified as:

- Study leader
- Designer
- User or intended user
- Expert
- Recorder

The study leader is responsible for managing the study. The designer role may be filled by different people depending on which part of the design is presented. It is crucial that the person filling the user role is able to explain operational and environmental issues. If not, this role will not add much value to the study and may even confuse the other team members. The expert role can be filled by one of the other team members, for example the study leader or designer. This would reduce the cost of the study, but it is an advantage to have an extra explorer, particularly when this person is a specialist in a field essential to the study [6]. The study will have several meetings before the group has covered the complete design. Different people can fill a given role in different meetings. The study leader must make sure that the intended group is available at the time the meetings are being held.

2.3 Conducting the Study Meetings

"HazOp is based on the principle that several experts with different backgrounds can interact and identify more problems when working together than when working separately and combining their results [3]." The concept is primarily to make the team brainstorm the hazards of the system based on the design representation. The reason why HazOp emphasizes the design is that a hazard can result from a deviation from design intent. This can be within a single component like a module for SQL queries, or the deviation can occur in an interaction between two components, for instance a database and a program module that tries to update a table.

To get an overview of what a system can do and how, it is important that the design representation covers all concerns of the system's stakeholders. It is impossible to detect any deviation from a design representation that does not exist. Therefore, the team must always be sure that the design representation is complete. If they should see that they lack the necessary information for carrying out their intended work, the designer must return with a complete design representation later.

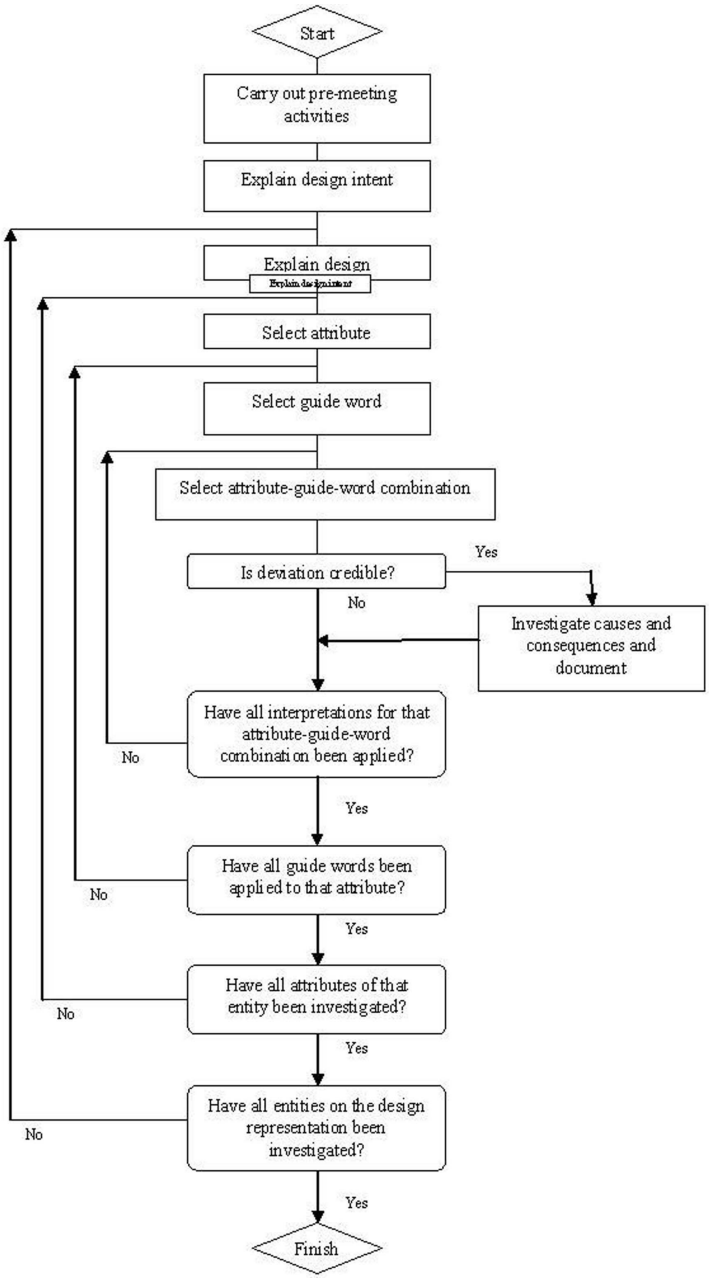

Figure 1: HazOp Meeting Structure

HazOp study meetings have a defined structure for how they are carried out, as seen in figure 1. A study meeting starts with pre-meeting activities. This includes recording the presence of the team members, explaining the meeting rules etc.

The next step is to explain the intention of the design representation which the group is going to study. The study leader selects an entity for further investigation. It is the designer's responsibility to explain the design of that entity.

After this has been carried out, the study leader's next step is to identify one of its attributes for study. An attribute is a relevant property of the entity. This could be response time, data flow etc. The team should then take a closer look at each guideword that could be combined with the selected attribute. *"A guideword is a word or phrase which expresses and defines a specific type of deviation*

from design intent [6].” It is a common practice to have a list of generic guidewords like ”no”, ”more”, ”less”, ”as well as”, ”part of”, ”reverse”, and ”other than”. When it is relevant to examine timing as a part of a HazOp, guidewords like ”early”, ”late”, ”before”, and ”after” should be used.

There are three different ”schools” on how to treat guidewords in a HazOp study. One can either use the list of words ”as is” directly in the analysis, where any special interpretations of the guidewords need to be found at the study meetings. Alternatively, the study leader can work out all the interpretations beforehand in light of the system at hand and present these to the HazOp group. The last option is to work out standard domain-specific guidewords that are brought to the HazOp studies in stead of the standard list. Experience shows that the derived guidewords can be too restricted, and are not flexible enough to bring out the analysts' creativity to the fullest [18].

Table 1 is an example of HazOp sheet row for a helicopter diagnosis project taken from [6]. The system can be represented in many views to cover the stakeholders concern. The HazOp team is interested in the design representation that involves the part of the system which can have consequences for the environment. This could for example be the software modules of a control system for a nuclear power plant; what could happen if the graphical user interface module doesn't work in the intended manner? The study should cover all entities in the design representations and the interactions between them. The team must have a member who is responsible for recording the results from the meeting, otherwise it will be difficult to keep track of what has been done and what remains to do.

HazOp item	Entity	Attribute	guideword	Cause	Consequence/ implication	Question/ recommendation
15	Data Channel handle value to initiate evidence	Data flow	Part of	Generation of messages is event driven and so evidence might be missed	Algorithms fail to recognise a critical event	R15 The criticality of evidence should be considered and critical evidence should be sent respectively

Table 1: HazOp meeting sheet taken from [6]

2.4 Dealing With Follow-up Work

During the study, uncertainties can arise which might influence the work of discovering hazards. These questions must be dealt with so the associated issues can be resolved in subsequent meetings. A HazOp must end with conclusion and recommendations, not questions [6].

The study leader must make sure that all questions by the end of the study are resolved by delegating follow-up work to the appropriate persons who are involved in the design of the system. The study record should be used to verify the completeness of the HazOp. It is hard to give a precise answer to a question which is not well defined; thus, it is important for the team to reach a consensus.

Besides answering questions, the follow-up work could consist of dealing with the recommendations arising from the meetings. Recommendations differ from questions by not demanding an answer.

”The studies which are the subjects of the recommendations may be carried out after the HazOp study has been completed; whether they have been completed may be an issue for a later HazOp, or for a subsequent part of the continuing hazard or safety analysis, but it may not need to detain the current study.” [6]

3 Software Tool for Reuse of Experience in HazOp

Although HazOp for software intensive systems is a well-established technique which has been used by many organizations with good results, there is always room for improvement. Leaving hazards undetected can be disastrous. Every possibility to improve the methodology at a relatively low cost should be investigated further. By utilizing experience from previous projects within the organization, HazOp can be improved. To make this improvement the organization must focus on what information should be stored and how it should be retrieved. A well-designed program could be a solution to the problem. To avoid any misunderstanding, we point out that for this chapter, the term "system" will be used to denote the system being studied, not the software tool used in the study.

3.1 Improvement of HazOp by Use of Existing Knowledge

Experienced team members play an important role in achieving the desired results in HazOps [6]. However, there is no formal process for using existing knowledge within the organization at the beginning of a HazOp. This project focuses on the motivation and software support for such a pre-study. There can be many benefits from taking advantage of existing organization knowledge at the beginning of the study. Learning from experiences is an important factor for attaining the desired quality requirements [16]. Such an activity could be a part of the organization quality culture, a culture where quality is a focus of the software process. Even if the organization has good results with their HazOp projects and feel that they have the necessary human resources, there are several reasons why they should consider building a knowledge base for use in future studies.

One reason is that when a person leaves the organization, the knowledge and experience of that person is lost if knowledge and experience is not systematically structured and stored within the organization [16]. This could be a problem even if the person doesn't leave the organization. The person can simply be unavailable for the HazOp because he or she has other duties to take care of.

Another reason is the limits inherent in the human brain. We are not machines with a stable memory like a computer. There is no assurance that we remember all the things we want to. Long-term memory has a large capacity for storage of information for long periods of time. There is, however, no easy or obvious way to determine the limits of how much can be stored, or for how long it can be stored [7]. Storing information on a hard-disk or in a back-up ensures that nothing is forgotten.

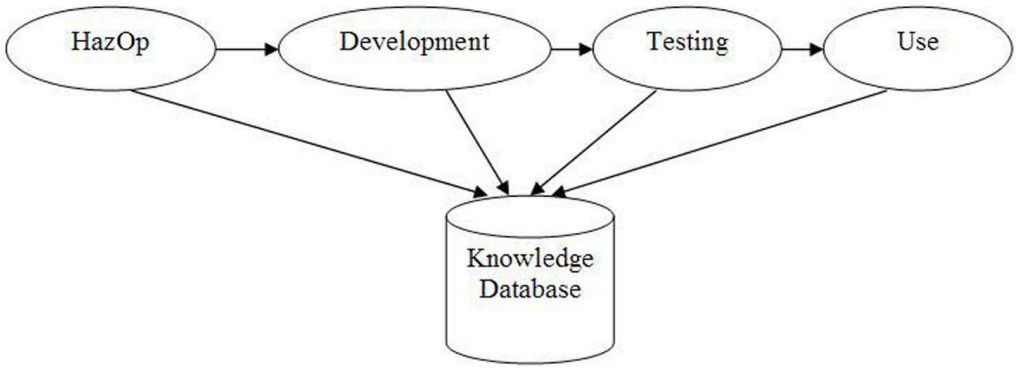

Figure 2: Knowledge Base in HazOp

As figure 2 shows, the knowledge base for the software tool contains information from several parts of the project life cycle. The first thing to do is to record important experience from the HazOp. Such experience could for instance be the consequences of deciding whether to consider multiple design aspects concurrently or sequentially [6]. Designing software systems is a complex task, and during the development phase programmers may experience possible hazards the HazOp team did not foresee. Recording this information can be invaluable. Why did not the study discover the hazard? Was there anything special about the combination of system intent and design representation? What can be done to help us discover such hazards during a HazOp in the future?

The outcome depends on the quality and relevance of the information. If the knowledge base is large, an efficient way to search through the documentation is needed to maximize the benefits of such a pre-study. If too much time is spent reading irrelevant information, the cost of the pre-study would not pay off. The way the organization structures the information can have impact on how an employee searches through the documentation.

3.2 Program Information Base

In order to plan a project we need knowledge and experience. A good plan is crucial for success in a project. By using information within the organization at the start of the project it could be easier to see what kind of design views are required to represent the project. The organization can look at what incidents former systems had and investigate why they occurred. What design representation could express the system in such a way that a HazOp team could have noticed a deviation from design intent? Such information can make the difference between failure and success for a HazOp.

Documenting a project is time-consuming and costly. As an organisation would want to get as much value as possible from such an activity, care should be taken on what information to store. If everything in connection with a project is recorded and documented, it can prove tedious to find the relevant information when planning a new HazOp. It could become too much of a bother making people feel they are drowning in information. Psychologists characterize this as a kind of malady, and call it the Information Fatigue Syndrome. Psychologist David Lewis comments it this way: *"We're often seeing a failure of concentration. We're seeing a loss of motivation, loss of morale. We're seeing greater irritability."* [10]

Another related subject is information overload. Information overload is defined by Mark R. Nelson as *"the inability to extract needed knowledge from an immense quantity of information for one of many reasons [13]."* All these factors can have negative impact on the HazOp planning and might even give worse results than not seeking stored experience. People's emotions impact their performance. *"Even if they recognise their need for information, people often lack the understandings and skills to identify, locate, access, evaluate and then apply the needed information [2]."* The value of information structuring and extraction should not be underestimated, and the problem of information overload should be considered in a software tool for reusing HazOp experience.

Storing too much data can lead to information overload unless the data is handled properly. Information overload increases the cost since the organization will use more time on recording and finding information. Information overload can also make it harder to sift through the irrelevant information to find the things that are relevant. Storing only relevant information/knowledge should be a goal for the organization. The problem is to understand what exactly relevant information is. Furthermore, what is considered relevant today may not be so in the future. The best way to ensure a sound recording and retrieving of information will be to continually improve the processes and artifacts used for this purpose.

Selecting what information should be stored is a hard task, since it difficult to predict future needs. Making sure that all vital information has been stored can make the information database big. Structuring information could then resolve a potential conflict between storing all necessary knowledge and preventing information overload when seeking relevant information.

HazOp focuses on deviation from design intent. The HazOp team examines the design representation looking for possible hazards in the system. An appropriate design representation is crucial for the result of the HazOp [11]. The problem is to know when you have the right representation of the system design. We will use IEEE's recommended practice for architectural descriptions of software-intensive systems [14] as a guideline when dealing with the design representation topic. Here is a brief explanation:

A system has many stakeholders with different interests and concerns. Examples of stakeholders are users, acquirers and developers. The HazOp team is also a stakeholder in safety-critical systems. Viewpoints are used to cover all the concerns of the stakeholders. A viewpoint is a specification of the convention for constructing and using a view. The design representation that the HazOp team is examining can be called "views" in the architecture of the software system. A view is a representation of the whole system from the perspective of a related set of concerns. Every view should conform to a viewpoint, which are ER diagrams for databases, UML diagrams etc.

Adhering to this recommended standard ensures that the design representation the HazOp team will be examining covers the whole system. However, it can be difficult to select the appropriate viewpoints

for the software architecture. Selecting the right viewpoints has consequences for how many hazards will be detected. A view is an abstraction of the system from a certain viewpoint and it is crucial that the abstraction gives the team members a good understanding of how the system has met its requirements.

Learning from previous projects can give a better understanding of which viewpoints to select in order to cover the concerns of the HazOp team. During the study the team might discover that one viewpoint makes it hard to detect hazards. The software tool could then be used to store information on why this viewpoint made it hard to do their work. This experience can result in recommending a different kind of viewpoint for study in similar HazOps in the future, possibly resulting in more detected hazards.

Other topics the program should cover are group selection and what hazards were detected. As mentioned in section 2, group selection has serious impact on the result of the study. Information about what roles and personalities a group should consist of can be valuable when selecting the team. Some projects can have much in common, and hazards that were detected in a previous project similar to the one that is being studied might be relevant to the new system.

Interpretation of guide words is a vital part of detecting hazards in HazOp. As mentioned in section 2.3, there are three ways of treating guide words in a HazOp study. The tool should ideally support all these ways of doing HazOp. A dedicated category for guide word interpretation can make the HazOp participants aware of how the guide words can be interpreted in certain contexts. In this way it can be used by the HazOp team for inspiration on how to interpret guide words in a study meeting, by the study leader when preparing for study meetings by providing insights into how guide words can be interpreted for different types of systems, and via searchable keywords which can be alternative guide words, the followers of the third way of using guide words (a specialised set of guide words per domain) can be satisfied.

There are numerous other topics that can be of relevance for future studies. It is impossible to know in advance all topics that the software tool should support. Trying to build a separate module for every single topic can not succeed because there will always evolve new topics no person was aware of in advance. To overcome this problem the software tool should support customizing user specific modules to cover requested topics. The advantage of this choice is that it gives the users more freedom in structuring their own information. The drawback is that the program will require a higher awareness of what constitutes relevant information and how to structure that knowledge from its users. This awareness will grow with the experience gained by the use of the program. The example topics and how they are used in the program can help the user to create and structure new topics. A technique such as Post Mortem Analysis, PMA, can help the organization structure information and decide what experience should be stored after a HazOp. The PMA concept is described by Stålhane, Dingsøyr, Hanssen and Moe in [19] to be to: *"(...)gather all participants from a project that is ongoing or just finished and ask them to identify which aspects of the project worked well and should be repeated, which worked badly and should be avoided, and what was merely "OK" but leave room for improvement."*

There can be other methods and techniques that can be applied to decide what experience to be stored, and the organization should select one that works for them. The idea is that they are conscious of how they elicit and record experience after a project. In the beginning, when an organization starts using the tool, they might experience that the information base is too small to be helpful in many circumstances. If the system being studied is quite different from those already in the knowledge base, they can feel that the software tool is only wasting their time. Such feelings might lead to the HazOp team disliking the tool, which again will have a negative effect on how the tool is used, thus starting a vicious cycle ultimately leading to the company discontinuing the use of the software tool. To avoid this, the HazOp team leader should look at the description of the projects in the information base to see if there are any projects that have similarities with the current HazOp. If so, the programs search function should be able to yield relevant information. If not, using the program might be inappropriate for that particular HazOp.

4 Software Tool Requirements Specification

The software tool must have a defined scope. This project is not intended to develop a single program that meets every need a HazOp team would ever have. The program will focus on recording and finding information on previous HazOps. As a minimum requirement, the organization should be able to store all the information they deem necessary. This means they should be able to store not just plain text, but also images, video streams, and other data in different formats. In principle it must be capable of storing every file format the organization uses.

Functional requirements are statements of services that the system should provide. The requirements in this section continues the discussion in section 3. They are a more formal way to express what functionality the software tool should contain. Also, the software requirements include how the system should react inputs made by users of the software.

The functional requirements will be illustrated by use cases as used by Martin Fowler in the book "UML Distilled" [5] to see in what context they are relevant. Each use case will be followed by a list of requirements relating to that use case.

Non-functional requirements are constraints on the services or functions offered by the system. Examples of such requirements are performance, usability etc. These requirements are also sometimes called the quality attributes of the system. The non-functional requirements of the software tool will be discussed after the functional requirements.

4.1 Functional Requirements

The functional requirements can be divided into different main functions the program should provide. We illustrate the functional requirements with use cases both textually and figurative. The implemented requirements will be highlighted in sections 5 and 6.

4.1.1 Main Functions of the Software Tool

The main functions of the software tool are to store information and later find that information so that more hazards may be discovered. Being conscious about how the knowledge should be structured will make it easier to find relevant information. The example topics in the program: "design representation", "group selection", "guideword interpretation" and "detected hazards", will illustrate how the knowledge can be structured. All stored information must be linked to a project so it can be read with an understanding of the context.

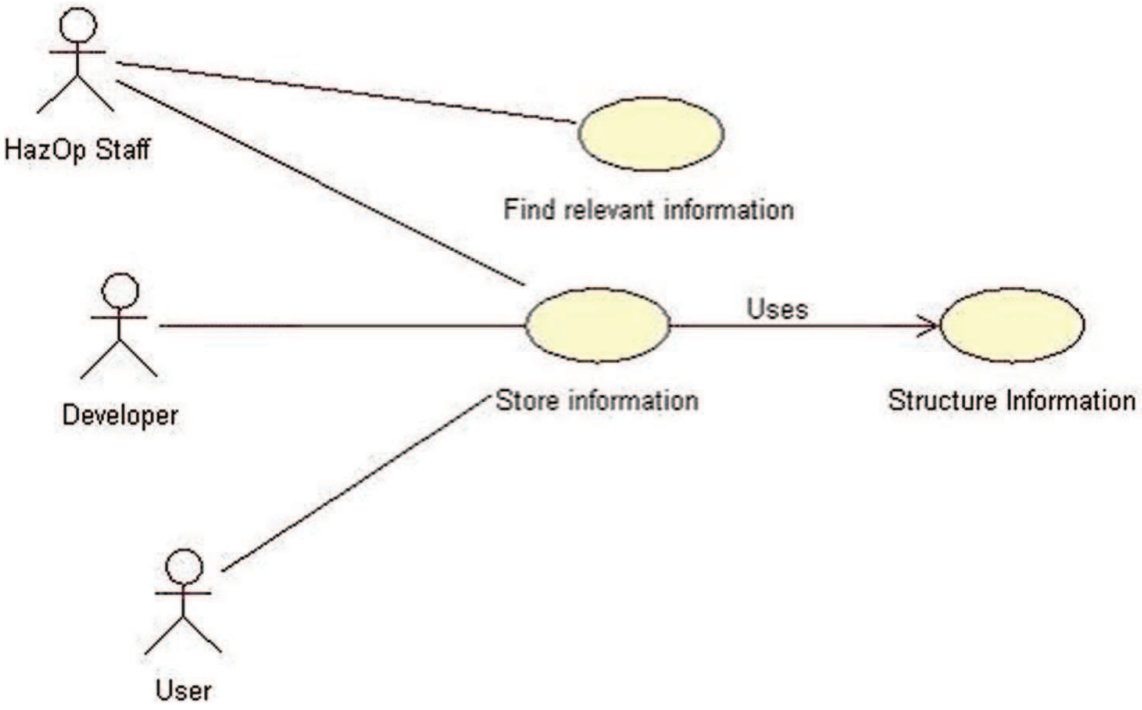

Figure 3: Program Use Case

4.1.2 Selecting Project

The program is intended to store experience related to specific projects. All problems, reflections and observations must be seen in a context if a good understanding is to be achieved. We define the context needed to properly make sense of the experience to be the "system project". The system project includes the HazOp, development, testing, maintenance and operation of the system.

It is not enough to simply relate the experience to just specific HazOps as there might be information valuable to a HazOp even if it has been discovered in other parts of the system life cycle.

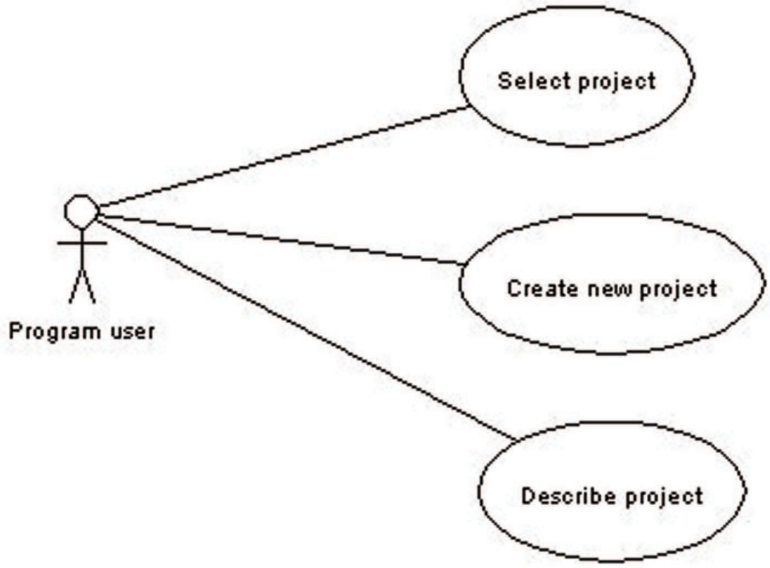

Figure 4: Select Project Use Case

Use Case Name	**Select project**
Actor	**Program user**
Summary	The user wants to select a project the experience can be related to
Basic Course of Events	1. The user creates a new project and gives it an appropriate name
	2. The user fills in information to describe the project
Alternative Paths	In step 1, the user instead selects an existing project. If requested the user can edit the field that describes the project.
Trigger	The user wants to store some information and must relate this to a specific project.

Figure 5: Select Project Use Case Table

- F-1 The user must be able to create a new project
- F-2 The user must be able to describe a project
- F-3 The user must be able select an existing project

4.1.3 Storing Information

When the user wants to store information, the program must ensure that it is properly catalogued. Each topic should have a distinct category. This is done by forcing the user to select an existing category or create a new one if none of the existing one can be used. Each category should have a certain structure. Examples of categories are Design Representation, Group Selection, Detected Hazards and guideword Interpretation. This can be accomplished by having dialog boxes where the users fill in the appropriate information.

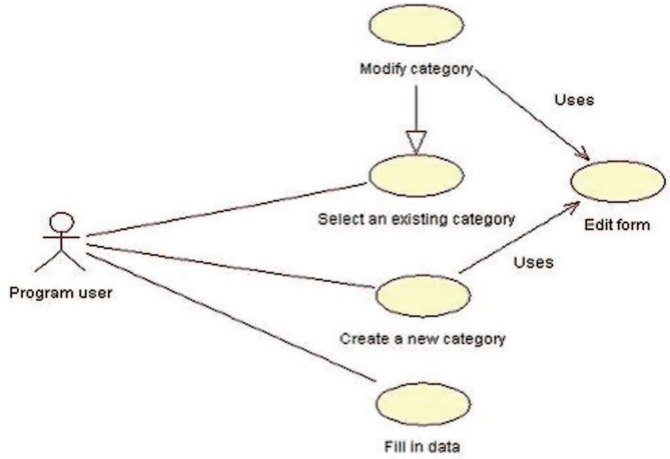

Figure 6: Store Information Use Case

Use Case Name	**Store information**
Actor	**Program user**
Summary	The user wants to store information that can be used in future HazOps. The user must figure out how the information should be structured and catalogued. After an appropriate category has been selected or created, the user fills in all the data that needs to be stored.
Basic Course of Events	1. The user creates a new category
	2. The user specifies which fields the category form shall have by editing the category form
	3. The user fills in the data in the fields
	4. The user requests the program to save the data
Alternative Paths	In step 1, the user instead selects an existing category, jump to step 3 or in step 3, the user adds one or more fields to the category form before filling in the fields
Exception Paths	In step 1 the user tries to give the category a name that is already taken. The program tells the user to give the category another name and explains why the first name did not work.
Trigger	The user wants to store some information

Figure 7: Store Information Use Case Table

- F-4 The user must be able to create a new category
- F-5 The system must have no categories which share the same name
- F-6 The user must be able to modify an existing category
- F-7 The user must be able to fill the category form fields with appropriate data
- F-8 The user must be able to save the data

4.1.4 Editing Category Dialog Box

By letting the user specify the dialog box of a category, the program becomes more flexible. The dialog box is made up from user-specified fields. The user can set constraints on what kind of data a field can contain, the default being the string format. Examples of data formats can be strings, integers or certain file types such as pdf, doc, gif, mpeg etc. The purpose of constraining the categories dialog box fields is to enforce that the information is structured in a standard way. If the users specify certain fields when searching for information they can use these constraints to get a better search result. The program must give the users the opportunity to remove or rename fields. The dialog box editing has many similarities with editing tables in databases.

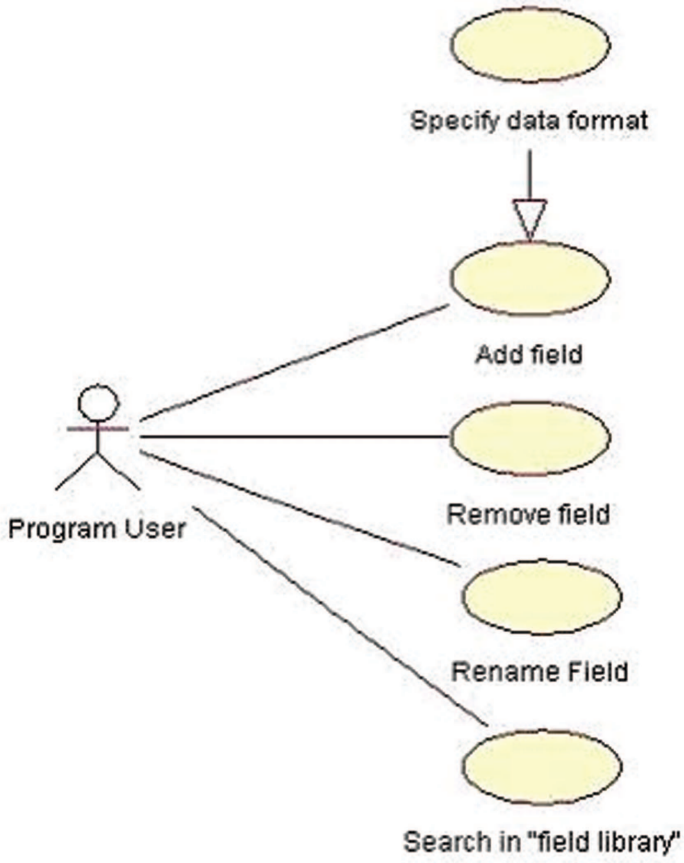

Figure 8: Edit Category Use Case

Use Case Name	**Editing a category dialog box**
Iteration	**Filled**
Summary	The user edits a category dialog box by specifying which fields the dialog box shall contain. This is done by adding, removing and renaming fields. The user set optional constraints on the fields by specifying what data format they should have. A *field library* will support reuse of existing fields.
Basic Course of Events	1. The user adds a field to the category dialog box and gives this field a name 2. The user removes a field from the category dialog box 3. The user renames a field in the category dialog box 4. The user adds a field to the category dialog box by selecting an existing field from the *field library*.
Alternative Paths	In 1 the program user can specify the data format for the field. This editing task can be done by combining each step one or more times in any desired combination.
Exception Paths	In step 1, 3 and 4 a name conflict may occur and the program tells the user that the every field in the dialog box must have an unique name
Trigger	The user wants edit a category dialog box

Figure 9: Edit Category Use Case Table

- F-9 The user must be able to add fields to the category dialog box
- F-10 The user must be able to remove fields from a category dialog box.
- F-11 The user must be able to specify a fields data format
- F-12 The user must be able to rename a field
- F-13 The user must be able to select fields from a field library that can be added to the category dialog box
- F-14 All fields in the category must have a unique name

4.1.5 Finding Information

When the users want to find information it is important that they have functionality which can be used to filter the search so they can avoid information overload. This can be done by selecting a category, selecting some keywords from a fixed list, and optionally associate these words with certain fields in the dialog box of the selected category. The reason why the users select keywords from a list is that this can force the different users to have a common understanding of how information should be described. This will hopefully make the search result better.

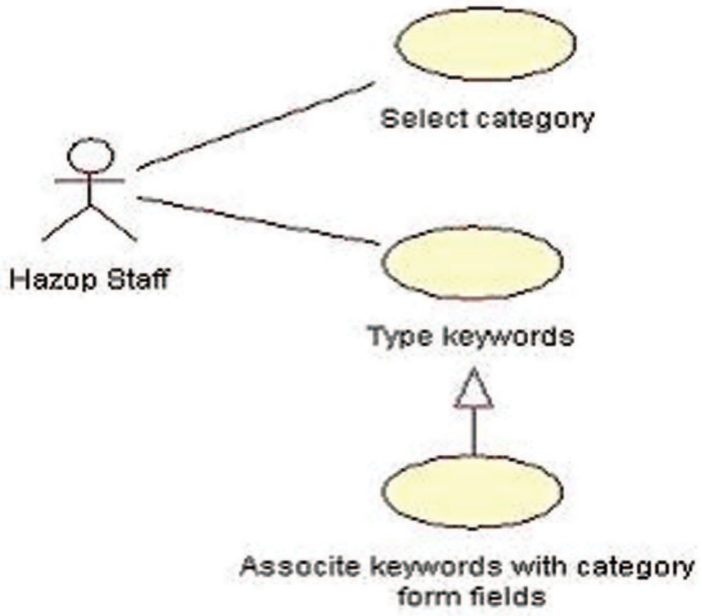

Figure 10: Find Information Use Case

Use Case Name	**Find information**
Iteration	**Filled**
Summary	The user wants to find information that can be used to find more hazards in the HazOp. The user selects one or more categories which may be relevant for the search. Keywords, optionally associated with certain fields, can be filled into the search dialog box for achieving a better search result.
Basic Course of Events	1. The user selects one or more categories 2. The user selects keywords which constrain the search 3. The user instructs the program to find information to match the specified search constraints 4. The program returns a result list from the search
Alternative Paths	In step 2, the user associates the keywords with certain fields jump to step 3.
Exception Paths	In step 3, the user adds one or more fields to the category dialog box before filling the fields
Trigger	The HazOp team wants to take advantage of stored knowledge to detect more hazards.

Figure 11: Find Information Use Case Table

- F-15 The user must be able to select one or more categories
- F-16 The user must be able to specify keywords for the search
- F-17 The user must be able to select keywords from a list.
- F-18 The program must return a list of hits after the search
- F-19 The user must be able to browse through the search result

4.1.6 Design Representation Category

If the users do not have any relevant category dialog boxes to look at and work with in the beginning they might avoid using the program. There are some categories which can be useful for any kind of HazOp. One of these categories is design representation. A good design representation is crucial for achieving success with the HazOp. This is because the HazOp staff discover hazards by systematically working through the design representation looking for deviations from design intent. Bad material will make the work harder. A minimum requirement is that the design representation covers the whole system. Another requirement is selecting the best way to represent a system when carrying out a HazOp. This will depend on the system represented. Even if there is a good selection of viewpoints there might be some aspects the HazOp staff should be aware of working through that part of the design representation. Some hazards may be hard to find no matter what choice of viewpoints has been made. Reading guidelines could then help the group studying the design representation.

The design representation category dialog box should cover all these issues. The problem is to have fields that are not too specific, since this may restrain the expressiveness of the users. Also, if they are too generic, information overload can be a problem since the user might get more information than required when they read through the search result.

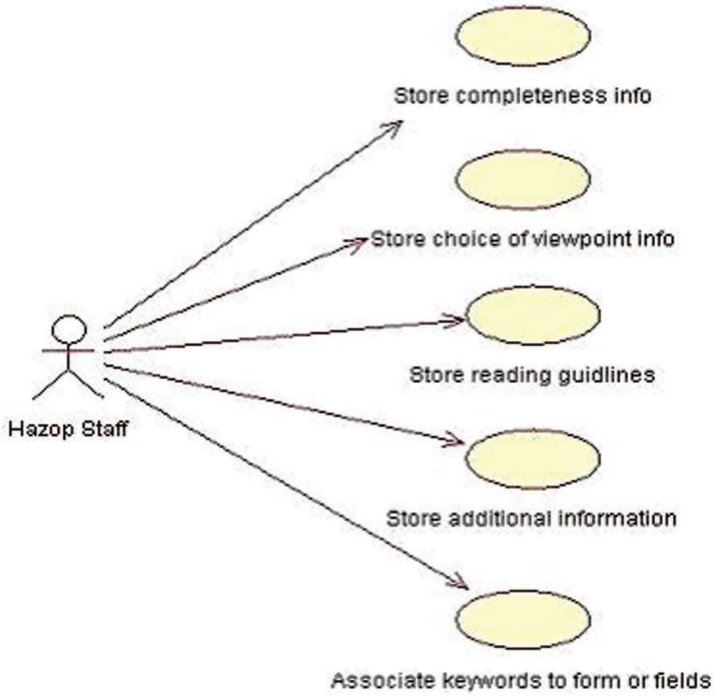

Figure 12: Design Representation Use Case

Use Case Name	**Store information about design representation**
Iteration	**Filled**
Actor	Program user
Summary	The user wants to store their experience about design representation and the HazOp process.
Basic Course of Events	1. The user fills in experiences he or she had about completeness of the design representation 2. The user fills in experiences he or she had about selection of view points in the representation 3. The user fills in experiences he or she had about working through different design representations 4. The user fills in any additional experiences he or she had about the problems related to design representation 5. The user associates keywords to the different fields or category dialog box
Alternative Paths	It does not matter in which sequence the user fills in information, but he or she can not associate keywords to fields that are empty
Trigger	The user wants to store information about issues he or she has experienced related to the design representations.

Figure 13: Design Representation Use Case Table

- F-20 The user must be able to fill in information in the field for completeness (of the design representation)
- F-21 The user must be able to fill in information in the field for selections of viewpoints
- F-22 The user must be able to fill in information in the field for reading guidelines
- F-23 The user must be able to fill in information in the categories field for additional information
- F-24 The user must be able to associate key words from a fixed list to fields that are non-empty
- F-25 The user must be able to fill the fields with data on the formats string, picture files, movie files and document files.

4.1.7 Group Selection

A good team is crucial for achieving success with a HazOp. The groups performance depends on each members skills and knowledge combined with how well they work as a team. Before a HazOp can start there must be a group selection. By having information on what expertise a group should have for a particular kind of system, the study leader is better suited to select the right team. Not only information about the roles can be of importance for the organization, but also knowledge of the skills and experience for each person who can have the different roles can help the organization selecting the right people.

A groups performance depends not only on each members contribution but also how they work together. The combination of personalities has impact on how the group works as a team. After a study it might be concluded that the personality composition of the HazOp staff was too homogenous. Lack of certain personalities can also be a problem.

After a HazOp, an evaluation may suggest that the group lacked certain roles. For example during the development a software engineer might detect some difficulties that can lead to a hazard. During an evaluation of what could have been done to detect this hazard the conclusion can be that the group lacked an expert on the topic UML state diagrams.

If the organization is large, different roles can be filled by many individuals. Stored data of each persons experience can help the organization in selecting the right person for a particular role. Information about the employees personalities can also be of interest. The program user must be careful when evaluating the persons who were involved in the HazOp as the staff might feel uncomfortable about this, and can this could have a negative impact on the working environment.

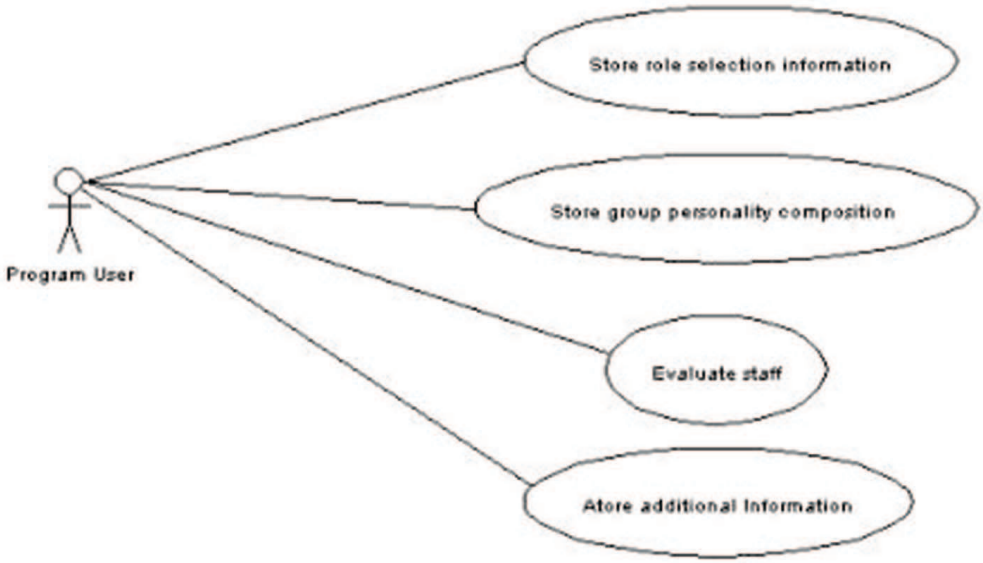

Figure 14: Group Selection Use Case

Use Case Name	**Store information about group selection**
Iteration	**Filled**
Actor	Program user, usually the study leader.
Summary	The users store their ideas on the optimal group composition.
Basic Course of Events	1. The user fills in experiences he or she had about the role selection. 2. The user fills in experiences he or she had about how different personalities influence the team work. 3. The user evaluates every person who worked in the group by selecting the persons from the organisation employee list within the program, and fill in all relevant information about this person's contribution to the HazOp. 4. The user fills in any additional experiences he or she had of the group selection. 5. The user associates keywords to the different fields or category dialog box
Alternative Paths	It does not matter in which sequence the user fills in information but he or she can not associate keywords to fields that are empty
Trigger	The user wants to store information about issues he or she has experienced related to the group selection.

Figure 15: Group Selection Use Case Table

- F-26 The user must be able to fill in information in the field for role selection
- F-27 The user must be able to fill in information in the field for group personality composition
- F-28 The user must be able to evaluate every person that was a part of the HazOp group.
- F-29 The user must be able to select the persons that should from an employee list within the program
- F-30 If the user wants to evaluate a person that is not on the organization employee list the user should be able to add persons to the list.
- F-31 The user must be able to fill in information in the categories field for additional information

4.1.8 Detected Hazards

Hazards that were detected in a certain system and environment are likely to appear in similar systems and environments. By comparing the system to be studied with previous ones the HazOp group can see if the hazards that were detected are relevant to the study. The category should contain a short heading for each hazard and a more detailed description. Information about which environment it belongs to can also be valuable since a system might operate in several environments. The program user should have the opportunity to browse all the documentation for the HazOp included the design representation. This documentation would be files from external programs since this program does not support recording of the study meetings. If the HazOp group uses this information as checklist when reading the design representation, that may have a negative effect on the creativity. Looking for hazards that appeared in similar systems and environments should therefore first take place after the ordinary inspection.

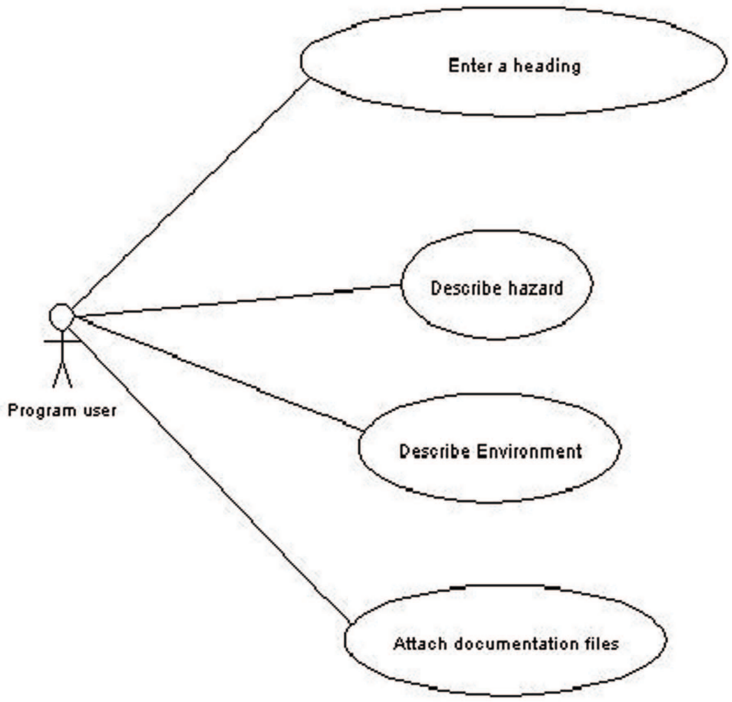

Figure 16: Detected Hazards Use Case

Use Case Name	**Store information about detected hazards**
Iteration	**Filled**
Actor	Program user
Summary	The user stores all information about the hazards in the system.
Basic Course of Events	1. The user instructs the program to add a hazard to the project. 2. The user fills in an appropriate text in the hading field. 3. The user fills in information which describes the hazard. 4. The user fills in information which describes the environment the hazard belongs to. 5. The user adds files that give more information about the hazard. 6. The user associates keywords to the different fields or category form
Alternative Paths	It does not matter in which sequence the user fills in information but he or she cannot associate keywords to fields that are empty
Trigger	The user wants to describe all hazards and their environment.

Figure 17: Detected Hazards Use Case Table

- F-32 The user must be able to fill in information in the hazard heading field
- F-33 The user must be able to fill in information in the field for hazard description
- F-34 The user must be able to fill in information that describes the hazards environment.
- F-35 The user must be able to add any documentation files which are relevant for the hazards.
- F-36 The user must be able to associate keywords with every field in this category.

4.1.9 Guideword Interpretation

Interpretation of guidewords is a vital part of the HazOp. Some interpretations can be relevant in other HazOps. These interpretations can be creative and not immediately obvious. Being aware of such interpretations might help detecting more hazards. To get a good understanding of the interpretation, information about the context is required. Additional information might also be useful.

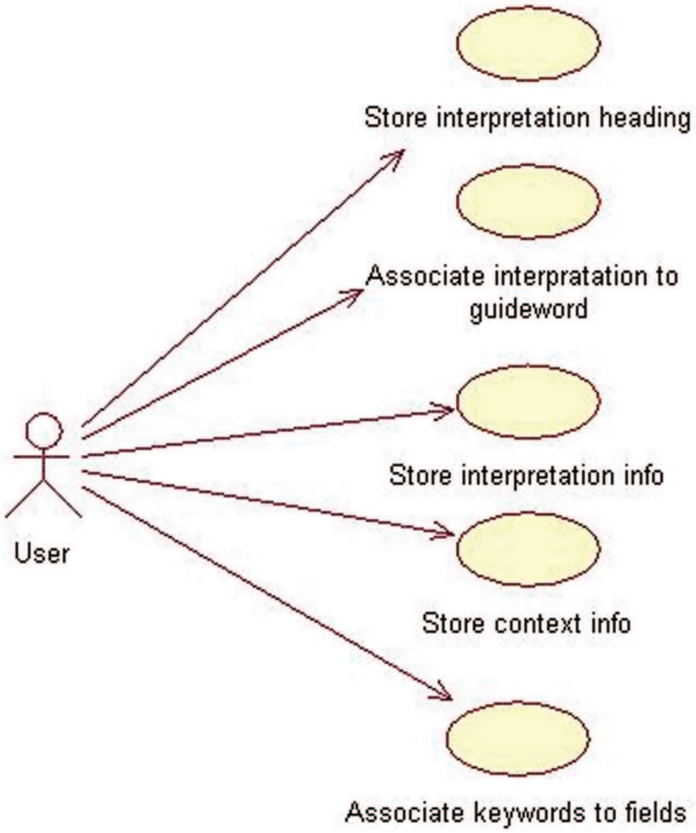

Figure 18: Guideword Interpretation Use Case

Use Case Name	Store information about guideword interpretation
Actor	Program user
Summary	The user stores all information about certain guideword interpretations
Basic Course of Events	1. The user instructs the program to add a guideword interpretation to the project. 2. The user fills in an appropriate text in the interpretation heading field. 3. The user associates the interpretation with a guide word from a list. 4. The user fills in information which describes the interpretation. 5. The users fills in information about the context for the interpretation 6. The users fills in additional information about the interpretation 7. The user associates keywords to the different fields or category form.
Alternative Paths	It does not matter in which sequence the user fills in information but he or she cannot associate keywords to fields which are empty
Trigger	The user wants to describe some guideword interpretations

Figure 19: Guideword Interpretation Use Case Table

- F-37 The user must be able to fill in information in the interpretation heading field.
- F-38 The user must be able to associate the interpretation with a guideword from a list.
- F-39 The user must be able to fill in information in the interpretation field.
- F-40 The user must be able to fill in information in the context field.
- F-41 The user must be able to fill in information in the additional information field.
- F-42 The user must be able to associate keywords with every field in this category.

4.2 Non-Functional Requirements

There are four main non-functional requirements which must be taken under consideration when implementing the software tool:

1. **Learnability -** It should be relatively easy to understand how to use the tool. The user should not spend too much time learning the program. If the user have problems with understanding how the program works, it might become discarded.
2. **Usability -** Using the program should be as effective as possible. If the user must invest a lot of time and effort into getting the information from the program, they might think that it is not worth the bother, ultimately discarding the tool.
3. **Maintainability -** The program must have an architecture that enables the possibility to extend and maintain it. This is important since when a program is being used, people often see things which could have been done different and better. It is also likely that users want to add new functions which the implementers did not think of.
4. **Reliability -** The program must should be stable and crash as little as possible. If the users do not feel they can depend on the program, they might not use it because it poses a risk to only wasting their time. Additionally, the tool should not lose any data as this kind of data is potentially a very valuable resource to organisations.

5 Software Tool Design

5.1 Introduction

When designing the software tool it was important to consider two factors. Firstly, the ease with which the software could be made to implement all the requirements from the requirements specification. Secondly, the necessity to implement the software quickly, leaving enough time to do the experiment. These two factors were found to be mutually exclusive, so a compromise had to be made. The basis of the software was made general enough to accommodate later increments implementing the more advanced functionality in the software requirements specification. The software, however, was made for the requirements in section 4. The single most significant deviation from the requirements was the ability to design new categories in which to store knowledge, and the ability to modify existing ones. This functionality was left out in order for us to be sure we would have the tool ready in time for the experiment. Since this was left out, the tool that was created can not be classified as anything but a prototype on which further development is possible.

5.2 Architecture

The software tool was implemented using a three layer architecture. A cleanly layered architecture makes it easier to replace parts of the program without having to restructure the entire program each time a change is made [9]. There is a separate database-access module, a separate data object model, graphics module and controlling module.

For the rest of this section of the document, the term 'Project' will be used to denote HazOp studies that has been done, for which information is to be stored in the HazOp Tool.

5.2.1 Realised Architecture

The following is an overview of the design that was implemented. The program consists of a project

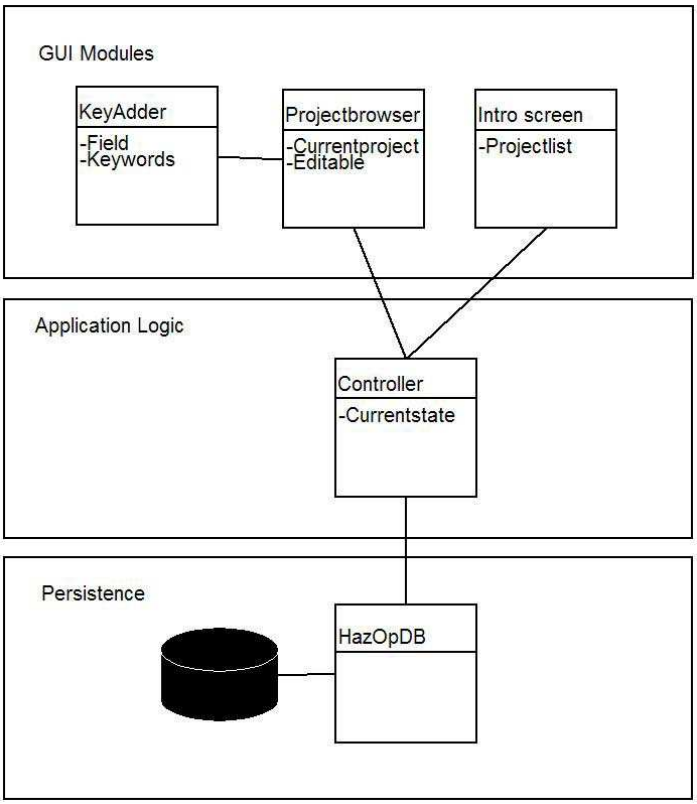

Figure 20: The Architecture of the implemented HazOp Tool.

browser that is used both to input data into the database, and to provide data to the user in an easily navigable form. The project browser is controlled by a controller that handles changes to the state of the program. The controller is also the link between the browser and the database module. The database module acts as a server which provides and stores data on request from the controller. When starting up, the program first collects all the data from the database and stores this data in an object model. This model is then accessed and modified by the project browser when the program is in use. Finally, when the program is shut down, it stores all added and modified data into the database, and deletes all data that has been deleted by the user. This is suboptimal at best, but it made for rapid development. A better solution would be to load data only when needed. The participants of the experiment were not inconvenienced by this, however, since the database was small at the time of the experiment.

The category management part of the tool was not implemented, but three hard-coded categories were provided. These categories were the ones deemed most appropriate and helpful for this experiment. The categories were "Design Representation", "Guide Word Interpretation" and "Hazards Found". Since only three categories were used in the experiment, the workload involved in hard-coding them was far smaller than the workload required to make a general system, capable of handling any number of categories.

5.2.2 Ideal Architecture

The following is what we believe to be an optimal design, offering a general solution to satisfy a vast amount of categories. Due to constraints on time, however, it was not implemented.

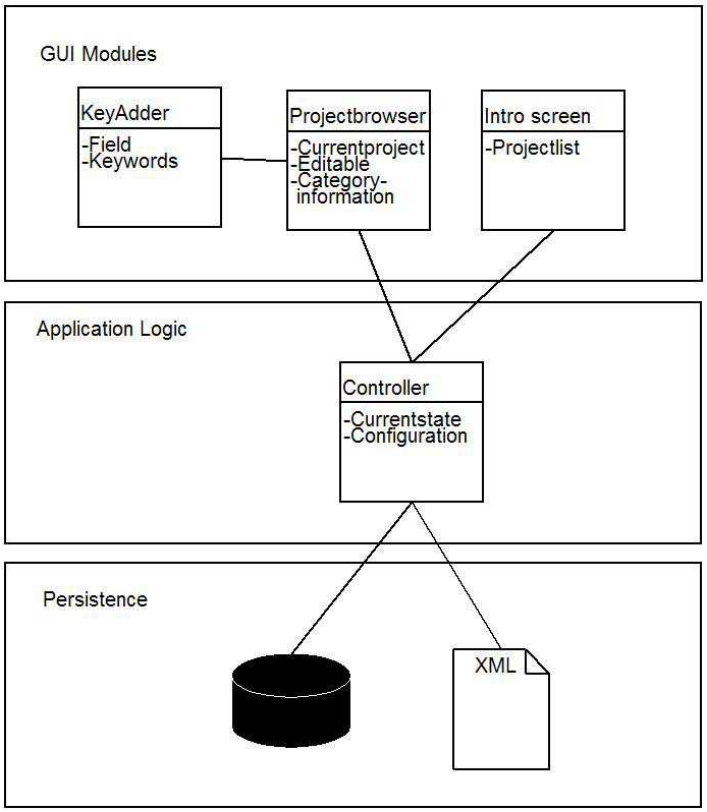

Figure 21: The Architecture thought to be ideal for the HazOp Tool.

The program consists of a form editor, which creates and edits xml-forms. These forms are templates for the different categories. It also modifies the database to incorporate the changes made. When the other parts of the program are loaded, these xml-forms are read and the categories and their database-linking are rendered from them. Thus, at compile time, none of the categories might exist, these are created and modified later by users. The program therefore consists of a module to interpret these XML-files and feed information to the graphical layer.

The graphical layer generates a graphical user interface (GUI) at runtime, based on the categories in any given project. When editing projects, new categories can be added to that project from the list of already existing categories or ones already present can, along with all their data, be deleted from a project.

XML files are used to keep track of the following: Which categories exist, what fields they have, what each field is called in the database, whether any field is a list of constants, and in that case whether that list can be added to. In effect this gives the program enough information to give the users the latest categories with their respective data for any project stored in the database without the need of foreseeing such information at compile time.

The XML files are themselves put into the database and are the first things the program reads from the database. This then sets up the program to use the rest of the database.

5.3 Data Model

In order to reuse data in later increments of the software, the database was created as general as possible. All textual data found in the software is located in "fields" These fields can have keywords attached to them via a key list, and files attached to them. Each field is attached to one and only one project via categories. The idea is that as the program evolves more categories will be made through a user interface, and the program will keep stock of which categories exist.

The implemented data model is shown below

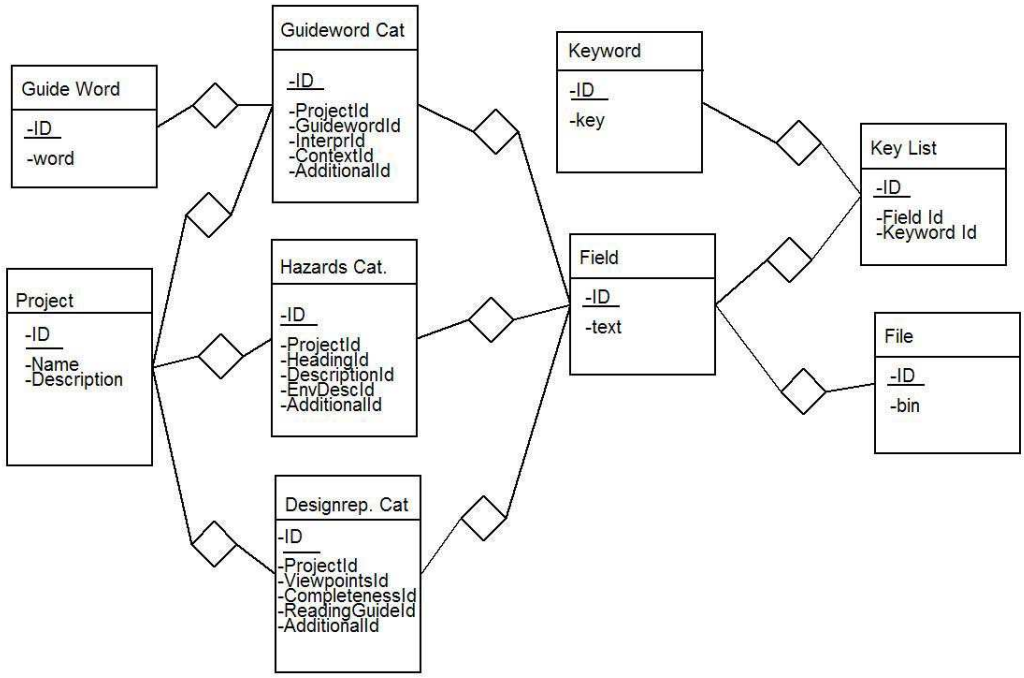

Figure 22: Data model of the HazOp Tool as implemented.

For practical reasons, we made an object model of the data as an interface between the database and the user interface. The main difference between this model and the database model is that the fields only exist as strings within the categories. The categories themselves keeping stock of all its fields and their key lists and files. This was done to be able to work on separate parts of the program concurrently without the need for database access. Unfortunately, this led to the necessity of a lot of extra code for database management. Currently, this code resides in one class of about 1000 lines. In order to add even one category, one would have to add about 150 lines of code at various places in this class. This is suboptimal at best, but in order to guarantee a prototype within the time we had to work, this was the safest option.

A completely general data model was also designed that differs only minutely from the implemented one. Instead of having a special "guide word" table, a more general constant list is used. This list offers the same functionality for the guide word category that the guide words did, as well as being available to other categories where a list of constants might be needed. This could be a list of employees, a list of grades etc. The constant list does not even need to be "constant". A variable set in the XML files which stores the category information and are read at startup signifies whether the list can be changed or not. In this way, several different categories may share the same list of constants if required.

The data model thought to be ideal and general is shown below:

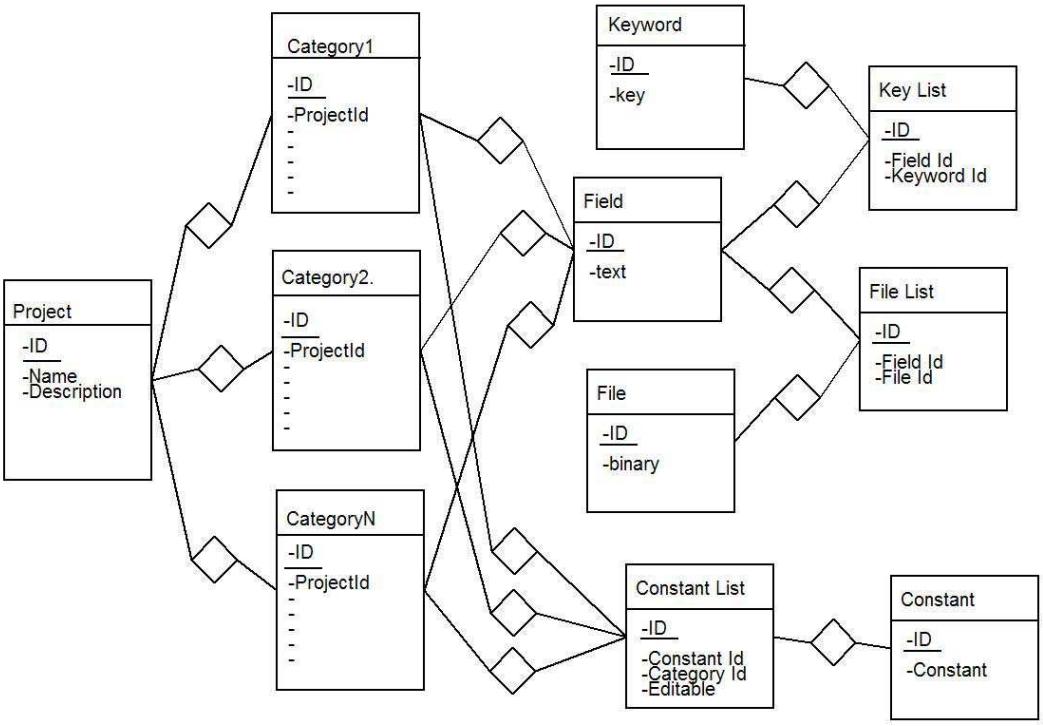

Figure 23: Data model of the HazOp Tool as thought to be ideal.

5.4 Example of Use

The following is an overview of the functionality offered by the software tool. The tool is structured as a set of windows forms

5.4.1 Introduction Screen

Figure 24: Screenshot from the Introduction Screen in the HazOp tool

The HazOp tool implemented welcomes users with an introduction screen in which users can choose the desired mode of operation. Either the users can view projects in a safe mode in which no data can be modified or they can edit projects already inserted into the system. A last possibility is to make a new project that can be filled with new data. As a safety feature, it is impossible to delete projects from this interface. That functionality should be left to managers as it could potentially mean the loss of valuable knowledge. Since a separate interface for managers was not called for in our experiment, we did not implement such functionality. The only way to delete projects in the tool as it stands is by manually editing the database.

5.4.2 Design Representation Screen

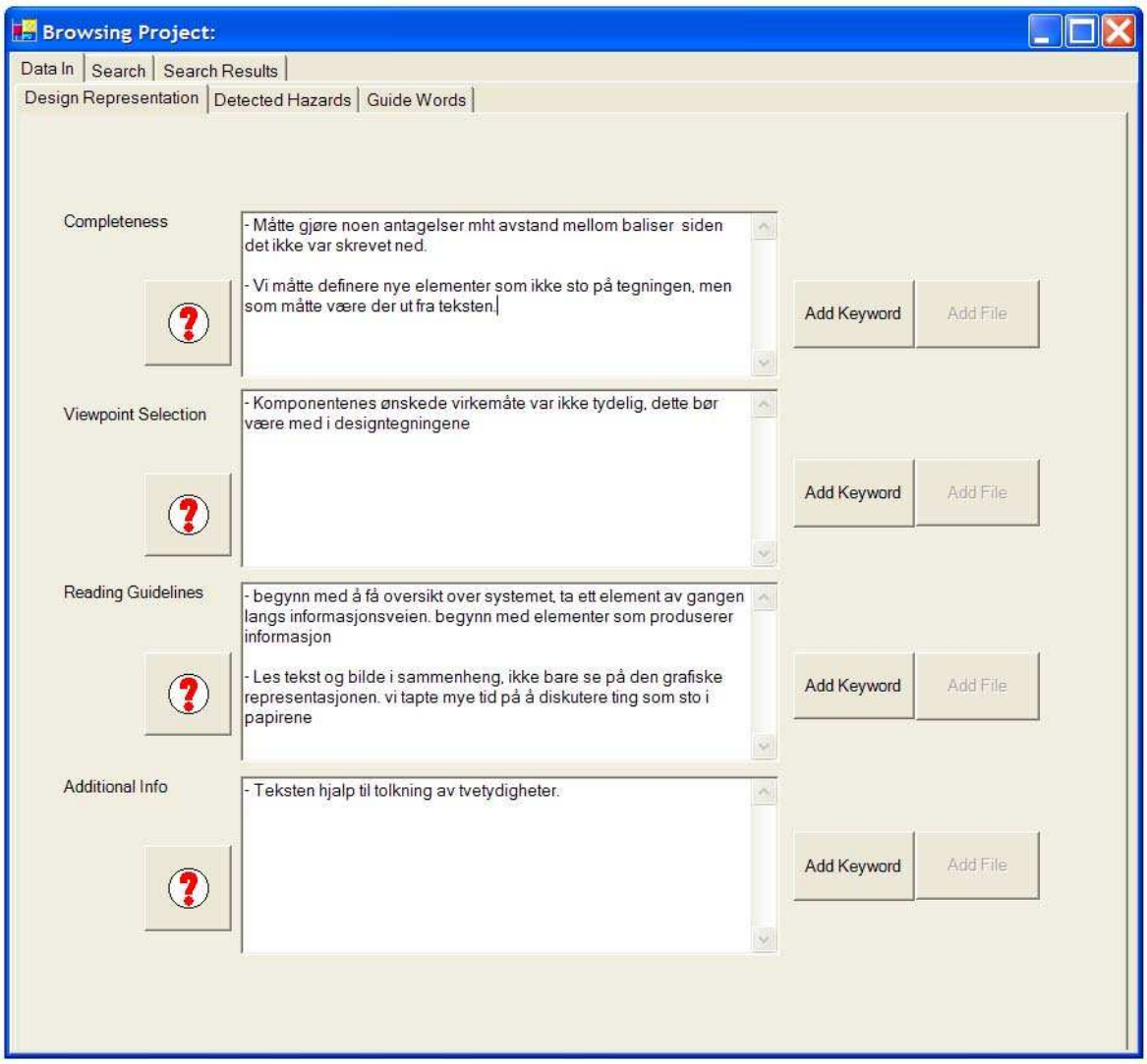

Figure 25: Screenshot from the design representation screen

When entering the project browser, the user is presented with three main tabs, "Data in", "Search" and "Results". These are linked to the two main functionalities of the tool; inputting data into the HazOp tool and searching through the data already in the tool. When inputting new information into a project and when only browsing projects, the users will be in the "Data in" section of the program. Within this tab a new tab-selection is found in which all the categories already used for the project can be found. For the system we made, this was limited to the three categories already mentioned. The user can freely switch between these and edit them if in edit mode.

5.4.3 Detected Hazards Screen

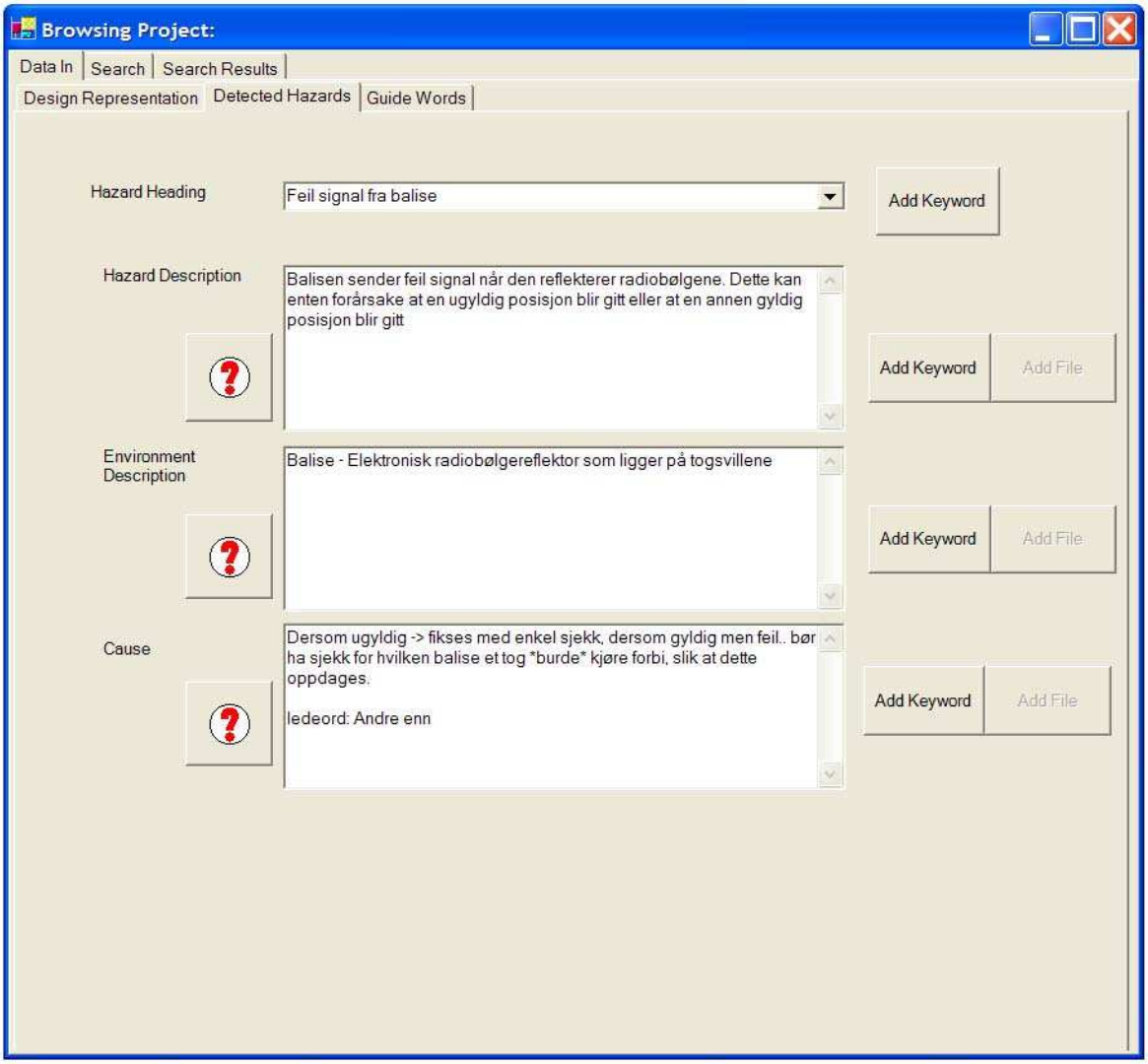

Figure 26: Screenshot from the hazards screen

Each category may have one or several branching points. Branching points are places where later information depends on the value at that point. In practice this is done through the use of Combo Boxes. When a new value is inserted, all fields below are blanked out and ready to accept new data. When selecting an already created value, the fields below the branching point are filled with the data for that value.

5.4.4 Guide Word Screen

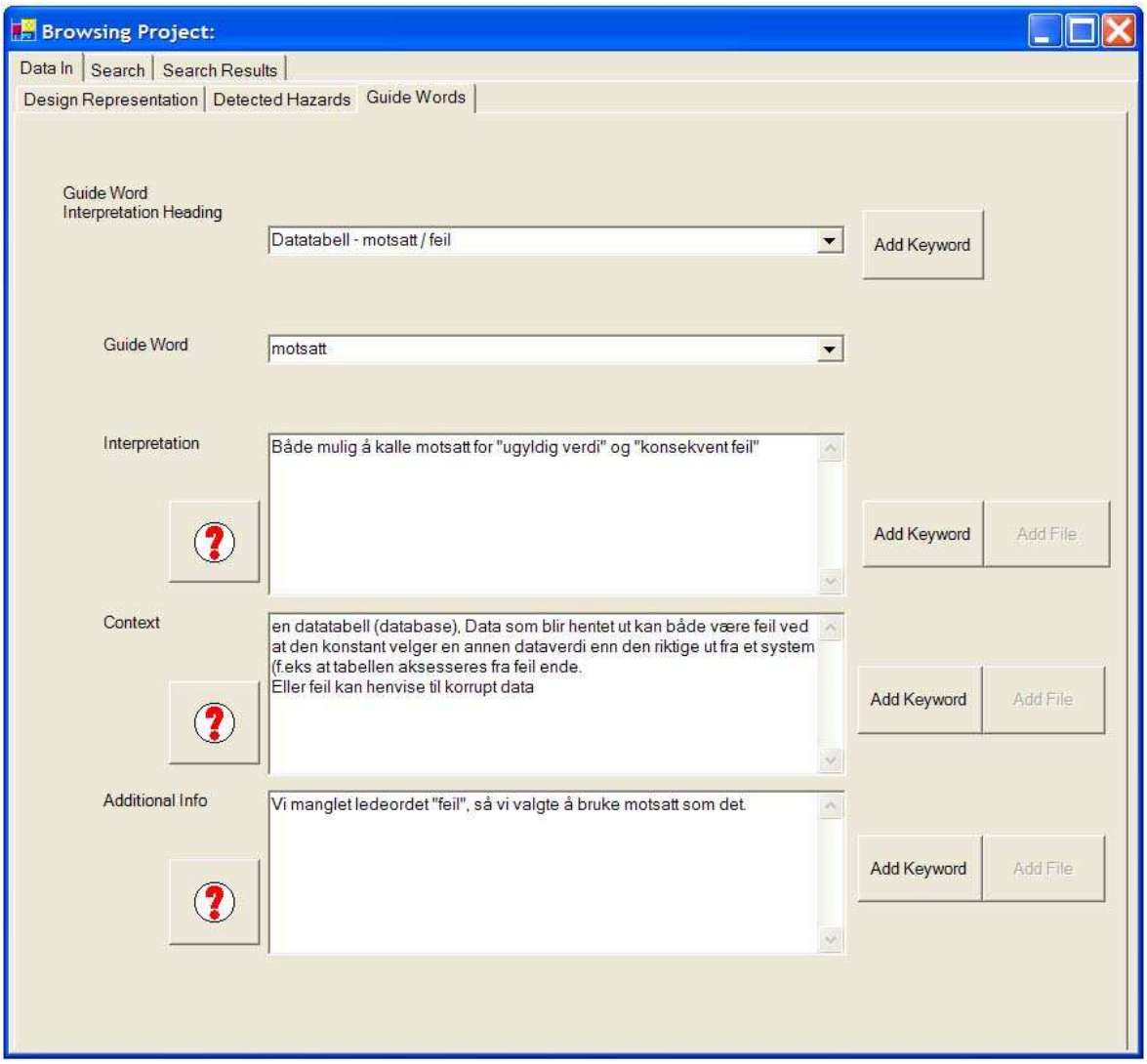

Figure 27: Screenshot from the guide word interpretation screen

In our implementation, two such branching points were made, one in the "Hazards Found" category where all the fields were dependent on which hazard the user was looking at any given time. The other was located in the Guide Word Interpretation category.

5.4.5 Key Adder Screen

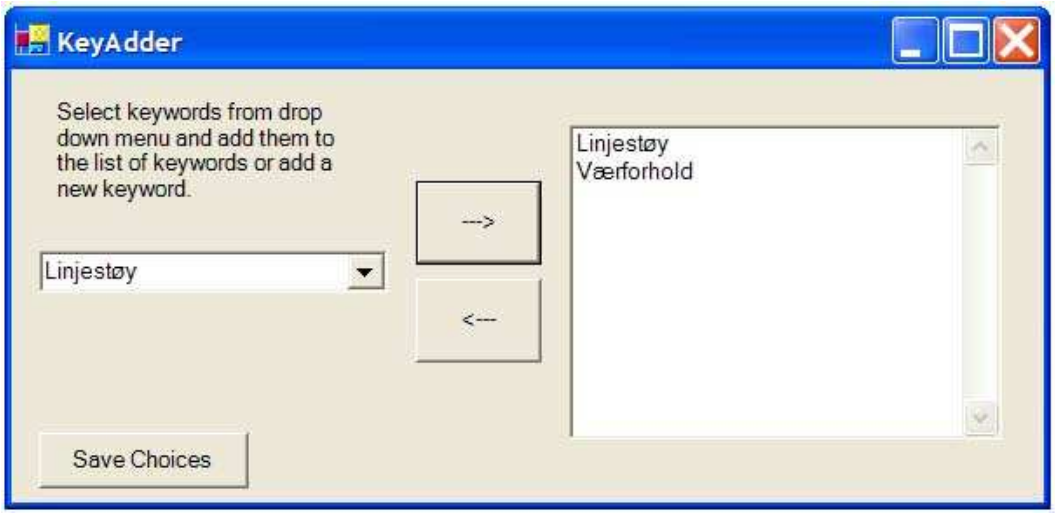

Figure 28: Screenshot from the keyword adding screen

When editing projects, keywords can be added to any field via the "Add keyword" button next to each field. When pressing this button, an overview of all keywords used so far in any project is presented to the user, and the user may decide whether to add a new keyword or use an already existing keyword. This is done to ease the search functionality. Too many similar keywords will make searching harder as all synonyms would have to be chosen to ensure a complete search of the database. The user is encouraged to first search through the list and se whether a good candidate already exists before adding a new key word.

5.4.6 Search Screen

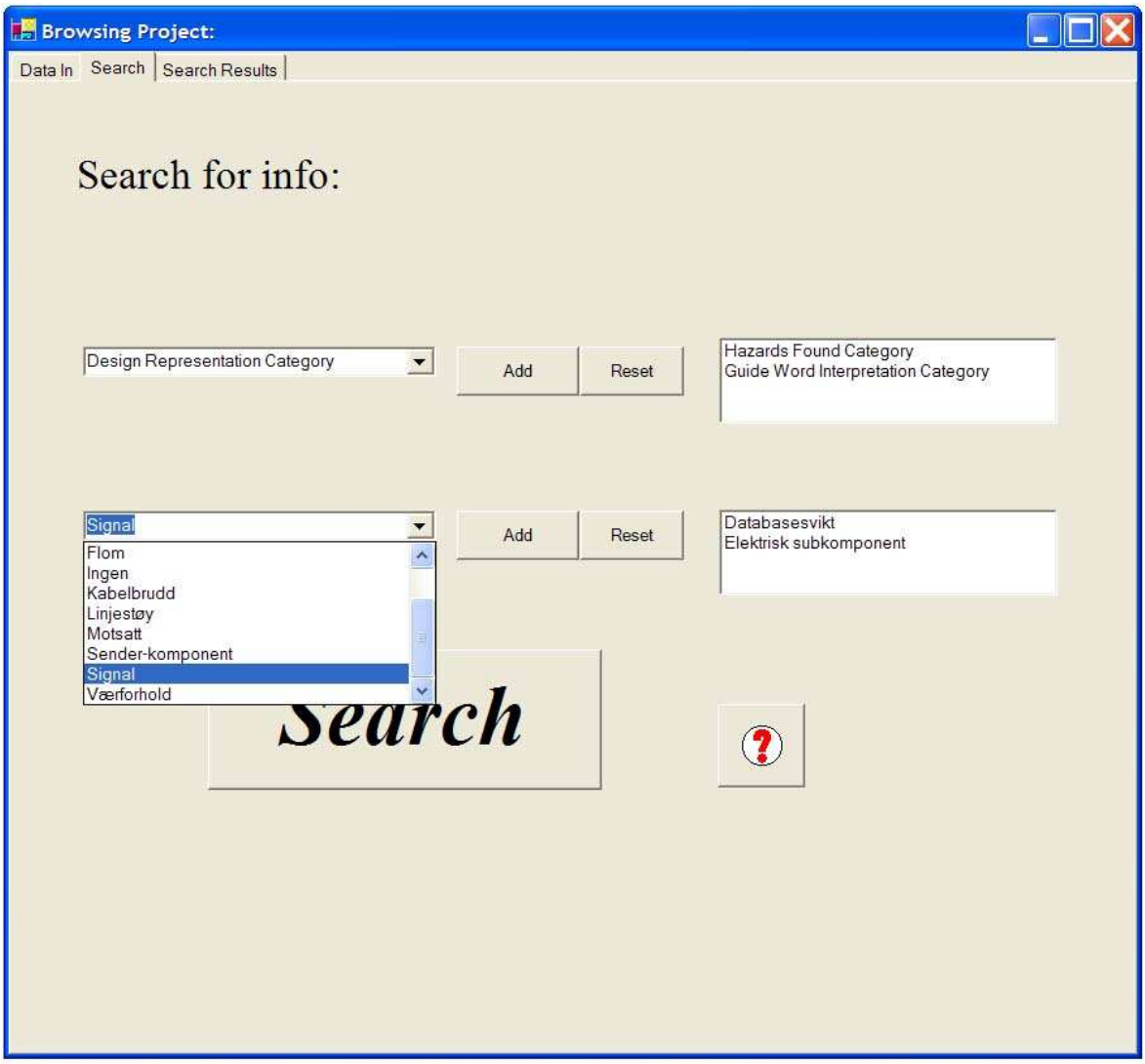

Figure 29: Screenshot from the search screen

When the user wants to search the database for data relevant to what he/she may be facing in a current HazOp, he/she can use the search functionality. By adding keywords to the fields as they are entered into the tool, one can ensure that information will be easy to track down at a later stage. Users may specify a search where the entire database is searched and all hits are returned in a table where their origin HazOp study can be seen and a small excerpt from the field in question is shown to help the user decide whether this is a relevant hit or not.

5.4.7 Search Result Screen

Figure 30: Screenshot from the search result screen

The user can select one of the fields by clicking on it, the project from which this field was taken will automatically pop up in a separate browser in view mode. In addition, the browser will be focused on the field selected by the user in the search, making it easy for the user to use the search functions. When the user is finished browsing the project that was presented in the search he or she simply closes the new browser and is returned to the search results where a different entry may be selected. If the user should want to change the search criteria he/she simply presses the ”search” tab and is returned to the search criteria section of the program.

5.5 Further Development

In order to get the tool to implement the requirements for generality as set out in section 4, a few things need to be done. Firstly and most importantly, the graphical module aka the project browser needs to be able to display a user interface based on the content of xml-files. This means that no graphical elements can be hard-coded into the program for the "Data In" part of the browser. Secondly, we need a controller module that is able to read XML files and use the information in the files to generate the graphical user interface from scratch, based on the content of those files. Thirdly, the category editor needs to be made. This editor needs to be able to generate XML files for the categories as specified by the users and update the database accordingly.

The data model can be kept more or less as is, the only difference being the constant lists replacing the guidewords. Hence, all the data we have collected in this experiment can be reused in a later experiment. We made sure that the data model was general enough from the start. This was done so that it would be possible to conduct more experiments in the future while re-implementing the tool part by part at later stages. Some graphical elements such as the welcome screen and the keyword management screen as well as the screens for searching for fields via keywords can also be kept as is.

Other changes that is advised for future development of the software tool is to remove the object model of projects and their fields and instead using the database directly to feed the graphical layer with information. Additionally, a better search functionality should be implemented based on searching directly through the database. This will lead to a significant simplification in the program when changing it to a dynamic self-changing tool.

6 Software Tool Implementation

This section of the document is mostly written for anyone wanting to continue our work on the software tool. It explains some of the choices we made when implementing, For a more detailed and thorough description of the implementation see E

Our main concerns for the software tool was that it should be both easy to use and fast to implement. We wanted our participants to be able to use the program with ease so that little error was introduced by people not understanding what to do. We also needed to make this program in a short period of time as we needed time to do the experiments and analyse the results.

C# seemed a suitable tool to make the program in, as it delivered a familiar user interface with a windows-feel to it, and proved quick to implement using Visual Studio .NET. Using C# resulted in a lot of code, but using Visual Studio, implementing this code was relatively easy.

The total number of lines of source code in the software tool ended up to be 5100. Some of this code was auto-generated by Visual Studio.NET. We estimate that a completely universal program would need some 10000 lines of code in order to work as specified in section 4. The program we used in the experiment had 5100 lines of code. Out of our code, some 1000-2000 lines should be directly reusable in such a program, the rest would only work as a starting point on which to base more general code.

We have included the source code for the software tool as well as a compiled version in Appendix E. Anyone wishing to extend our work is free to do so, and all code is offered as is. In Appendix E, a more complete overview of the code produced is offered.

7 Experiment

We base our experiment on the theory as presented by Claes Wohlin et al. in [22]. First we will present the theory, then we will follow their template in presenting how we conducted the experiment.

7.1 Experiment Theory

Experiments are a form of empirical quantitative research. The aim of an experiment is to identify a cause-effect relationship. An advantage of experiments, is that they provide quantitative data that promote comparison and statistical analysis. This makes an experiment useful in our case since we want to study the effect of using a software tool for reuse of experience in a HazOp.

According to Wohlin et al. [22], the phases typically involved in software experiments are:

- Experiment definition.
- Experiment planning.
- Experiment operation.
- Analysis and interpretation.
- Presentation and package.

In the definition phase, we state the purpose of the study, and define the goals of the study. A null hypothesis and alternative hypotheses are formulated in this phase.

In the planning phase, the context of the experiment is determined in detail. This includes setting up testing facilities and determining what is to be tested and how.

In the operations phase, the experiment environment is set up according to what has been determined in the planning phase. Actual experiments are run and data collected.

In the analysis and interpretation phase, descriptive statistics are used to understand the data informally. More detailed statistical analysis is done on data deemed to be relevant. The null hypothesis is either rejected or accepted.

In the presentation and package phase, the results are written down and made available to the public (this report).

All of these phases were used explicitly in our work.

7.2 Experiment Definition

In order to conduct a successful experiment it is necessary to have a good understanding of its purpose. Defining its goals of makes an experiment easier to plan, as this will avoid confusion of what it tries to achieve. Goal Question Metric (GQM) is a useful method for defining the goals of an experiment.

The GQM model is a standardized way of seeing software experiments through from start to end. The purpose of the GQM is to make sure that the focus is kept on fulfilling the goals of the experiment. Using the GQM method makes it easier to verify that important aspects of the experiment are defined before the planning and execution phase.

"GQM defines a goal, refines this goal into questions, and defines metrics that should provide the information to answer these questions. By answering the questions, the measured data defines the goals operationally, and can be analysed to identify whether or not the goals are attained. Thus GQM defines metrics from a top-down perspective and analyses and interprets the measurement data bottom-up." [21]

Using the Goal Question Metric Models goal template, the goal definition of the experiment is to:

Object:	The HazOp Tool
Purpose:	Find if use of the program leads to more hazards found in a HazOp studies.
Quality Focus:	Number of Hazards found, time needed for HazOp.
Perspective:	HazOp team member, Project management.
Context:	Student experiment with a toy size problem.

7.3 Experiment Plan

The purpose of the experiment plan is to describe how the experiment will be conducted. It is important that the plan correspond to the goal definition of the experiment. An experiment is built up around trying to check if the null hypothesis could be replaced with an alternative hypothesis.

The plan must include a context selection for the experiment. Next, hypotheses must be formulated and variables must be selected. Subjects for the experiment must be selected and together this forms a basis for the experiment design. The instrumentation, ie. what input to the experiment will be made and what outputs from the experiment will be measured, must be prepared. Finally, we need an evaluation of the validity aspects of the experiment.

7.3.1 Context Selection

Ideally, the experiment should reflect a real situation. In our case this would mean a real HazOp for an organization with a rich program information base. Unfortunately, this was not possible. The project problem was decided in January, at which time no tool had been implemented. As there was less than half a year to both create the tool and run an experiment on it, it proved impossible to make arrangements with an organisation for real life testing. A more realistic alternative was to run a HazOp experiment staffed with students. This, however, put limitations on how much time each treatment could take, as it was assumed to be difficult to get enough students to participate in treatments that lasted longer than 2 hours. This again set limitations on the size of the cases the students would analyse. Nevertheless, an experiment with students on a relatively small HazOp can still add value to the discussion of reusing experience in HazOp. Ours is not the first experiment to encounter such problems, using students in experiments is a common phenomenon. Even if using students leads to less external validity, many people, including Arne Sølvberg et al. believe that student experiments still can add value to research [20].

7.3.2 Hypotheses

As a starting point, our null hypothesis is that using a knowledge base tool in HazOp has no effect. We set this as our null hypothesis as we would like to show that it does have a positive effect on the HazOp. It is conceivable that using the tool could either improve the quality of a HazOp or reduce it. Since no two teams are the same, experience made by one team may be helpful to a different team and might help them find hazards they otherwise would have missed. On the other hand, using the tool takes up time. If it doesn't contribute enough to the process, it might not be worthwhile and result in time wasted, hence reducing the quality of HazOps. Therefore, our null hypothesis and alternative hypotheses are as follows:

- **Null Hypothesis:** Using the software tool has no effect on the number of hazards found in a HazOp.
- **Alternative Hypothesis A:** Using the software tool leads to more hazards found in a HazOp.
- **Alternative hypothesis B:** Using the software tool leads to less hazards found in a HazOp.

7.3.3 Variable Selection

Variables in an experiment are either independent or dependent. The independent variables are those variables we control and change in the experiment. The dependent variables are variables that might change as a result of applying the different independent variables. In this experiment there will be one independent variable; whether or not the students are using the software tool. The effect of the treatment of this variable will be the dependent variable; the number of hazards identified.

7.3.4 Selection of Subjects

HazOp is team work, where the team members must have enough knowledge to understand the design representation of the system being studied. Ideally, the study leader is experienced and many of the participants have taken part of a HazOp earlier. Unfortunately, there are not many students that fulfill this need. However, selecting students that had enough knowledge to understand the design representation of the flight case and train case was easy, since these design representations required only basic computer knowledge and an ability to read design documents. This taken into consideration, the selected subjects were chosen from the studies "Computer Science", "Cybernetics", "Electrical engineering" and "Communication Technologies" that had completed their first year of study.

We could not force anyone to take part in this experiment, we had to rely on convenience sampling within the limits just mentioned to get enough students to participate. We needed 42 students in order to carry out the experiments as planned. To get students to participate in the experiment, we sent e-mails to all the students doing the subject "TDT 4140 - Systemutvikling" at IDI, and also recruited students within the aforementioned limits from data labs on the campus of NTNU. We ended up with a semi-random sample from all of this by randomly selecting participants from the volunteers. In order to ensure enough interest in the experiment, each participant received a compensation of NOK 200.

7.3.5 Experiment Design

For testing the software tools effects on HazOps, we conducted a student experiment with two treatments (i and ii);

- (i) A HazOp without the use of the software Tool as a reference.
- (ii) A HazOp with the use of the software Tool to see how the tool affects the HazOp

The students involved in treatment (ii) used the tool when conducting the HazOp. Students in group (i) functioned as a baseline to measure the students in (ii) against.

The treatment (i) consisted of:

- A 15 minute introduction to HazOp using the robot cell case as an example
- A 90 minute period in which to do the HazOp on the provided case.
- A 10 minute period in which to fill in a questionnaire provided (see section 8.3).

The treatment (ii) consisted of:

- A 15 minute introduction to HazOp using the train position case as an example
- A 5 minute introduction to the software tool
- A 90 minute period in which to do the HazOp on the provided case.
- A 10 minute period in which to fill in a questionnaire provided (see section 8.3).

In treatment (ii) there was a 5 minute explanation of the HazOp tool to the students before the problem solving part so that they would be able to utilise it efficiently. The questionnaires provided both information to feed into the tool (after treatment (i)), and feedback on how they experienced the task of doing a HazOp. When doing the HazOp, the groups used a special form to record any hazards found. These forms also provided information to the HazOp tool.

When conducting the two treatments of the experiment, it was important to both get enough data that we might statistically show a difference between groups in (i) and (ii), and that we could be

confident that the data we collect is comparable. This was ensured through using only groups made up of a certain type of students for the experiment (see 7.3.4). Furthermore, any deviations from this optimal group composition was done symmetrically, so that the two treatments were as similar and comparable as possible.

7.3.6 Instrumentation

The instruments for an experiment can be divided into three types:

- objects
- guidelines
- measurement instruments

Objects The objects in this experiment was the cases the participants have to solve and a software tool with a knowledge base. The cases were taken from real world situations. There was a train position case and a flight landing case.

The software tool is useless unless it contains information that can be used during a HazOp. We needed to get data into the HazOp Tool before the experiment to test its usefulness. Since this would normally be done by persons used to the HazOp process, we opted to have more groups work on the same case to make the collected data more representative. We used four groups of three persons each in a pre-experiment. The resulting data was put into the system after being checked so that no two identical hazards were added. A group size of three persons seemed to be an optimal number in a student experiment to facilitate contribution from all members of each group. If too many students are placed in a group, it would be easy for some of them to "hide" in the mass, their ideas and experiences being lost to the users of the tool.

We conducted a pre-experiment with four groups for two reasons:

1. Gather information for the programs knowledge base
2. Gain experiment experience

It could not be taken for granted that the participants in the experiment could conduct a HazOp in a satisfying way when they had no experience with the methodology. In the pre-experiment we could also observe if any unforeseen problems would arise. The pre-experiment was conducted in this way:

- A 15 minute introduction to HazOp using a robot cell case as an example. See section C.1.
- A 60 minute period in which to do the HazOp on the provided train case. See section C.2.
- A 15 minute period in which to fill out a questionnaire provided. See section 8.3.

The data collection gave the following statistics:

Element	Hazards found
Group 1	18
Group 2	20
Group 3	23
Group 4	18
Mean:	19.75
Range:	5
Variance:	5.58

Table 2: Results from pre-experiment

The results show that the participants did not have much problem conducting the HazOp. This can be seen from the number of detected hazards, relatively low statistical range and standard deviation. If the results had been bad we would had to reconsider our introduction to HazOp. The purpose of our survey was to figure out if there were any problems that could not be seen from the results and observations. The survey pointed out that the time limit was a problem during the HazOp. Because of this, the HazOp in experiment treatments was expanded with 30 minutes from the pre-experiment.

An important issue to consider is the amount of experience stored in the knowledge base. Due to constraints on the experiment, we were only able to input experience from one HazOp done in the pre-experiment into the tool. In order to make sure that the participants would find relevant information in the knowledge base, we deliberately chose the two cases mentioned. They both portray systems for supervising position information of vehicles. For a thorough description of each system, see Appendix C. Specification and implementation of the software tool was also part of the instrumentation. Information on how we went ahead to produce the tool can be found in sections: 4, 5, 6.

Guidelines Since the subjects of the experiment did not have any experience with HazOp, we had to teach them and give them guidelines of how to work through the cases. For the students who were using the software tool we had to make guidelines for how to use the program when conducting the HazOp. More information about the HazOp guidelines we provided the participants can bee seen in B.

Instruments The measurement instrument used was the forms the groups used to record the hazards they found during the HazOp and a questionnaire in which they gave us feedback on the experiment. The hazard forms were of the type shown in Table 3.

Element	Attribute	guideword	Deviation	Consequence	Cause

Table 3: Form used by participants to record hazards

The questionnaire contained questions that both provided information to put into the knowledge base and feedback on how the participants experienced doing a HazOp.

7.3.7 Validity Evaluation

A discussion of validity threats of the experiment prior to the execution is needed to make sure that the experiment is planned in a proper way. The four kinds of validity in experiments according to Claes Wohlin et al in [22] are:

- Conclusion Validity: Ability to draw the correct conclusions.
- Internal Validity: Certainty that observed effect is caused by treatment.
- Construct Validity: Certainty that observed effect is part of assumed theory.
- External Validity: Ability to generalise results to industry practice.

For this experiment, the threats to conclusion validity were:

- **Low statistical power.**
  Not enough groups to get statistical significance. Since we had limited funding, we might risk not getting enough students to do the experiment as planned.
- **Random heterogeneity of subjects.**
  We might end up with having one treatment with very skilled students and a different treatment with very unskilled students. Since there was no clear cut way of testing this, and since using grades from university courses was not viable, due to the sensitivity of this information, we had no way of ensuring this does not happen.
- **Random irrelevancies in experimental setting.**
  Since the rooms we could book for the experiment was fairly small, the participants might be disturbed by noise from the other groups. There was also a possibility that some groups might hear and use what other groups discuss.

Threats to Internal Validity were:

- **Instrumentation.**
  The forms used may not catch the data we needed or the program might be unsuitable to use. Also, we had no guarantee that the categories we had chosen would be useful for future HazOps. The programs knowledge base might be too small to have significant impact in the experiment. In our experiment, we made the assumption that there was relevant information in the knowledge base. If the cases in the experiment were too different, the program wold probably have almost no effect on the HazOp.
- **Selection.**
  A certain type of student might be more likely to participate in the experiment, making it possible that observed effects stem from this biased selection instead of the treatment.

Threats to Construct Validity are:

- **Experimenter expectancies.**
  We might unconsciously give hints to the students that we expected the second treatment to be better than the first, thereby affecting them to put more effort into the second treatment than the first.
- **Hypothesis guessing.**
  When people take part in an experiment they might try to figure out what the purpose and intended results of the experiment is. This could influent their behavior in the experiment.

Threats to External Validity are:

- **Interaction of selection and treatment.**
  In practice, HazOp teams usually had a certain structure. Ideally, it has an experienced study leader, domain experts, and system designers. We could not compose such groups because of the resource constraints. This makes it difficult to generalise results to the real world.
- **Interaction of setting and treatment.**
  Due to time and economic constraints set on the experiment, the cases used for each treatment would necessarily be toy size or overly simplified real world problems. This would make it difficult to claim that any results could be generalised for problems of larger magnitude in the industry.

7.4 Experiment Operation

This section explains the preparations done before conducting the experiment and the measures that were taken to ensure a high degree of validity.

7.4.1 Preparation

During the experiment we experienced that getting enough participants was not as big a problem as we thought it to be. We were able to select 42 participants randomly from a pool of 75 interested students. We hope that this leads to a smaller bias on the experiment from the students involved.

We also found that our estimated 15 minute introduction to the HazOp process easily could be cut down to 10 minutes as the students took to the process quickly. The extra 5 minutes was given to them to do the HazOp, bringing the total time to analyse the system up to one hour and 35 minutes.

At the start of the experiment, each student was given a booklet containing a brief introduction to HazOp and an example case with an example HazOp form filled in to exemplify the HazOp process. The introduction they were given can be found in section B. In addition, they each received an explanation of the case they would work on, the flight landing case, and a design representation of that case. This information can be found in section C.3.

7.4.2 Execution

Physical location The groups were placed in relatively large rooms dimensioned for meetings with 20+ people. We had at most four groups at any one time in a room, each placed in a corner of the room. We would sit in the middle of the room ready to answer any question that might arise. The groups using the HazOp Tool were each given one computer at their desk to use as they pleased. Care was taken to make sure that the groups were spread out, so that they would not be within earshot of each other when talking softly. We had to do the experiment in four batches. This was done for practical reasons, both regards to the size of the room and with consideration of when the students were available for the experiment.

Subjects We ended up having to use two students from 5. year Computer Science on the first treatment of the experiment (HazOp without tool), these were spread on two different groups in that treatment. To balance the experience, we also used two 5. year students in the second treatment (with tool). When observing the groups, no difference could be observed in the groups that had one 5. year student from those groups that had none.

In order to avoid any Hawthorne Effect [12] in the experiment, we kept the real purpose of the experiment hidden from the participants until they were done with their HazOp. This was done so that the participants without the program were not to feel less obliged to do high quality work. Likewise that the participants using the tool should not feel a higher degree of stress to do good in the experiment than the other participants. To accomplish this we made care not to mention the HazOp tool to the participants in the first treatment, and we made care not to mention the first treatment to the participants of the second treatment. Wohlin et al. say that deception should only be used as a last resort:

"If deception is the only alternative, it should only be applied if it concerns aspects that are insignificant to the participants and do not affect their willingness to participate in the experiment. ... If deception is applied it should be explained and revealed to the participants as early as possible." [22]

We have lived up to this rule, we found no alternative to "fooling" the participants to believe the experiment tested them in a different way.

7.4.3 Data Validation

All the groups found between 18 and 30 hazards out of a total of 73 hazards found. This shows that none of the groups had too little understanding of the HazOp process to be able to work efficiently. In order to verify the hazards, we checked that the proposed hazards was indeed credible, and that the groups did not get credit for the same hazard more than once. During the experiment we were occasionally asked questions on the HazOp process by the groups, and we made a round after approximately 10 minutes to check that all the groups did indeed know what they were doing. During these rounds, we did not encounter a single case where the students were still with in the wilds regards to the HazOp process. Furthermore, all the groups were asked whether they experienced any problems during the HazOp and none answered that they thought the process was difficult to do or understand. During the experiment we observed that the students took their job seriously, and that they seemed genuinely concerned with getting the best results possible. All of this adds up to giving us a high confidence that the data we collected was produced by the best of the students' abilities.

8 Results

This section of the document shows the results from the experiment. First, an account of the observations made during the experiment will be made, Next, the hazards found by each group are given. Lastly, the feedback forms filled in by the groups will be discussed.

8.1 Statistical Data from Experiment

The groups in the experiment found a total of 73 different hazards. These are listed in the appendix section D. Here follows an overview of the statistical data found in the experiment.

8.1.1 #Total Hazards Found

In order to see whether the tool was beneficial to the process, we measured the total number of hazards found by each group and averaged those numbers to find the average number of hazards found both by groups using the tool, and groups without the tool. The results are shown in table 4.

	Average	Variance	Std. deviation	% of total
Without tool	22	18.5	4.3	30.1
With tool	26.6	15.8	3.97	36.4

Table 4: Statistical data from the experiment. Total amount of hazards found

The groups with the software tool found on average 36.4% of the hazards. The groups without found 30.1%. This means that, on average, the groups with the tool found 21% more hazards than their counterparts without the tool.

8.1.2 #Hazards Found Element by Element

In addition to comparing the total number of hazards found by each group we also wanted to see if the program could have an impact on detecting more hazards associated with certain elements. To do this, we measured how many hazards each group found on each element.

The spread of the hazards after interpreting them can be found in tables 6 and 5 shown below. The letters at the top represents modules in the GPS - landing system as identified by the groups. The letters are assigned to the modules as follows: (see section C.3 for a thorough explanation of the system.)

- a: The GPS satellites
- b: The different modules aboard the airplane
- c: The communication part of the ground system
- d: The RMM unit
- e: The remote control unit
- f: The ATC panel
- g: The central controlling system

Element	a	b	c	d	e	f	g	sum
Group A	4	4	3	0	8	0	0	19
Group B	3	7	4	1	2	1	0	18
Group C	0	5	3	4	6	2	0	20
Group D	6	7	5	4	0	3	0	25
Group H	3	7	5	3	4	6	0	28

Table 5: Results from treatment 1, without tool

Group E	8	2	8	4	8	0	0	30
Group F	8	10	0	0	8	0	3	29
Group G	3	7	6	1	3	0	0	20
Group I	5	3	7	0	10	0	3	28
Group J	4	3	9	2	0	8	0	26

Table 6: Results from treatment 2, with tool

These data are analysed in section 9 of this document.

8.2 Observations During Experiment

Some differences were observed in the way the groups worked on the problem. The largest difference could be seen in the groups where a clear leader surfaced. These groups had more effective discussions when finding hazards than the other groups. For the most part, this was achieved through keeping focus away from the possible causes and consequences of hazards, and instead focusing on identifying possible hazards. In the most efficient groups, the leader would cut right through a discussion and demand progress to the next hazard with authority enough to ensure efficiency.

On the other hand, in one group two leaders surfaced. This lead to some rather unwanted consequences. Firstly, the group would lose time bickering over the wording of certain hazards, neither of the leaders willing to give in. Secondly, neither part would admit "defeat" and ask questions about the design when issues were thought to be ambiguous. This naturally lead to frustration within the group and more time wasted. Lastly, the last member of the group almost did not contribute to the HazOp at all, his comments being ignored by the other two. In the end the group did not do as bad as could be expected, finding a total of 20 hazards. This is largely due to the fact that one of the "leaders" was acting as secretary, and would write down his own opinions while discussing with the other "leader".

The groups seemed to get into the HazOp method quickly, but some groups would sit and read the case for about 15 minutes before starting to work. This was more true for the groups that used the tool than for the others. We believe this was a direct result of entries in the tool which proclaimed the value of doing this. We also observed much fewer questions from groups that had taken the time to read the design representation properly from start to finish before starting to work on parts of it. The groups that had read through the design representation seemed more confident, and also gave more positive feedback afterwards.

When doing the HazOp, controversy within the groups as to the validity of a certain hazard would often arise. Some of the groups would spend much time deliberating over the hazard in question, effectively wasting their time while debating some esoteric point. The groups were never interrupted in their discussions so as to affect the experiment as little as possible. The futility of longwinded discussions was, however, pointed out in the start of the experiment during the introduction to the HazOp method. To be fair, all feedback was given directly to the groups asking questions, not to the room at large, so that the other groups present in the room at the time would not be unfairly favored over the other groups not in the room at that time. Each group had the same opportunity to ask questions during the experiment.

The groups working with the tool learned from the tool that previous groups had found it advantageous to start in one end of the "signal path" and work their way through to the other end. This seemed to work well, but it also lead some of the groups to not reach the end, being too detailed in their treatment of some of the elements in the design representation to have time to cover all of the design. The groups without the tool seemed "better" at budgeting their time to cover the entire design than the ones with the tool. On the other hand, the ones with the tool seemed to go deeper into each element. In effect, they seemed to go into more detail and to discover more hazards per element than the groups without the tool.

No treatment was noticeably better at finding non-trivial interpretations of guidewords. This is not surprising, however, since the data available in the tool provided few insights on how to interpret guide words. The two treatments thus had almost exactly the same starting point in this respect.

A difference we observed between those that used the software tool and those that didn't was that the latter was more sporadic in their use of guide words. Those that had the tool would go through the list of guide words methodically and write hazards for each guide word that made even remotely sense, even if that meant repeating hazards. Those without the tool would only write those that made sense once, and sometimes even forget about a guide word interpretation until after some time, going back to write a new hazard on an element they had "finished" some time ago. This made for a rather erratic progress with the groups constantly trying to see whether they had forgotten something

earlier, instead of focusing on the element at hand.

These observations all support that using inexperienced students has quite a few similarities to using personnel used to the HasOp method for experiments. The problems often encountered by HazOp teams are; time wasted while focusing on the problem and not on the solution, and the importance of an efficient and clear leader. These problems are often cited in HazOp literature as key issues that must be sorted out in order to ensure a successful HazOp [6] [15]. Having observed that the same tendencies in a way governed the successfulness of the groups in our experiment, this makes us more confident that the environment in the experiment at least to some degree matches the real-world environment of HazOp studies.

8.3 Questionnaire

When the groups had finished their HazOp or were told to stop, they filled in a questionnaire. This questionnaire had some standard feedback fields from which we could see how they felt they fared and what they thought of HazOp etc. Also, there were fields that were linked to the HazOp tool, giving us information both to put into the tool for later groups to use, and on how to improve the tool. Information put into the tool was for example information on what the students had learned about design representations during the HazOp etc. Later users of the tool will be able to use this information and add to it if need be. Here follows the questions, asked along with a summary of the answers.

What do you think of HazOp as a method for finding hazards? The groups without the tool answered almost unanimously that HazOp is both systematical and simple enough to be useful. One group pointed out that every hazard seems equally probable when doing the HazOp, thus one need a later study to identify probabilities.

Of the groups with the tool, three claimed that the method was a good one, emphasising the use of guide words to direct the thought process. Two groups thought the method was boring.

Do you think that HazOp is suited for finding hazards in practice? The groups without the tool answered that they thought HazOp was a good idea in principle, but that for the method to work in practice, knowledge of the domain at hand was paramount. They also pointed out that the HazOp method seemed to find all the "simple" hazards, but that it didn't investigate deeply into the systems.

One of the groups using the tool said that they didn't think HazOp dealt properly with hazards as a result of multiple components failing at the same time. Another claimed that they felt restricted by earlier observations. The rest thought the method was well suited in practice.

How was it to conduct a (limited) HazOp? some of the groups not using the tool said they felt they lacked the domain knowledge to do a good job. One group claimed that the study was boring, and one group said they felt they had identified all the hazards.

One group with the tool said they missed probability in the analysis, two group claimed they were unsure with what to fill in the hazard forms, and one group said they had been too detailed, thus not being able to cover the whole system.

Did you experience any problems in the experiment? Out of the groups not using the software tool, one group said they had a hard time identifying the elements in the design and one group said they needed more time to complete the study.

Three of the groups using the program felt they had too little time. One group said they felt they didn't know how to use the program properly. Incidentally, this was the group that scored lowest of the groups with the tool.

Did you use the tool actively during the HazOp Only one of the groups said they had not used the tool during the HazOp, this was group G, the group that scored lowest out of all the groups using the tool. The other groups had used it to some degree. One group claimed they had used it mostly at the beginning to get a flying start. One group said they used the tool to help brainstorming when they found no hazards on elements.

Was the tool helpful? One group (group G) claimed the tool just hindered the process consuming time without them getting anything back. The other groups claimed the tool had been helpful.

What functionality would you like the tool to have One group felt that the tool should be closer linked to the hazard forms. Two groups said they would want more search-functionality in the tool. The ability to specialise searches and group the results on the keywords.

Questionnaire summary Groups with the tool were generally more sure of their own domain knowledge when doing the HazOp. They also found the tool to be helpful during the study. The only negative aspect of using the tool was that they to a larger extent felt they had too little time. Seeing as they found more hazards than their counterparts, this could indicate that they were more thorough in their study. This, of course, is not negative at all. From observing the answers given by the groups the software tool seems to be a success. There does seem to be a potential for improvement in the tool. Most notably, the search functionality should be changed to reflect better the wishes of the participants.

9 Analysis

In the analysis part of an experiment, the results are discussed via descriptive statistics and hypothesis testing is done. The goal of the analysis is to be able to within a certain confidence reject the null hypothesis and accept an alternative hypothesis.

9.1 Measuring and Organizing Collected Data

After we had collected the schemas from the groups, we had to get a measurement of their work. As a part of the experiment plan, we wanted to compare hazards found using the program versus those found not using the program.

One problem with analyzing the schemas was that the same hazard can be described in different ways. Specifically, the granularity of the data offered by the groups was often different.

As an example there were some groups that interpreted the guide word "no" combined with the satellite's attribute signal as two hazards:

Element	Attribute	Guideword	Deviation	Consequences	Cause
Satellite	Signal	No	The satellite does not send any signal to the ground station	The ground station will not have information to find out how precise the GPS-signal is	Technical failure
Satellite	Signal	No	The satellite does not send any signal to the airplane	The airplane will lack necessary information needed for landing	Technical failure

Figure 31: Example of fine granularity in the data provided by the groups

Many of the other groups, however, interpreted this combination of element, attribute and guide word as one hazard:

Element	Attribute	Guideword	Deviation	Consequences	Cause
Satellite	Signal	No	The satellite does not send any signal	Lack of information to complete a safe landing	Technical failure

Figure 32: Example of coarse granularity in the data provided by the groups

In order to be able to compare the two different granularities, we interpreted the second example as expressing two distinct hazards in one row. Since they did not specify which signal was missing, no signal to both of the elements was implied.

This sort of organising the collected data was done throughout the experiment data, in effect bringing the data down to the lowest granularity level suggested by any group. We did this consistently throughout the results provided by the groups. This lead to the fact that groups tending to give more vague hazards were rewarded by having more hazards total. There was, however, very few instances of us having to modify the granularity of the data. As we had provided a very fine granularity in the example data, the groups tended to use this granularity as a template.

9.2 Descriptive Statistics

We have divided the participating groups into two categories. One that used the program and one that did not. Groups A through D and H did not use the software tool, groups E through G and I through J did use the tool.

As can be seen from the results in tables 6 and 5; the groups that used the software tool seemed to find more hazards than the ones without the tool. This was also noticed in the operation of the experiment as seen in section 8.

It was also noticed that the groups working with the software tool did not touch some of the elements. At the time of the experiment it was suggested that this came as a result from the groups not having enough time to complete the HazOp.

Group E, F, G and I did not find any hazards on the ATC panel. This component is a central part of the system and is to a large degree visible in the design representation. It is not likely that the groups merely "forgot" the component when doing the analysis. Far more likely is the possibility that they just did not make it because the limited time available. A total of 11 different hazards were identified on this component alone by other groups. We therefore made the assumption that it was not the absence of possible hazards that led to 0 hazards found for the four groups with this component. Three of the four groups in question were the groups that identified the most hazards in total, this leads us to believe that the oversight was not the result of sloppy work. Instead, the explanation is assumed to be that the groups did not have time to start analysing this component. This is supported by the observations made that suggested that the groups did not get to finish their analysis. See section 8.

With this assumption made, we had to look beyond merely counting the total number of hazards found, as this might not give a correct view of the effectiveness of the groups. It is conceivable that some groups started out with "easy" elements which had a lot of hazards connected to them, whereas some groups may have started out with elements without that many hazards. If the groups spent a comparable amount of time on each element, this would lead some groups to end up with more hazards found in total by skipping the "hard" elements. Thus, in order to see how effective a group was, a better measurement would be how thorough they were on each element they did get to investigate. Ie how many hazards they found on each element they got to start on.

None of the groups without the software tool found anything wrong with component g, the central controlling system. This might be because they did not think to include it into the analysis. This element could not be directly seen from the design representation. It had to be derived from the system explanation. In the program knowledge base there was information about a similar element abstraction for the train position system. The design representation category contained this advice and some of the entries in the detected hazards category could help the groups identifying this element. We believe that this is the reason why two of the groups using the program did analyse g and found 3 hazards each. The total hazards detected for this element was four.

When disregarding element g, the total count of un-analysed components are 4 out of 30 for the groups without the tool and 8 out of 30 for the groups with the tool. Since the groups who where using the program found more hazards, this indicates that these groups were more thorough when analyzing each element, and that they did not have enough time to complete the study.

9.3 Dataset Reduction

In addition to comparing the total number of hazards found by each set of groups, we also analysed the results element by element. During the experiment we observed that the groups using the software tool seemed less able to cover the entire design representation when doing the analysis. Thus, the groups could well be doing the HazOp more thoroughly, but still end up with approximately the same amount of identified hazards as the other groups. In order to test whether this was the case, the groups' analysis of each element was compared. When doing this comparison, the element called "g" was omitted as none of the groups not using the software tool found any hazards in this element. Furthermore, this element was found only implicitly in the design representation and case description. Additionally, the dataset was reduced so that only groups that had actually covered the elements were taken into consideration. Since we could not observe during the experiment which elements were and were not covered by each group, we instead interpreted zero hazards found on a given element to mean that the group did not have time to start work on that element. Thus, all groups that found zero hazards on an element were not counted when analysing the results element by element.

9.4 Statistical Analysis

In order to reject the null hypothesis, one must show statistically within a certain level of confidence that the alternative hypothesis can be assumed to be correct. One method to do this when comparing two treatments is the Student's t-test. For the Student's t-test to work, the observations need to be normally distributed. We have no way to confirm this with just 5 observations made in each category, but since we are confident that we have a good semi-random sample of experiment subjects, and that the variations in the number of hazards found for each group stem from randomness in the sample, we assume that the samples are normally distributed and we can hence use the Student's t-test.

The student's t-test takes into consideration the observed variance and tests if there is a significant difference in the observed mean values. The null hypothesis is that there is no significant difference between the two treatments. If a significant difference is found on the mean values, this means that we can reject the null hypothesis.

Our null hypothesis was that the students using the software tool and the ones without discovers approximately the same amount of hazards in a system during a HazOp. In order to reject this null hypothesis, we needed to show that the students using the software tool either found significantly more hazards (alternative hypothesis A) or that the students using the software found significantly less (alterative hypothesis B). From the data available, it was clear that the alternative hypothesis B could not be accepted, since the students using the software tool found more hazards than the one without. Whether they found significantly more hazards, however, can be tested using the Student's t-test.

The groups using the software tool found an average of 26.6 hazards in the GPS landing system. The ones without the tool found a total average of 22 hazards.

Using a one sided Student's t-test, we found that the null hypothesis could be discarded with a 6% risk of error. In other words, there was a 94% possibility that the observed increase in hazards found by the groups using the software tool was not due to random effects. We were thus able to discard the null hypothesis and replace it with the alternative hypothesis A: "Using the software tool leads to more hazards found in a HazOp"

As was observed throughout the experiment, it seemed that the groups using the program were generally more thorough when going through the elements in the design representation. As a result of this, they did not cover the entire design representation. In order to check whether the groups with the program generally found more hazards on each element, the following test was applied:

For each of the identified elements named "a" through "f", skipping "g" on account of its ambivalent nature, a Student's t-test was done on an element by element basis, where groups with no hazards found for a particular element was disregarded (see section 9.3). When doing this, we got the following results:

	Average	Variance	Std. deviation	#Groups
a	4	2	1.41	4
b	6	2	1.41	5
c	4	1	1	5
d	3	2	1.41	4
e	5	6.67	2.58	4
f	3	4.67	2.16	4

Table 7: Statistical data from the experiment. Groups without tool

	Average	Variance	Std. deviation	#Groups
a	5.6	5.3	2.3	5
b	5	11.5	3.39	5
c	7.5	1.67	1.29	4
d	2.3	2.33	1.53	3
e	7.25	8.92	2.99	4
f	8	0	0	1

Table 8: Statistical data from the experiment. Groups with tool

When feeding this data into the t-test, the results were as follows:

- Element a: The groups using the software tool was more thorough than the other groups with 88% confidence.
- Element b: The groups using the software tool was more thorough than the other groups with 71% confidence.
- Element c: The groups using the software tool was more thorough than the other groups with 92% confidence.
- Element d: The groups using the software tool was more thorough than the other groups with 70% confidence.
- Element e: The groups using the software tool was more thorough than the other groups with 85% confidence.
- Element f: This proved impossible to test due to there being only one observation from the groups using the program.

Setting a limit at 90% confidence, only element b was shown to be more thoroughly investigated by the groups using the software tool than the others. Thus we were not able to show that using a HazOp software tool for knowledge reuse leads to each element being more thoroughly investigated. We have, however, been able to show that the total amount of hazards found increases with the use of such a tool.

9.5 Discussion

The tool available to the students held only experience from one previous HazOp. The results showed that even if only one project is found in the programs knowledge base, the groups could use this experience to detect more hazards. In a real life setting, such a program might hold experience from tens, even hundreds, of HazOps. Thus, the usefulness of a HazOp tool for experience transfer would be much greater in real-life. It is probably the similarity of the two cases that compensates the limited program information. If the knowledge base was huge, it could have been more difficult to find relevant information which again could have effect on the groups' motivation on using the program.

One should be careful, however, to proclaim the software tool to be a necessity for real-life HazOp based on the results from this experiment. The participants in this experiment had absolutely no experience with the HazOp process and only a little experience in reading design documents. It could well be that the information they got from the HazOp tool only helped them get a better understanding of the HazOp process. If this is the case, people in the industry would not benefit from using the tool to the same extent as they are already familiar with the process.

During the experiment we observed that the groups did not find it to be a inconvenience having a computer in front of them during the HazOp. The process seemed well suited to accommodate the participants occasionally seeking information and inspiration from the tool. This, we think, could be transferrable to real life HazOps.

9.5.1 Knowledge Base Size

It is important to consider how much data is needed in a knowledge base before it can be considered to be useful. Clearly, a large knowledge base would be able to provide more clues to help the HazOp process than a small one. If there is insufficient data, the tool could become a burden more than a help to the process as explained in section 3.2. The question therefore arises how much information is needed as a minimum. If this amount could be identified, the data could be pre-loaded into the system before it was used by any company.

We found that experience from but one study was enough to help the students do a better HazOp. In the real world, this would probably not be the case. There is one factor that needs to be considered in order to make a claim about what the minimum value might be. Knowledge recorded by the HazOp teams would probably be limited to just the non-trivial experience they would gain. This to limit the knowledge base to a manageable size and to cut the costs of recording the experience.

Thus, a related question will be how much non-trivial experience is gained in a standard HazOp. If the HazOp teams are continually reinventing the wheel, having even one HazOp recorded would be a help. On the other hand, if most of the work is routine work, a different tool supporting automation of the HazOp process would be more suitable. The literature suggests, however, that the process works best when done as a creative process conducted by experienced persons [6]. This suggests that the practice of doing HazOps is nearer the former type.

Apart from basic knowledge on the HazOp method itself, all the experience gained in HazOp will be domain-specific to some extent. If doing a HazOp in a dissimilar domain, such knowledge could turn out to be next to irrelevant. In order to ensure that the tool is useful to an organisation no matter what the object of the HazOp might be, it is our opinion that there should at least be experience from one HazOp done on a system within each of the domains the organisation would need to do HazOps on in the future.

9.5.2 Validity

As was identified during the experiment plan, certain threats to the validity of the experiment exist. The complete list of threats we have found to apply for this experiment can be found in section 7.3.7

Conclusion Validity We had enough groups to reject the null hypothesis with 94% statistical significance. Thus, the threat of low statistical power was answered.

Almost all the participants were from the same year of university studies, and little difference in the skill-levels of the groups were observed. The deviations we made from the rule of just using second year students were done symmetrically. Thus, the threat of random heterogeneity seems to have been answered.

The rooms used for the experiment proved to be suitable for the experiments. The participants were able to speak softly to each other without being overheard by other groups. Our presence in the rooms made sure that the groups did not cooperate. This answers the identified threat of random irrelevancies

Internal Validity The tool proved to be useful during a HazOp, and the information we collected into the tool also proved useful. Furthermore, since we chose participants randomly from the pool of available students, and also chose randomly which treatment to put each student in, any bias from the participants should have been properly dealt with.

Construct Validity None of the groups were told beforehand the setup of the experiment, so they were not aware that we were to have two treatments. This effectively removed the threat of experimenter expectancy. The threat of hypothesis guessing was also countered in the same way.

External Validity We found it hard to counter the threats to external validity in this experiment.

Using only students from a limited set of possible students made for a very narrow spread in the experience of the students. Although this makes us able to claim that there is little threat to the conclusion validity, it means that we are unable to claim that our results could have been duplicated in a real-world HazOp study where this homogeneity would certainly not be found.

Using only toy-sized cases on which the students did HazOps, we could make sure that the groups would be able to cover most of the cases in the time allotted. But this also means that we cannot claim our results to be valid for-real world HazOps that might go on for months with larger groups of people working on much larger problems.

10 Conclusion

In this paper we have shown the possible benefits with using a tool for experience transfer in HazOps. We have laid out a requirements specification for such a tool, and implemented a prototype of that tool. Furthermore we have run an experiment to test whether using the tool is helpful during HazOps.

During the experiment we found that using a tool in which previous HazOp experience was stored lead to more hazards found in a HazOp done by students. Sadly, the lack of external validity in the experiment means that the results cannot easily be generalised to be true in the real world. We found that students with little or no experience with the HazOp methodology benefited significantly from having a tool available in which previous experience was stored. As such we have shown that there is some reason to believe that using experience transfer in HazOp might work in practice.

We were unable to show that using the HazOp tool led the groups to be more thorough on each individual element in the study, even though we did find some evidence to suggest this.

When using the tool, the groups felt less hindered by a lack of domain knowledge than their counterparts without the tool did. They also felt that they would need more time to complete the study. The only group that was negative towards the tool after the HazOp had not been able to use it properly, and was also the group with the least hazards found out of all the groups using the tool. (see section 8.3)

Observations made in section 8 support that the experiment we did have a lot in common with a real-world setting in that the groups faced the same problems often faced by HazOp teams in the industry.

All this makes us able to conclude that using the tool we have made can be beneficial to the HazOp process under certain conditions, especially when used by relatively inexperienced team members. In order to conclude on its overall usefulness, we would need to test it under several conditions. We explore this further in section 11.

11 Further work

Our experiment is only a beginning in quantitative research of experience-reuse in HazOp. Even though results from the experiment indicates that a well designed software tool can increase the performance of a HazOp, there are still many questions that need to be answered,

The rest of this section is organised as follows: First we will discuss what changes we see in hindsight we should have made to the experiment to make it better. Then we will explain how the tool should be improved to better facilitate cooperation with companies doing HazOp. Further, we will set out what we think to be ideal test conditions for the software tool. Lastly, we will discuss other topics that may be investigated in the wake of this report.

11.1 Better Experiment

The participants involved in the experiment had for the most part three complaints:

- They didn't have enough time to complete the study.
- The search functionality of the software tool could have been better.
- The amount of available data in the software tool was small.

Time: The time limit was imposed by us as we found it hard to believe that we would get enough participants if the experiment lasted much longer than two hours. As we didn't have any problems recruiting enough participants for this experiment, the next time a similar experiment is to be run, more time should be budgeted. Ideally, one should first get a few participants to do a HazOp where the goal is to see how long time they need to complete the study. Then one could limit the size of the system to be studied according to how much time it is possible to recruit the required number of participants with. From our experience, it should be possible to get students to participate in an experiment of at least three hours.

Having enough time to do the study is important. In a real-life HazOp, enough time is usually budgeted for the study. Lack of time leads to an inferior HazOp study which no party is interested in [6]. Thus, to better mimic a real life situation and thereby improve the external validity of the experiment, enough time should be afforded.

Tool: Some improvements on the software tool would also be preferable. The groups complained that they couldn't narrow down the search by putting in conditions like a logical "and" between keywords in the search. Also, they complained that the search data should be sorted in a better way. These shortcomings in the search functionality didn't seem to hinder the participants unduly, but it is likely that with a larger information base to search through, problems like information overload would soon arise if nothing is done to ease information retrieval.

In order to improve the search functionality of the tool, a few small changes are needed. Firstly, when users search for multiple keywords simultaneously, the entries with the most keywords applying to them should appear at the top, being the most relevant hits. It should also be possible to specify some of the keywords as obligatory in the search. Entries failing to have these keywords would not be taken into consideration when building the result set of the search. Finally, being able to specify logical operators like AND and OR would make the search much more sophisticated.

Data: In our experiment, the participants only had access to experience from one previous HazOp. In an industrial setting, the tool would contain information from several HazOps. Since we had a serious lack of data, we should perhaps have supplemented the data in the knowledge base with information found in the literature in addition to common sense that just wasn't put down in writing

by any of the groups in the pre-experiment. Adding more data into the tool might make it more beneficial to use, provided that the search functionality facilitates easy information retrieval.

11.2 Case Study

A next logical step is to bring the software tool out into the real world. A viable option is to present the tool to an organisation and see how it affects their effectiveness in HazOps. Such a case study would need to be compared to the company baseline in order to check effectiveness. Multiple case studies would be needed in order to see whether the tool is helpful on a regular basis.

The need to move beyond student experiments is great. In a student experiment, we must limit ourselves to toy-sized problems and subjects with little or no prior experience with the HazOp process. This makes for little external validity in the experiments. A case study does have its disadvantages though. Most noteworthy is the inability to generalise the results from one case study in one organisation to the industry at large. We are seeing this drawback already in a student experiment, however.

In spite of this drawback, a case study would provide valuable information on how the organisation used the tool, and what they would want such a tool to do. However, before an organisation will want to use the tool, it must be made configurable and should be filled with useful data. Without initial data, there is little chance of any return on the investment of using the tool the first time around.

The case study should therefore be done in two parts. First, one should collect data to put into the tool and find which categories are needed. This should be done in cooperation with companies that do HazOps on a regular basis. This work could also be done simultaneously with upgrading the tool to encompass the category editor by one or two Master of Technology, Computer Science 5. year students for their project work in the ninth semester. Second, the follow-up case study of the HazOp tool used in practice in one or more companies for real life HazOps could then be done as the Master Thesis in the tenth semester. A small HazOp would be required so that there would be time to complete it in the course of six months.

11.3 Improving the Software Tool

Before a case study can be performed with the software tool, it will need to be improved beyond the prototype used in this experiment. Most of the changes needed are mentioned in section 5.5 of this document. Firstly, the category editor and universal behavior of the tool is paramount. Without this functionality, the tool will be very cumbersome to update, and it will generally be more trouble than it's worth to modify the existing categories. This might lead to little goodwill among the companies towards putting the tool into production. As mentioned earlier, a better approach towards information retrieval is needed, as the search functionality currently embedded in the tool is too simple. Additionally, having a standard minimum knowledge base when distributing the tool might encourage smaller organisations to use the tool, as they might not be able to build a large knowledge base by themselves.

11.4 Other Topics

Apart from seeing whether or not the tool is helpful in a HazOp, more interesting topics could be investigated during a case study. One interesting point would be how the tool was used in practice, who would use it and when. An additional point is what sort of data is stored. Keeping in mind that the tool would be universal enough that the different organisations would be able to change it over time as needed, different strategies for knowledge storing and retrieval could possibly arise with different needs in each organisation. It would also be interesting to see whether the tool had any effect on the employees' views on HazOp and its value within the organisation.

Given that enough organisations participated, knowledge on how to do HazOps could be shared between them within the tool, making for better HazOps in each organisation, and a faster growing knowledge base. If this proved to be politically feasible within the industry, it could lead to safer systems produced at a lower cost. In our opinion this is a goal that further studies should try and work towards.

References

[1] Kristian Abrahamsen. A software tool for reuse of experience in hazop, 2004. url: www.idi.ntnu.no/grupper/su/fordypningsprosjekt-2003/fordypning2003-Kristian-Marheim-Abrahamsen.pdf, accessed 13.06.04. 1.3

[2] A. Bundy. *Drowning in information, starved for knowledge: information literacy, not technology, is the issue.* VALA Conference, Melbourne 2000, 2000. url: http://www.library.unisa.edu.au/about/papers/drowning.htm, accessed 13.06.04. 3.2

[3] Prepared by Battelle Columbus Division for The Center for Chemical Process Safety of the American Institute of Chemical Engineers. Guidelines for hazard evaluation procedures. url: http://pie.che.ufl.edu/guides/HazOp/, accessed 13.06.04. 1.2, 2.3

[4] B. Carter, T. Hancock, J. morin, and N. Robins. *Introducing RISKMAN methodology.* Blackwell Ltd., 1994. 1

[5] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language. 2. Ed.* Booch Jacobson Rumbaugh, 2003. ISBN: 0-321-19368-7. 4

[6] F.Redmill, M. Chodleigh, and J. Catmur. *System Safety: HAZOP and Software HAZOP.* WILEY, 1999. ISBN: 0-471-98280-6. (document), 1.1, 1.2, 2, 2.1, 2.2, 2.3, 1, 2.4, 3.1, 3.1, 8.2, 9.5.1, 11.1

[7] T. Hewett. Cognitive factors in design: Basic phenomena in human memory and problem solving. url: www.acm.org/sigchi/chi96/proceedings/tutorial/Hewett/tth_txt.htm, accessed 13.06.04. 3.1

[8] C. Knutson and S Carmichael. Safety first: Avoiding software mishaps. url: http://www.embedded.com/2000/0011/0011feat1.htm, accessed 13.06.04. 1

[9] Rick Kazman. Len Bass, Paul Clements. *Software Architecture in Practice, second ed.* Addison-Wesley, 2003. ISBN: 0-321-15495-9. 5.2

[10] D. Lewis. Psychologist d. lewis quoted on cnn webpage. url: http://www.cnn.com/TECH/9704/15/info.overload/index.html, accessed 13.06.04. 3.2

[11] J. McDermid, M. Micholson, D. Pumfrey, and P. Fenelon. *Experience with the application of HAZOP to computer-based systems.* British Aerospace Dependable Computing Systems Centre and High Integrity Systems Engineering Group of Computer Science, 2000. url: http://www-users.cs.york.ac.uk/ djp/publications/djp-compass95.pdf, accessed 13.06.04. 3.2

[12] Phillip Meyer. The new precision journalism. url: http://www.csudh.edu/dearhabermas/hawthbk02.htm, , accessed 13.06.04. 7.4.2

[13] M. Nelson. We have the information you want, but getting it will cost you: Being held hostage by information overload, 2001. url: http://info.acm.org/crossroads/xrds1-1/mnelson.html, accessed 13.06.04. 3.2

[14] IEEE press. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* IEEE press, 2000. 3.2

[15] Marvin Rausand. *Risikoanalyse, veiledning til NS 5814.* Tapir, 1991. 8.2

[16] F. Sazama. *An organizational approach for experience-based process improvement in software engineering: The Software Experience Center.* Software Quality Systems, 2001. 3.1

[17] I. Sommerville. *Software Engineering.* Addison - Wesly, 2001. 2.2

[18] T. Srivatanakul, John A. Clarck, and Fiona Polack. Writing effective security abuse cases, 2004. url: http://www-users.cs.york.ac.uk/ fiona/PUBS/UseCaseYR.pdf, accessed 13.06.04. 2.3

[19] Tor Staalhane, Torgeir Dingsøyr, Nils Brede Moe, and Geir Kjetil Hanssen. Post mortem - an assessment of two approaches, 2001. url: http://idi.ntnu.no/grupper/su/publ/doc/PMA_two_approaches-final.doc, accessed 13.06.04, accessed 13.06.04. 3.2

[20] Arne Sølvberg, Daniel L. Moody, Guttorm Sindre, and Terje Brasethvik. Evaluting the quality of information models: Empirical testing of a conceptual model quality framework, 2003. 7.3.1

[21] Rini van Solingen and Egon Berghout. *The Goal/Question/Metric Method, A Practical Guide for Quality Improvement of Software Development.* Mc Graw Hill, 1999. ISBN: 007 709553 7. 7.2

[22] Claes Wohlin, Per Runeson, Martin Host, Magnus C. Ohlsson, Bjorn Regnell, and Anders Wesslén. *Experimentation In Software Engineering. An Introduction.* Kluwer Academic Publishers, 2002. ISBN: 0-7923-8682-2. 7, 7.1, 7.3.7, 7.4.2

A Glossary

Attribute - An attribute is a relevant property of an entity.

Design representation - All documentation of the system design which is the HazOp team needs to complete the study.

Dialog box - A window in a graphical user interface which appears in order to request information from the user. The user confirms the information by clicking the "OK" button

Experiment - Empirical quantitative research aiming to identify cause-effect relationships

Guide word - A guide word is a word or phrase which expresses and defines a specific type of deviation from design intent.

GQM - Goal Question Metric. Method for defining the goals of an experiment.

Hazard - A hazard is a set of conditions, or a state, that could lead to an accident, given the right environmental trigger or set of events. An accident is the realization of the negative potential inherent in a hazard.

HazOp - Hazards and Operability study. A technique that helps an organization detecting hazards by having a group brainstorms through the systems design representation in a structured manner.

Information overload - The problem of exposing individuals of too much information

PMA - Post Mortem Analysis. A technique for eliciting experience in a project or after it has finished

SQL - Structured Query Language. Query language which is used to update, insert, select and delete data from a database.

Study initiator - The person in the organization who is responsible for initiating the HazOp and selecting the study leader. The study initiator has the overall responsibility for the HazOp project.

Study leader - The person who is responsible for leading the meeting and makes sure that the follow-up work is done.

System - A collection of components organized to accomplish a specific function or set of functions.

System stakeholder - An individual, team, or organization (or classes thereof) with interest in, or concerns relative to, a system

View - A representation of a whole system from the perspective of a related set of concerns.

Viewpoint - A specification of the convention for constructing and using a view.

B Introduction to HazOp

This part of the report gives the Information given to the students in the experiment. Since we were only dealing with norwegian students, the introduction they got into the HazOp method was only given in Norwegian. This introduction was a synopsis of the explanation in 2 and was as follows:

HazOp er en metode for å identifisere "farlige" elementer i systemer (hasarder). Selv om metoden er svært enkel er den meget populær i industrien, siden den tilbyr en grundig gjennomgang av systemet og en systematisk prosess for å identifisere hasarder. Man tar utgangspunkt i designen av systemet som man ser vha. en såkalt "Design Representasjon" (Design representation). Og går igjennom denne element for element. For hvert element benytter man alle ledeordene som passer til det elementet.

Disse er:

1. Ingen (No) - Benektelse av Formålet.
2. Mindre (Less) -Kvantitativ Minking
3. Mer (More) - Kvantitativ Økning
4. Del Av (Part Of) - Kvalitativ Minking
5. Motsatt (Opposite) - Motsatt av Formålet
6. Andre Enn (Other Than) - Fullstendig Substitusjon
7. Tidlig (Early)
8. Sen (Late)

for hvert av disse ordene tolker man hva betydningen blir i forbindelse med elementet og undersøker så hvorvidt dette medfører en hasard. Ofte kan man støte på elementer hvor et gitt ledeord kan tolkes på flere ulike måter. I disse tilfellene tar man rett og slett og undersøker alle tolkningene hver for seg. Som konklusjon har man enten "hasard" eller "ingen effekt". Hvor hasard dekker over både personskader/dødsfall og forsinkelse/stopp i produksjon osv.

Hensikten med HazOp er å *avdekke* hasarder, deres årsaker og konsekvenser, ikke å foreslå løsninger på problemene. For at tiden skal kunne benyttes effektivt er det viktig å hele tiden drive prosessen fremover og unngå lange diskusjoner utover identifisering av hasarder. Dersom man ikke klarer å avgjøre hvorvidt anvendelsen av ledeord på element kan utgjøre en hasard eller ikke, skal dette dokumenteres som "Mer Informasjon Trengs".

C Cases

The following three cases were used in the experiment. The robot case as an example to the students in the pre-experiment on how the HazOp method works, the train case as an example to the groups in both treatments of the experiment and lastly, the landing system case was studied and analysed by the participants in the experiment.

C.1 Robot Case

This case was given as an example only on how a HazOp might be done to further enlighten the students in the HazOp process. The case was only given to the participants of the pre-experiment. These students did a HazOp on the "Train Case" *see* traincasend recorded their experience into the database. In a robot cell environment there is a robot with an arm powerful enough to maim or kill

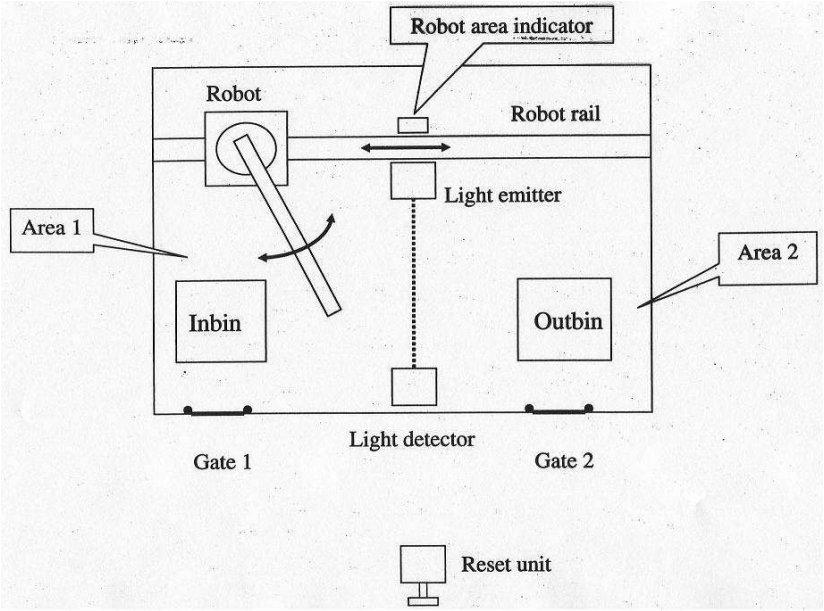

Figure 33: The Robot Cell System case used in the pre-experiment

a person. The robot works in a room with chicken wire walls (a cell) divided into two parts. a door leading into each. In the first there is an "inbox" where the robot picks up raw materials to work with. In the second there is an outbox where the robot puts finished artifacts into. Maintenance workers constantly need to refill the inbox and remove finished artifacts from the outbox. To do that, they go into the part of the cell where the robot is not, do the needed work and get out before the robot comes back to that part of the room. Additionally, sometimes the robot needs maintenance, the workers then need to get into the part of the room where the robot is.

To ensure the safety of the workers, the cell is equipped with some safety-features. If one were to open one of the doors, a circuit would be broken and the system registers the part of the cell where the door is located as "breached". If one were to cross into the other part of the room, a photoelectric switch would trigger, thus marking both rooms as breached. If the robot were to find itself in a breached part of the cell, it will automatically shut down. When passing from one part of the cell to the other, the robot triggers a flipswitch, thus making the system aware of its new position. In order to reset the system so that no part is marked as breached and restart a stopped robot, there is a button on the outside of the cell. This reset button cannot be reached from within the cell. If both doors are closed, a press on the reset button resets the system and restarts a stopped robot

C.2 Train Positioning Case

This case was given to four student groups. They were to do a HazOp on the system and so provide data for the groups using the software tool. The case was also given to the participants in the experiment as an example of how to do a HazOp analysis.

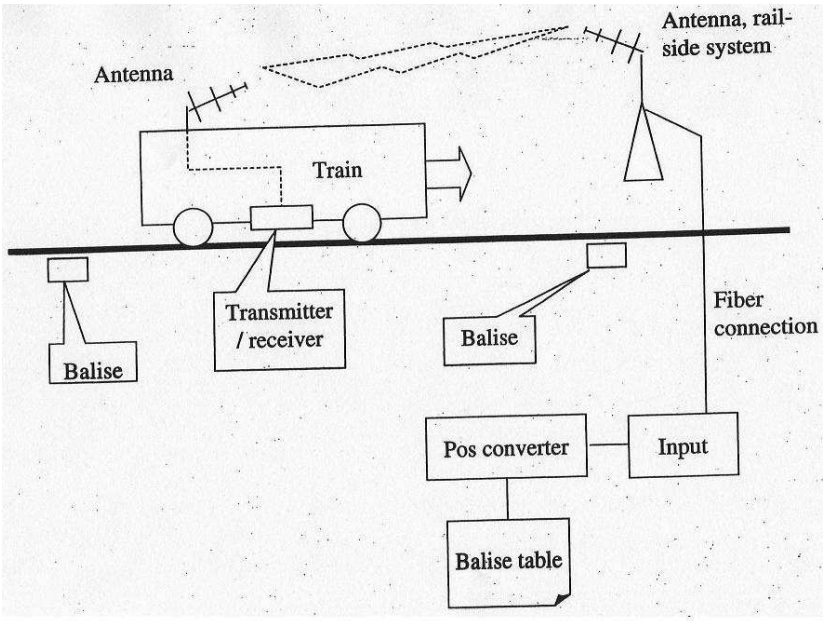

Figure 34: The Train Positioning System case used in the experiment

NSB (Norwegian State Railroad) is replacing the old safety system of line-telephones with a new one based on radio signals. In the old system, when an engine broke down or the line was blocked, the engineer needed to get out of the train and walk to the nearest mast where he would find a telephone. He would then call the traffic controller and tell him where he was. The controller would then hang up and call back. This would confirm the location. The controller would then direct other trains away from that part of the line using the light signals along the tracks to avoid accidents. This system requires two engineers in every train, one to guard the train and another to find a telephone.

In order to save money and at the same time be able to give better directions to the engineer NSB is replacing this system with one based on radio. This system consists of radio masts along the tracks so that a train always will be within reach of two of them. There's an antenna on the train which receives and sends signals to the radio masts. The radio masts are connected to the traffic controller room via fiberoptic cables. In addition to regular voice contact, the train sends position information. It happens like this: The train constantly sends a radio signal directed at the tracks. In the tracks there are "balises" that reflect the radio signal back to the train with a signature unique to each balise. This signature is then concatenated with the trains own unique signature and sent via the antenna, radio masts and fiberoptic cables to the traffic controller. The signals are received by an input unit, processed by a position-converter unit which uses a balise-table to keep track of where the balises are, and finally the traffic controller gets an indication on his display telling where the train is located. As balises from time to time get destroyed by weather or accidents, they occasionally need to be replaced. When a balise has been replaced, the old balise is removed from the balise-table and the new is added with the correct position stored. This is done manually. If two trains are headed into the same area, the traffic controller can give them a red light and use the radio to talk them through the area. In this way collisions are avoided

C.3 Landing Case

This case was given to the students participating in the experiment. They did a HazOp on this system which provided us with statistical data for our analysis. Most airports have a system to assist planes

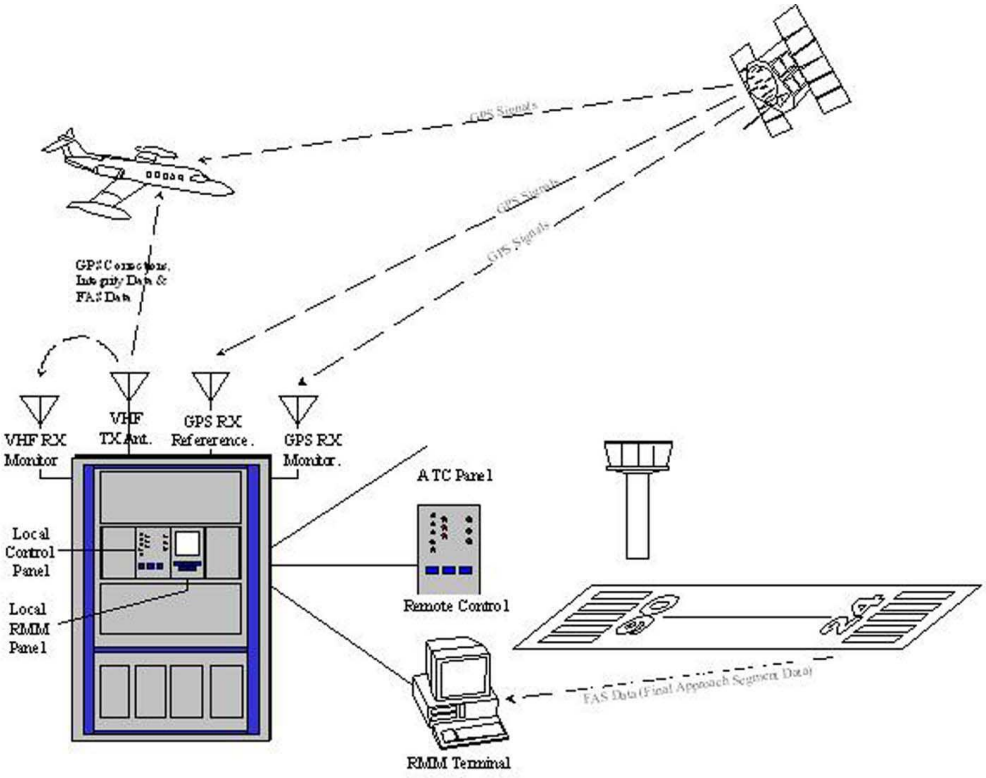

Figure 35: The GPS Landing system case used for the experiment

when landing. This system consists of a radio beam sent from the start of the runway and gives the optimal landing trajectory. Unfortunately, this beam only goes in a straight line. Some airports are situated in such a way that flying in a straight line when landing is not an option. Thus, these airports cannot use the current system, making it impossible to land there during fog. The new system is based on positions acquired through GPS satellites. GPS satellites sends signals to a GPS receiver on the plane which calculates the coordinates and height the plane currently has. This information is then fed into an onboard computer which, together with a 3d coordinate model calculates how far off the plane is from the optimal trajectory stored in the model.

As this system is meant to work under all conditions, including fog, it is of utmost importance that the coordinates are accurate. GPS positions are not accurate enough, however, and to counter this inaccuracy, there is a ground station with its own GPS receiver. The ground station knows its own exact position since it never moves. When the plane is close enough to initiate the landing system the error in the GPS signals for the plane will be almost identical to that of the ground station. The ground station can, however, knowing its own location exactly, send a correction signal to the plane. The GPS signals are adjusted by the correction signal before being fed into the on board computer.

Should any fault happen (such as if the signals from the satellite are of poor quality), an alarm will be sent to the plane, and the control panel for the system will display an alert.

The system has three modes of operation:

- Regular - the system runs and planes can land.

- Test - the system runs, but does not do self-checks.
- Service - the system may be reprogrammed.

Test Mode is used to calibrate the system and to check that everything works before using it for landing planes. The 3D model of the perfect landing trajectory is stored with the ground station and transmitted to the plane when it approaches. In order to change the model, the system needs to be in service mode. When the system is in service mode, service personnel may change the system through the Remote Monitoring and Maintenance System (RMMS). When in test mode or regular mode this system offers statistics and surveillance data.

In order to control the system there is a control panel attached to the ground station and a remote control located in the air traffic control tower. The ground control is located by the runway and is connected to the remote via wires. Both have the following switches and indicators:

- on/off switch
- alarm indicator
  - indicates whether an alarm situation has arisen.
- operation mode indicator
  - shows which mode of operation the system is currently in
- Test mode switch
  - switches the system in and out of test mode.
- Service mode switch
  - switches the system in and out of service mode.

These are used by airport employees when there is a need of a mode switch, and by air traffic controllers to verify which mode of operation the system is in and whether an alarm has arisen.

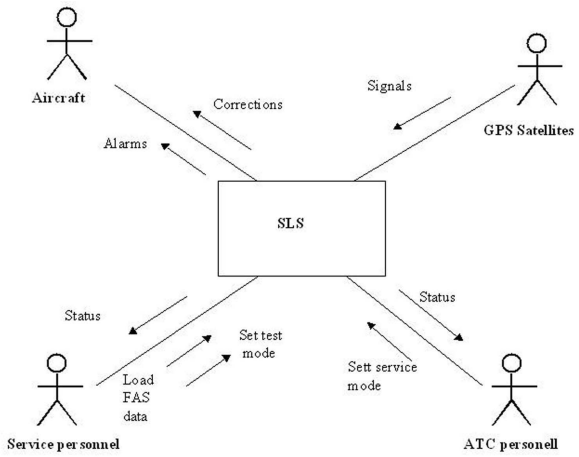

Figure 36: The various usages of the GPS Landing System

D List of all hazards found

Here follows the complete list of the hazards identified by all the groups.

D.1 Hazards found in GPS satelites

a1	No satelites available
a2	No signal comes through
a3	Fewer GPS satelites, erratic signal
a4	GPS signals very weak, erratic.
a5	too late GPS signals, wrong pos.
a6	GPS signal incomplete
a7	GPS signals too infrequent
a8	GPS signal is wrong.
a9	too late GPS signals, wrong pos.
a10	Signal wrong but passes CRC
a11	Signal contains too much information.
a12	Satelite in wrong position.

Table 9: The hazards the groups found in the GPS satelites.

D.2 Hazards found in The airplane modules

b1	On board computer dead
b2	Pilot misinterprets signals
b3	Pilot reacts too late
b4	On board computer missing
b5	Weak signals from Ground Station
b6	No signals from GPS
b7	No signals from Ground Station
b8	Onboard Antenna dead
b9	Signal misinterpreted
b10	Wrong signal from Ground Station
b11	GPS signal too late
b12	Too much noise on signals
b13	Signal from Ground station too late
b14	Weak signals from satelite

D.3 Hazards found in the Ground station communication module

c1	No GPS signals
c2	Weak GPS signals
c3	Wrong GPS signal recieved
c4	GPS signals recieved too late
c5	Too much noise on GPS signals
c6	Antenna gone
c7	Antenna dead
c8	No signals sent
c9	Less signals sent
c10	wrong signals sent
c11	late signal sent

D.4 Hazards found in the RMM unit

d1	RMM unit dead
d2	RMM unit cannot contact Ground Station
d3	RMM only recieves parrt of the signals
d4	RMM recieves wrong data
d5	RMM does not send data
d6	RMM sends data too late
d7	RMM corrupts the data

D.5 Hazards found in the remote control

e1	Remote dead
e2	Remote sends wrong signals
e3	Remote does not recieve signals
e4	Remote does not send signals
e5	Remote only sends part of signals
e6	Testmode switch dead
e7	Servicemode switch dead
e8	Alarm indicator does not indicate an alarm
e9	Alarm indicator indicates alarms too late
e10	Alarm indicator indicates false alarms
e11	Operation Indicator dead
e12	Operation Indicator indicates mode switches too late
e13	Operation Indicator indicates wrong mode
e14	Remote sends signals too late
e15	Remote does not indicate all alarms

D.6 Hazards found in the ATC Panel

f1	ATC Panel dead
f2	Wrong info shown on ATC Panel
f3	Human error in using ATC Panel
f4	No one present to operate ATC Panel
f5	ATC Panel operated too late
f6	Alarm Indicator indicates alarms too late
f7	Alarm Indicator indicates false alarms
f8	Alarm Indicator does not indicate an alarm
f9	Operation Indicator dead
f10	Operation Indicator indicates wrong mode

D.7 Hazards found in the central Landing System control system

g1	System failure, system down
g2	System calculates the wrong corrections
g3	System calculates corrections too late
g4	System does not calculate all the corrections

E Software Tool

As has been mentioned in section 5, we made some choices during implementation. We decided that we could not risk implementing the complete set of requirements as set out in section 4, but that we instead implemented a specific subset of these that made the tool usable in the experiment setting. We made the tool use central database access instead of local file storage to facilitate multiple users simultaneously accessing the same knowledge base. We did not, however, build in any checks to make sure that simultaneous editing would work as this would not be needed in the experiment.

The software tool consists of three parts, the graphical user interface and central logic module, the database access module, and the object model of the data.

The GUI and central logic module consists of the following classes:

Control.cs This class contains the main logic of the program, its main job is to switch between the different screens in the program and to collect data from the database and search through the data. It holds one active ProjectBrowser and can generate more as the user searches through the knowledge base and wants to see other projects. This is the class that contains the main method which starts off the program.

Intro.cs This class contains the graphics and logics belonging to the introduction screen. (see section 5.4.1 It is linked to the Control.

ProjectBrowser.cs This class contains the graphics and logics belonging to the project-browsing functionality of the program. In practice, this covers the uses as shown in sections 5.4.2, 5.4.3 and 5.4.4. The ProjectBrowser is loaded with all the data pertaining to one "project", ie a HazOp already done, when it is started. The ProjectBrowser is linked to the Control unit.

KeyAdder.cs This class contains the graphics belonging to the key adder screen as shown in section 5.4.5.

The Database Access Module consists of the following classes

IDBAccess.cs This is an interface used by Control in order to get data from the database.

HazOpDB.cs This class implements the IDBAccess interface. It delivers data objects to the Control unit on the basis of data in the database and makes updates in the database based on data objects given from the Control unit.

The Data Object Model consists of the following classes

Project.cs All data in the database is in some way connected to a project. In the object model, the project is the object that is loaded into project browsers when users want to see or edit data from previous HazOp studies. Each project contains one of each of CatHazardFound, CatGuideWord and CatDesignRep.

ProjectCollection.cs This is a collection of projects. It is the data format delivered to and from the database module

CatHazardFound.cs This is one of the categories implemented. It may contain one or more HazardFound objects.

CatGuideWord.cs This is one of the categories implemented. It may contain one or more GuideWord objects.

CatDesignRep.cs This is one of the categories implemented.

HazardFound.cs This object contains data belonging to the CatHazardFound object.

GuideWord.cs This object contains data belonging to the CatGuideWord object

SearchRequest.cs This object is part of the search-functionality of the program. The search request is tossed around the object model and generates SearchResults whenever appropriate

SearchResult.cs This represents a single hit between the keywords searched for and data.

SearchResultCollection.cs This represents the complete set of hits which are returned to the Control unit and used when the users are presented the search results in the tool as shown in section 5.4.7.

the sql file provided is enough to reconstruct the database the way it was when we did the experiment. In order to use it, a database must first be made using for example mysql. Then this file must be imported into the database.

In the physical copy of this appendix, here follows a cd containing the software tool with source code and sql script.