Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Offshore Technology – Industrial Asset Management	Spring semester, 2009 Open
Author: Christian Balestrand	…… (signature author)

Instructor: Tore Markeset

Supervisor(s): Maneesh Singh, DNV

Title of Master's Thesis: Usability Study of a Risk Based Inspection Software

ECTS: 30

Subject headings: Risk based inspection Usability Software engineering	Pages: …… (44) + attachments/other: …… (5) Stavanger, …….. Date/year

Abstract

Usability, or the "ease of use" of a software application, is important in order to allow for a user to carry out the intended task in an efficient and effective manner. For a technical software, as the risk based inspection (RBI) software studied in this thesis, such aspects is equally important. But in order to ensure a software which is "easy to use" there must be carried out usability activities in relation to the software's development.

The goal of the thesis was to carry out a usability study on DNVs risk based inspection software in order to identify what had a positive and negative impact on the software's usability. By indentifying it strengths and weaknesses it should be possible to present recommendations in order to improve usability in future RBI software applications.

In the thesis there was proposed a framework for carrying out the usability study as such guidance were not found in the literature. The proposed framework was developed specifically for the project and was based upon recognized usability evaluation and design methods. In addition the most beneficial usability evaluation method for the project had to be identified. The method of choice was Nielsen's ten heuristics; a set of broad based rules which if followed should ensure a software application with a high degree of usability.

Through the use of the software application to carry out an RBI analysis, each of the heuristics was applied to the software user interface, and thereby making it possible to indentify its usability strengths and weaknesses. The usability evaluation revealed several aspects which had an impact on the overall usability of the software. Especially the software's RBI working process and its somewhat limited error handling was found to have the most significant impact on the software's usability.

In addition to the usability evaluation there was carried out a literature survey to identify guidelines and design rules which could be relevant for future RBI software developments. From the usability evaluation and literature survey there were presented more than 30 suggestions and recommendations that should be considered when developing RBI software applications in the future.

Through the application of the proposed framework usability problems were identified and subsequent suggestions and recommendations for future software were presented. Based upon the latter accomplishments the usability study of Orbit Offshore was found to be a success.

Acknowledgements

This thesis was submitted as a part of the requirements for completing the Master degree program at the University of Stavanger (UiS), Norway. The Master thesis was carried out at DNV in the period January until June 2009.

I would like to express my sincere gratitude to my supervisor at DNV, Maneesh Singh, for his guidance and support throughout this project. I'm also grateful for the feedback and guidance related to the thesis work provided by my supervisor at the University of Stavanger, Associate Professor Tore Markeset.

Table of contents

1. Introduction

1.1 Background

Risk based inspection (RBI) is a methodology used for optimizing inspection efforts on equipment used in relation to processing of oil and gas, either onshore or offshore. The objective of RBI is to allow for prioritizing of inspection efforts and resources on equipment which have a high contribution to the plant risk, while minimizing the resources spent on equipment which has low plant risk. To be able to plan such inspection efforts an RBI analysis will have to be carried out. The RBI analysis is carried out by the use of a software application specialized for the specific task.

Orbit Offshore is a software application used by DNV Energy to carry out RBI analysis. The current software application was first developed more than ten years ago, but was significantly upgraded in early 2002. After this the current version has through the years only gone through minor changes in order to conform to the user requirements and to allow for an efficient working process for carrying out RBI analysis. The software's graphical user interface has because of the latter not been updated in the recent years. Because of this the software user interface is no longer up to date and is not very intuitive for the users to work with. The threshold for using the program has become so high that it is a problem for new DNV personnel to work with software application, the latter being a problem for DNV as the software is important for their business.

1.2 Aim of the Project

The aim of the project was to carry out a critical analysis of the current version of Orbit offshore and give suggestions for future developments.

1.3 Scope of work

The scope of work was to carry out a literature survey to identify the most applicable approach for the usability study of Orbit Offshore. From the usability study it should be possible to identify the software applications' strengths and weaknesses, and based upon those findings propose recommendations for future RBI software developments.

1.4 Project activities

I relation to the project a literature survey was carried out in order for the author to familiarize himself with the usability and RBI concept to allow for the selection of the most beneficial usability approach for this specific software application. From the Literature survey there was developed a framework for carrying out the usability study of Orbit Offshore.

In relation to the usability study Orbit Offshore has been used extensively. The software was used to redo a previously completed analysis in order to get a first-hand experience on how it is to work with the software. There was also carried out discussions with experienced user in order to obtain information about the use of the software and their opinion about its performance.

As a part of the developed framework and the usability study of Orbit Offshore there were also carried out a literature survey to possibly identify design rules and recommendations for future RBI software developments.

1.5 What have been done

The usability study of Orbit Offshore was focused upon the software's usability in relation its technical domain and more specifically on aspects that had a significant impact on the way the RBI analysis are carried out. Therefore some of the general software usability aspects which may otherwise be included in a traditional usability have been characterized to be negligible. The reason for this is that the author has not a formal education as usability or software engineer, and therefore such general aspects of usability would be difficult to identify.

The usability study will focus on the usability of the software's user interface, and not its application of the RBI methodology. The thesis will therefore not look into how the software applies the RBI methodology or evaluate the results correctness, but rather focus on the use of the software to carry out the RBI analysis.

1.6 Report Structure

In the following theory chapter there is given an introduction to the methodology of risk based inspection and that of usability evaluation. Also a short introduction to Orbit Offshore is given. The goal of the chapter is to give the reader a general understanding of the software and the concepts met upon later in the thesis.

In the third chapter the approach adopted to carry out the usability study and the related activities are explained. The choice of usability evaluation method and discussion of its applicability are also presented.

The fourth chapter is where the application of the approach and the subsequent results are presented. Here the results for the usability evaluation will be presented and discussed, together with the recommendations for future developments.

In the fifth and last chapter the framework adopted to carry out the usability study will be discussed, and a conclusion on the project accomplishments will be presented.

2. Theory

2.1 Introduction to Orbit Offshore

As mentioned in the thesis introduction Orbit Offshore is a computer software developed by DNV for carrying out RBI analysis. The software is used for carrying out risk calculations and can based upon the results output a detailed inspection plan. The software application is based upon DNVs methodology for RBI analysis which is presented in the DNV RP-G101 Recommended practice (2009). Because of the latter, the introduction to RBI presented in the next section is largely based upon the RP-G101.

While working with Orbit Offshore there was developed a flow chart of the applications working process, this can be found in Appendix 1. The flowchart might work as an aid for the reader when reading the results from the usability evaluation, which is presented later in the thesis.

In order to carry out the usability study of the Orbit Offshore, the author had to familiarize himself with the two concepts relevant for the project, the RBI and usability concepts. In the following paragraphs an introduction to the two concepts will be given.

2.2 Introduction to RBI

The following sub-chapter will give an introduction to the concept and the working process of the RBI methodology. The introduction to RBI is based upon DNVs Recommended Practice RP-G101 (2009) and API 580 Recommended Practice (2002). The RBI working process will be given special attention as Orbit Offshore is based upon the proposed working process mentioned in the DNV RP-G101.

2.2.1 What is Risk Based Inspection?

Risk based inspection (RBI) can be described as a risk management process. Risk management can be explained as an application of a structured approach in order to assess, mitigate and monitor risks. RBI is a specific approach designed to aid the development of optimized inspection and testing plans for production systems for oil and gas. The application of RBI allow for focus of the inspection activity on treats to plant integrity and its capability to generate revenue trough production. RP-G101 defines risk based inspection to be:

"A decision making technique for inspection planning based on risk – comprising the probability of failure and consequence of failure. "

The decision making technique mentioned refers to the ability to prioritize inspection activities on the items which has a significant contribution to the plant risk, while minimizing the resources used in relation to lower risk items. The overall goal of RBI is to optimize the inspection cost while maintaining the plant risk within acceptable risk levels.

The scope of RBI is all of the plants pressure systems, both for systems containing hydrocarbons and support/utility systems. RBI assessment is carried out for system components like piping and vessels, including heat exchangers, tanks, pressure vessels, and filters. The results from the RBI assessment are used as input to the creation of an inspection program. In Figure 2.2.1 the deliverables from the RBI assessment is presented.

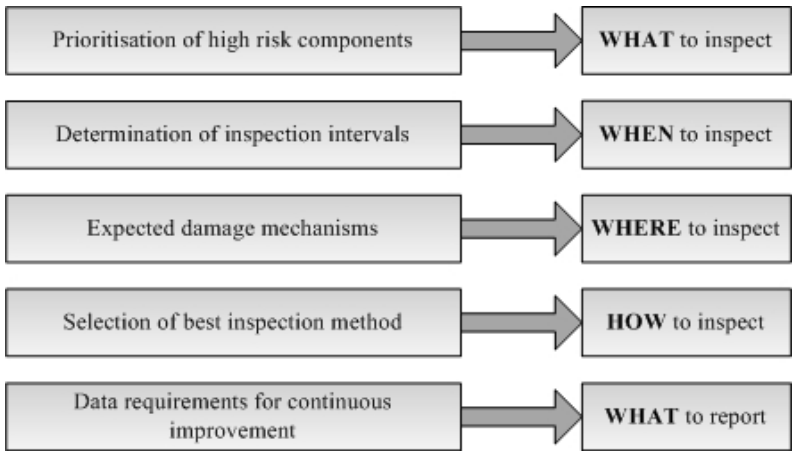


Figure 2.2.1. Deliverables of a RBI assessment to the inspection program (Source DNV RP-G101, 2009)

Inspection activities are carried out to check or establish if degradation is occurring, measure the progress of that degradation, and help maintaining the integrity of the system. In general the inspection activities can be said to be a tool for controlling and minimizing risks.

Risk is expressed as the combination of the components failure probability and its consequences. A single component may have several associated risk levels depending on the different consequences of failure and the different possibilities of those failures to occur. In the points below the three terms probability of failure, consequence of failure and risk are briefly explained.

- *Probability of failure* (PoF) is defined as the probability of an event occurring per unit time. Probability can either be given qualitatively as a ranking, low, medium, high, or it can be given quantitatively as the likelihood of the event occurring per year e.g $4.2x10^{-3}$.

- *Consequence of failure* is the expected outcome of a failure if it is to occur. In RBI there are defined three categories of consequence; safety, environment and economic. Safety consequences address personnel injury or death, environmental consequence address environmental damage, while economic consequences address financial loss. Consequence can be given qualitatively as a ranking, rated as minor, major or catastrophic. Or it can be given quantitatively, as for example for safety it will be potential loss of life, while for economic consequence it will be losses in monetary value.

- Risk is the combination of probability of an event occurring and the consequence of the event if it is to occur. Risk can be calculated by the use of the following calculation:

  Risk = Probability of an event x Consequence of that event

When carrying out the RBI assessment the probability of failure (POF) and consequence of failure (COF) will be assessed separately for each respective system or item, before they are combined to determine the risk.

2.2.2 RBI Working Process

DNV Recommended Practice RP-G101 proposes a recommended working process for carrying out the RBI assessment. The working process is divided into the following 5 stages:

1. Information gathering
2. Screening assessment
3. Detailed assessment
4. Planning
5. Execution and evaluation

The five stages of RBI assessment will be explained further in the following sections. A graphical overview of the RBI working process is available in Appendix 2.

1. Information gathering

In the first stage of the RBI working process information relevant for the RBI assessment are gathered. The amount of information needed to carry out the assessment depends upon if it is to be carried out quantitatively, qualitatively or semi-quantitatively. In general the screening assessment is carried out qualitatively while the detailed assessment is carried out quantitatively.

In order to carry out an RBI analysis there is needed large amount of data, some examples of such data are:

- Equipment type
- Material of construction
- Operating conditions
- Deterioration mechanisms
- Process fluid compositions

In order to obtain such information the RP-G101 mentions quite a few sources of information input which could be used, some typical sources are:

- Line list
- Drawings (P&IDs etc)
- Equipment list
- Material design specification and selection report
- Production data
- Operation and maintenance personnel

Line list which is mentioned above is an especially important source of information when utilizing a RBI software application. Line list is a digital computer file containing all components in a process system. The amount of data supplied in the line lists varies but often it contain material of construction, references to drawings, type of service, dimensions, operating conditions to mention a few. The line list is an important input when carrying out the RBI analysis by the use of a software application. The use of line list in relation to RBI analysis is will be discussed further later in the thesis.

2. Screening assessment

The purpose of the screening assessment is to determine on a higher level what items/systems that have a significant contribution to the plant risk. In most plants a large percentage of the total risk will be associated with a small amount of the equipment items. These potential high-

risk items should therefore receive greater attention in the risk assessment. Because of the latter, screening is carried out to identify these higher risk items which should be assessed in more detail in the later detailed assessment stage. When carrying out a screening assessment for an installation it is typically done so on in a qualitative manner and on a system by system, group by group, or major equipment item basis.

The consequence of failure and probability of failure are both assessed separately. Probability of failure is assessed by considering the operating conditions such as approximate chemical composition, temperature, and affects over time. These are considered in order to assess if there is a possibility of failure for the given system or item. The goal is to assess if the conditions are likely to give rise to negligible degradation ("low") or not negligible degradation rates ("high"). Consequence is assessed for the systems, and is often based upon the worst case scenario. The outcome of the worst case scenario is then rated to have a "low" or "high" consequence.

When the systems or items have been assigned with a PoF and CoF, the risk is assess and the specific item or system are assigned as having a "low", "medium" or "high" risk. A risk matrix for carrying out such a screening assessment is shown in Figure 2.2.2.

<table>
<tr><td colspan="3">Probability of Failure</td><td colspan="5">Risk Categories and Screening Actions</td></tr>
<tr><td>5
4
3
2</td><td>$>10^{-5}$</td><td>Significant probability of failure</td><td colspan="2">MEDIUM RISK
Inspection can be used to reduce the risk, but is unlikely to be cost-effective; the cheapest solution is often to carry out corrective maintenance upon failure.</td><td colspan="3">HIGH RISK
Detailed analysis of both consequence and probability of failure.</td></tr>
<tr><td>1</td><td>$>10^{-5}$</td><td>Negligible probability of failure</td><td colspan="2">LOW RISK
Minimum surveillance, with corrective maintenance, if any. Check that assumptions used in the damage assessment remain valid, e.g. due to changes in operating conditions.</td><td colspan="3">MEDIUM RISK
Consequence is high so actions (such as preventative maintenance) should be considered to ensure continued low probability as small changes in conditions can increase PoF and give high risk.</td></tr>
<tr><td colspan="3" rowspan="2">Consequence of Failure</td><td colspan="2">Acceptable consequence of failure</td><td colspan="3">Unacceptable consequence of failure</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td></tr>
</table>

Figure 2.2.2. Risk matrix for screening (Source: DNV RP-G101, 2009)

The combination of "high" PoF and "high" CoF necessitates a detailed RBI analysis /further analysis in the detailed assessment stage. If either PoF or CoF gets the score/result "low" it is recommended for maintenance activity. While if both get the score "low" the recommendations are that no further action is needed.

If a system or item is assigned with having "medium" or "high" risk it should be evaluated further in the subsequent detailed analysis.

3. Detailed assessment

The items or systems which were identified to have high or medium risk in the RBI screening stage will be examined further in the detailed assessment stage. The systems will be broken down to lower levels and assessed in more detailed. System and items are either evaluated

qualitative, quantitative or a semi-quantitative, but generally the detailed assessment is carried out quantitatively.

In the detailed assessment the goal is:

- Identification of relevant degradation mechanisms
- Estimate the extent of damage
- Estimate inspections timings
- Identification of inspection technique that ensure acceptable risk limit

In order to calculate the probability of failure, data for the different parts and systems are needed. Example of such data is materials, dimensions, pressure, temperature and service.
The internal and external degradation mechanism relevant for the circuits and parts are identified based upon their construction material and operating conditions. Subsequently the consequences of failure related to safety, financial and environmental are calculated.

Once the PoF and CoF have been calculated separately the product of the two can be calculated to obtain the part risk. When risk is calculated the time to risk limit can be estimated in relation to a previous defined risk acceptance limit. This makes it possible to estimate time to inspect and the frequency.

From the identified degradation mechanism it can be decided on the most applicable inspection method. The location or items which are to be inspected will also be determined in the detailed assessment stage.

4. Planning

The results from the screening and detailed assessment are used as input to the inspection planning stage. While the previous stages are carried out by RBI analysts the planning are done by inspection planners. They will use the previous mentioned results in addition to other relevant factors as for example available resources and logistics to create a preliminary inspection plan. The inspections are either listed per part or circuits. From the indentified item degradation mechanism hot spots can be identified, which are the specific locality on a part where the degradation is likely to occur or will be most significant. The identified hot spots are where on the item the inspection should be carried out. The identified inspection points are then marked-up system drawings.

There is also estimated how many points to be inspected in order to give a good indication of the item or circuit conditions. Inspection timings are identified and the inspection activities are grouped together, in order to carry them out in the most beneficial and cost efficient way.

When the preliminary inspection plan is finished it is reviewed by the inspection planner together with personnel from the maintenance and operations departments. If no changes are made, the plan will be finalized and used for execution of the inspection activities.

5. Execution and evaluation

The results from the previous stage are then transferred to an inspection management database for execution of the inspection activities. The inspections are then executed in accordance with the defined plan for inspections. When the inspections are completed the findings are

evaluated to determine if the conditions are as expected, or if they are better or even worse. The results are then updated to the inspection database.

Based upon the inspection findings there will be decided if further actions are required as for example the need for carrying out maintenance activities. The inspection findings as for example measured wall thickness will then be used in a RBI re-assessment, and the inspection plan will be updated accordingly.

2.3 Introduction to Usability

In the following sub-chapter the term "Usability" will be defined, and the reader will be introduced to the application of usability activities in relation to software developments.

2.3.1 What is Usability?

The word "usability" is often used about two different but related concepts. Usability can either be used to describe the "quality" of a software application, or about the process or the application of techniques in relation to software design and development to ensure a product with high quality. The latter concept is often referred to as usability engineering, which is a term used in this thesis.

Usability is probably a word that not many non-software or usability engineers have heard of or have a clear understanding of and therefore needs some explanation. Usability can generally be explained as a quality measure of how easy a product is to use in order to solve an intended task. The product can either be a software application, web page or any other product which requires user interaction. In this thesis the term "usability" will be used about software applications unless otherwise stated.

The term "user friendly" is something which most people have some kind of understanding of or relation to. In fact, the terms "usability" and "user friendly" have many things in common. The term "user friendly" was used earlier in software engineering but over the years its content had become vague and it no longer had a clear meaning; therefore it was replaced with the term "usability" (Folmer & Bosch, 2004). To get a more in-depth understanding of the term "usability" it will be necessary to define it more precisely.

So how is usability defined? Well, there are many different definitions of usability and no two are alike. Goodwin (1987) states that usability is too often discussed in abstract terms and, because of this, such terms do not offer any specific guidance. The latter was certainly found to be true, and therefore a literature survey was carried out to define it more accurately.

Generally, usability is defined as a quality measure that is the result of several attributes, often referred to as usability attributes. The level of usability can be measured or assessed by evaluating how a software application performs in relation to the defined usability attributes. How usability is defined varies with the attributes that the author or standard identifies to be important for the software application. To explain this more clearly, two different usability definitions will be presented, and compared and discussed in the next section.

ISO standard 9241-11 defines usability as:
"*The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.*"

The definition highlights three usability attributes: effectiveness, efficiency and satisfaction. Effectiveness in this context is used about the users' ability to complete the given task, while efficiency refers to the resources needed to complete the task, and satisfaction refers to how the experience complies with user expectations (ISO 9241-11, 1998). This might be more clearly understood if it is explained with the use of an example.

In today's technological environment, the use of software applications is widespread. Take, for example, the use of word-processing software. In order for such a software application to be *effective* it must allow the user to create a written document that fulfils its purpose of use. If the document is a job application, the software must allow the user to edit the document to meet the reader's expectations and requirements. Further the *efficiency* of the software could be the time used to finish the application. In order to defend its use the time spent should be equal or less than completing the job application either by hand or using a conventional typewriter. Satisfaction will, in this example, be the user's impression of the overall process of creating, writing and finishing the job application.

By measuring or assessing each of the three usability attributes, something can be said about the usability of the software. By "measuring", this thesis refers to a usability evaluation where the attributes are measured quantitatively, while "assessing" is a qualitative assessment of the different attributes based upon evaluator judgment. Often the attributes are defined in more specific attribute components which are easier to measure; this can be - as mentioned in the example above - time used for efficiency or how accurate the results are for effectiveness. More examples of such can be seen in Figure 2.3.1 and 2.3.2. As mentioned before, how usability is defined varies from definition to definition, which also means that the usability attributes and their components vary.

The second usability definition is presented by one of the pioneers within the field of usability: Jakob Nielsen.

He defines usability as (Nielsen, n.d. A):
"*Usability is a quality attribute that assesses how easy user interfaces are to use."*

He further defines usability as consisting of five attributes: learnability, memorability, errors, efficiency and satisfaction. The five attributes and their explanation are paraphrased from Nielsen's homepage *useit.com* (Nielsen, n.d B).

- Learnability is a quality measure of how easy the software application is to learn, and how easy it for the user to carry out basic tasks the first time they use it.

- Memorability is a measure of how easy it is to recollect the use of the software application. This is important for users who only occasionally use the application, to allow them to easily regain their proficiency and thus they do not have to learn to use it all over again.

- Error(s) is a measure for how many errors the user would make, the severity of the errors, and how easily the user can recover from them.

- Efficiency is a measure of how quickly the user can carry out the intended task with the use of the software application. Once the user has learned to use the software, it should allow for him/her to carry out the task in a time-efficient manner.

- Satisfaction is a measure of how pleasant the software is to use. The software application should solve the given task in a way that the user finds pleasing and satisfactory.

These five attributes can then be used to evaluate the graphical user interface of the application either qualitatively or quantitatively. Often these usability attributes are used by usability experts to perform a qualitative evaluation of the user interface. If it is desired to carry out a quantitative evaluation, it would be necessary to define what attribute components to measure, for the efficiency attribute this could be, for example, to measure the time used to carry out the intended task.

Nielsen states that the overall *usefulness* of the software application consists of two aspects, software *usability* and *utility* (Nielsen, n.d. A). While usability in regard to Nielsen's definition only relates to the graphical user interface, the utility describes the software application's functionality.

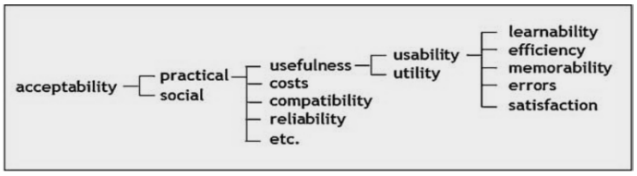


Figure 2.3.1. Nielsen's definition of usability
(Source: Folmer & Bosch, 2004)

Figure 2.3.2. ISO 9241-11 definition of usability
(Source: Folmer & Bosch, 2004)

In Figure 2.3.1 and 2.3.2 an overview is presented of what usability attributes and components the two presented definitions contain. In the author's opinion, the ISO 9241-11 definition takes into consideration all aspects of the software application's use, and its meaning is closer to what Nielsen characterizes as the software application's usefulness. What Nielsen defines as utility is included under the ISO standards' definition of effectiveness of the software, in other words, that the application has the functionality or ability allowing the user to reach the intended goal by the use of the software. Therefore it can be stated that what is defined as usability in the ISO standards is what Nielsen defines as software usefulness.

The above comparison is an example of the differences between how standards and usability experts define usability, and which attributes are considered to be important. But even if they differ in some areas, usability definitions largely overlap. This was also found to be the case in a survey carried out by Folmer and Bosch (2004). They compared several usability definitions and found that they were very much alike; the main differences were the attribute names, their combination and which attributes the authors state to be important for usability.

So, based upon the above comparison, it can be stated that there is not *one* correct definition which allows evaluators to measure or assess the "quality" or "ease of use" of the software. Therefore, the definition which is to be used in relation to usability activities should reflect those attributes that are considered as being important for the specific software's usability. If

it is important that a task is carried out fast, the software efficiency should be measured and more specifically time used (temporal) to solve the task.

In this thesis the author will use Nielsen's definition of usability, but with one modification. This will be that functionality (utility) will be included as a part of the five usability attributes already defined by Nielsen. This can be defended, as lack of functions will have an impact on all the five above-mentioned usability components. Goodwin (1987) states that functionality itself can determine usability, as lack of functions required to carry out the task can cause the software to become unusable. Nielsen (Nielsen, n.d. A) also states that to evaluate the software's functionality one can adopt the same methods used for evaluating usability. As the goal of the usability effort in this thesis was to find the strengths and weaknesses of the software application, it was desired to evaluate these two aspects at the same time.

The reason that some of the definitions distinguish so clearly between functionality and usability is that when developing a new software application, each of these aspects has to be evaluated carefully. Therefore, some definitions emphasize this by separating them as two different aspects, as is the case with Nielsen's definition of usefulness. As this thesis will evaluate an existing software application, it is easier to consider usability and functionality (utility) as a single quality measure, and define that as usability.

The choice to use Nielsen's usability definition is based upon the fact that it allows for a qualitative assessment of the software, and its attributes are all relevant for the usability study of Orbit Offshore.

2.3.2 Methods for evaluation of usability

Usability engineering is the discipline of carrying out usability evaluation activities in relation to software development to assure a software application with a high degree of usability. There are many different methods and tools for carrying out usability evaluations. This thesis will not give an introduction to all of them as that is out of its scope, but an explanation of the main types of usability methods available will be given.

Zhang (2001 cited in Folmer &Bosch, 2004) states that there are three main categories of usability evaluation which are:

- Usability testing
- Usability inspection
- Usability inquiry

To get an understanding of what these different methods involve, a brief introduction will be given to each of the categories. The following sections are based upon the evaluation methods which Folmer and Bosch (2004) presented in their article.

Usability testing requires representative users to carry out typical tasks using the software or a prototype of it. An evaluator observes how the user works with software and notes problems that the user encounters while using it. This makes it possible for the evaluator to obtain information about how the software supports the user in carrying out the intended task. The identified problems are then corrected and a new test will then be carried out to confirm that the problems are corrected.

Usability inspection requires usability specialist or software developers, users or others to compare how the user interface of the software application conforms to established usability principles and standards. The problems identified by the evaluator may then improved in the software application.

When carrying out a usability inquiry, the evaluators obtain information about what the user needs, his/her understanding of the software, and what he or she likes or dislikes. This is done either by discussing the software with relevant users, watching them use it for actual work, or through the use of questions or questionnaires. Based upon these findings, it is possible for the evaluator to come up with recommendations for the improvement of the software.

Usability testing and inquiry allows for measuring usability quantitatively, as for example how many errors a user makes while carrying out a task or the time used to complete it. It is also possible to obtain quantitative information about the user's satisfaction through questionnaires and the like. Such methods may be very costly and time-consuming to carry out as several users have to participate to get accurate results and the equipment needed to carry out some of these evaluation methods is expensive. Some companies have even built their own usability testing labs. Such labs allow a user to work with the application on a computer in a closed room, while the evaluator can observe the user's actions on his computer and the user's behaviour through a one-way mirror, thus making it possible for the evaluator to observe the user and his actions, without disturbing him.

Usability inspection, on the other hand, is a way of determining the usability qualitatively. The evaluators use the software application in question and evaluate how the user interface conforms to usability principles and standards. Such an evaluation is cost-efficient but may not be as accurate as usability testing and inquiry, mainly due to the fact that the identified problems are based on the evaluator's judgment and preferences.

So, when choosing an evaluation method for a software application, it is important to take into consideration what the goal of the evaluation is and the extent of usability effort which is necessary.

2.3.3 When to carry out usability evaluations

Usability activities can be carried out prior to, during or after completion of the software's development. When carrying out usability evaluation prior to new software development, the usability effort is meant to identify which parts of the existing software are good and work well and therefore should be implemented in the new software. Obviously, it should also be the goal to identify which problems cause trouble for the user and therefore should be avoided in the new software. The author found very little information about how to carry out such an evaluation which indicates that there is not any specific approach to carry out such activities.

Usability activities are mostly carried out during the design and development phase of the software. This is to evaluate the software usability and identify problems as early as possible in order to correct them with the least amount of cost. If faults and problems are identified early in the design and development phase it is easier for the software engineers who are programming the software to correct them.

Ideally, usability activities should be carried out as early as possible to allow for a complete redesign of the user interface (Lauesen, 1997). Therefore, it is common to test prototypes of

the software during the design and development phase. This can either be done by using a programmed prototype of the software user interface or by sketching it on paper and testing it on users.

Nielsen (Nielsen, n.d. B) states that the user interface should be designed iteratively as it is virtually impossible to design a user interface perfectly in one try. Iterative design is carried out by testing a design prototype on actual users to identify what problems they have with it. The problems are then corrected in a new iteration, and a new test of the software is carried out to identify whether the problems are solved, and to possibly identify new usability problems (Nielsen, n.d. B).

In relation to the iterative design process, Nielsen (Nielsen, n.d. B) recommends to start testing the user interface on users early in the design process, and to carry out testing at every step of the design and development phase. He further states that this is the only way to get a high quality user experience. The use of the iterative design process is a way of improving the user interface based upon the lessons learned from previous iterations (Nielsen, n.d. B).

When the software development is completed, usability evaluations are carried out to ensure that the software applications fulfil usability requirements. If that is not the case, some minor changes may be made to the design, but there are very limited possibilities for such without the need for major reprogramming of the software. This is why the usability effort is so highly emphasized in the early stages of the design and development process.

3. Development of a framework for the usability study of Orbit Offshore

3.1 Introduction

Due to the specific nature of the project there were not found any suitable usability approach in the literature. Hence, the methodology suitable for the project was developed. While the main concepts were taken from the literature, some sections were modified to suit the specific requirements. This chapter presents the framework developed to carry out a usability study of Orbit Offshore. The results of the study will be presented in the next chapter.

3.2 Framework for the usability study of Orbit Offshore

The approach for the usability study of software design is based upon a usability design approach presented by Eric Schaffer in his book *"Institutionalizing of Usability" (2004).* The method which is simply called *"The Schaffer Method"* is based upon a usability engineering methodology called user-centred design (UCD). User-centred design is a structured development process where the organizational business objectives and the user's needs, limitations and preferences are used as guidelines throughout the whole design and development process (usability.gov). The major advantage of the Schaffer method is that it clearly identifies the activities that should be carried out in the design and development process. In Appendix 3 an overview of the activities related to the Schaffer Method is presented.

Even though the Schaffer Method originally is an approach to assure software usability, it could be argued that some of these steps are also important when carrying out a usability evaluation. Thus, in the framework developed, some activities are similar to those found in the Schaffer Method but the sequence of the activities has been rearranged.

Two of the activities which were included in the framework were to define the user and identify the organizational requirement. As the planned evaluation was to be applied to a very technical software application, it would be beneficial to evaluate how the usability of Orbit Offshore would be in relation to the expected computer knowledge of a new potential user. This can be important as the characteristic that makes a software application usable for one user group may make it unusable for another (Goodwin, 1987).

In other words, the same features which help new users to easily carry out a task and prevent them from making mistakes, can be a limiting factor for how efficiently an experienced user can work with the software. New users may need a wizard to help them carry out a task, while the experienced users may not have use for such a feature. If it is not possible to carry out the task without the use of the feature it will have a significant impact on the experienced users' efficiency. Therefore, what has a positive impact on one user group's usability might have the reverse effect on another. The reason for this is that the differences in skill level will reflect which aspects of the program the specific user would characterize as having an impact on the software's usability. So, in order to take into account such considerations in relation to the usability evaluation of Orbit Offshore, a user has to be defined.

Another aspect which the author thought to be important and decided to include as an input to the evaluation was how the software conforms to the organizational requirements. For technical software, as for all software applications, it is important that it fulfils its purpose and completes the intended task. A software application cannot have a high degree of usability if it is not capable of carrying out the intended task. Therefore, in order for the software to have

a high level of usability, it should not only be "easy to use" but it should also fulfil the requirements set by the organization. Therefore, the organizational requirements should be identified and used as input in the usability evaluation.

The framework developed for the analysis of the software design of Orbit Offshore is presented in Figure 3.1. The figure shows that the approach is divided into six steps. Each of the steps presented in the figure and their content will be explained below.

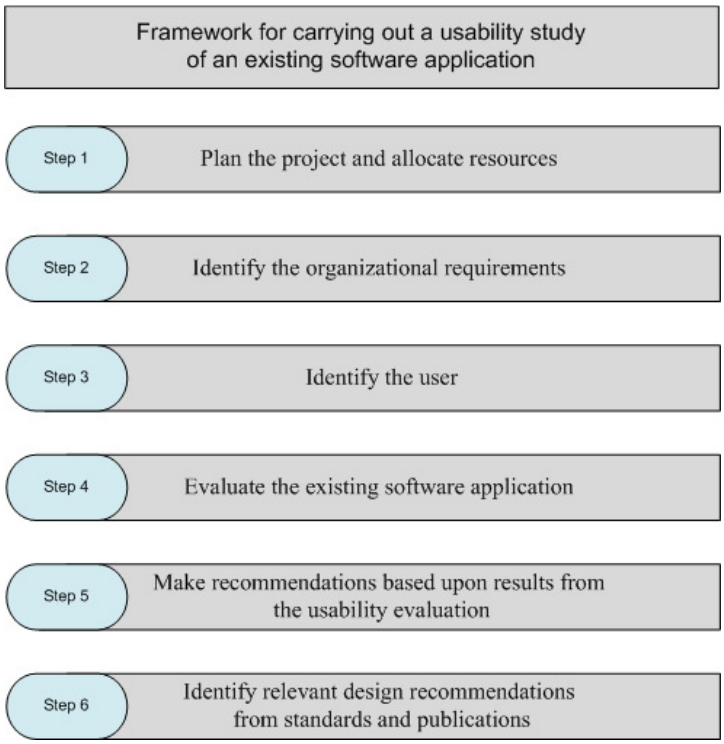


Figure 3.1. Framework for the usability study of Orbit Offshore

Step 1 :. Plan the Project and Allocate Resources

In the first step of the usability approach, the project goals should be identified. From the identified project goals, the extent of the usability work required can be decided and resources needed to carry out the usability work can be assigned.

Step 2 :. Identify the Organizational Requirements

In the second step the company or organizational requirements should be identified. These are requirements that the software should fulfil in order to be a tool which can support the company's business objectives. By identifying the requirements, it should be possible to evaluate whether the software meets the objectives, and then use them as a basis for recommendations for future software developments.

Step 3 :. Identify the User

In order to evaluate the usability of the software, it is important to know who the users are and level of their computer skills. Based upon this information it will be possible to evaluate if the software conforms to the user expectations and to what degree the user would be able to carry out the specific tasks in the software. As with

the organizational requirements, the identification of the potential user can be used as a basis for the recommendations for future software developments.

Step 4 :. Evaluate the Existing Software Application

Evaluation of the usability of the software can be then carried out using the chosen evaluation method. Information about the organizational requirements and the users are used as an input when carrying out the usability evaluation. The method adopted for carrying out this stage will be presented in Section 3.3.

Step 5 :. Make Recommendations based upon Results from the Usability evaluation

Based upon the findings from the usability evaluation and the identification of the software's strengths and weaknesses it should be possible to suggest some specific recommendations for future developments.

Step 6 :. Identify Relevant Design Recommendations from Standards and Publications

In this stage guidelines and recommendations from relevant standards and publications are used as a basis for specific recommendations.

By adopting this approach, it is possible to have a framework which incorporates important activities related to the evaluation of the existing software, and also helps to identify recommendations for future developments.

3.3 Evaluation of Software Usability

A literature survey was carried out in order to identify the most applicable usability evaluation method for this specific project. As the goal of the project was not to carry out a traditional usability, it would have to be a usability method which could be adapted to the specific project goal.

The adopted usability evaluation method should make it possible to identify Orbit Offshore's strengths and weaknesses in relation to the task of carrying out the RBI analysis. In other words, the method should allow for the evaluator to take the RBI methodology into consideration when carrying out the evaluation.

In order to achieve the project goal it would not be useful to carry out a quantitative usability evaluation, for example measuring the number of faults users make or the time spent by users to carry out a certain task.

If usability testing or inquiry is to be properly carried out, it is important to have quite a few users to participate in the evaluation activities in order to get representative results. Unfortunately, in this study only three experienced users could participate in the usability evaluations. Hence, the choice of a possible usability evaluation method should take into account this limiting factor.

Taking these factors into account, it became clear that a qualitative assessment and the use of a usability inspection method would be the most beneficial approach. Usability evaluation methods found in this category mainly relate to the action of judging whether a software application's user interface conforms to usability principles and guidelines. Heuristics evaluation was chosen as the specific method to evaluate the usability of Orbit Offshore. This is a method in which the evaluator examines how a user interface conforms to a specific set of

usability heuristics. The heuristics can be better explained as rules of thumb or usability guidelines which, if followed, should ensure a software application with a high level of usability. These rules can either be used in the design and development process or they can be used to evaluate the usability of existing programs (Faulkner, 2000, pp.-189).

There are several different sets of heuristics which can be applied to software applications. The use of heuristics to evaluate usability is actually the most popular usability inspection method (Nielsen, n.d. C). This is mainly because it allows for a fast and cost-efficient evaluation of the software's usability. There are many different sets of heuristics and principles published by authors, but some of the most widespread heuristics were found to be Nielsen's ten heuristics, Ben Shneiderman's "Eight golden rules" and Norman's "Seven principles for transforming difficult tasks into simple ones".

In this project, Nielsen's ten heuristics have been used for the usability evaluation of Orbit Offshore. The choice of heuristics was largely because Nielsen's heuristics was found to be easiest to adopt, and also because differences between the various sets of heuristics were found to be minor. Faulkner (*Usability Engineering*, 2000, pp.179) states that the content of these heuristics largely overlap and the same conclusion can be claimed from the comparison of heuristics and principles done by Keinonen (1998, cited in Folmer & Bosch, 2004 pp. 72).

Nielsen's ten heuristic rules are presented below. These rules are quoted from Nielsen's own homepage *Useit.com* (Nielsen, 2005). The heuristics are quoted in order to give the reader the opportunity to consider the heuristics exactly as they are originally formulated.

Rule 1 :. Visibility of System Status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Rule 2 :. Match Between System and the Real World

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

Rule 3 :. User Control and Freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Rule 4 :. Consistency and Standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Rule 5 :. Error Prevention

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Rule 6 :. Recognition Rather Than Recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Rule 7 :. Flexibility and Efficiency of Use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Rule 8 :. Aesthetic and Minimalist Design

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Rule 9 :. Help Users Recognize, Diagnose, and Recover from Errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Rule 10 :. Help and Documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

The heuristics highlight ten aspects of the software which are important for usability. For each of the points there are also specific guidelines for how to ensure conformance to the ten heuristics. Nielsen points out that it is more correct to view these heuristics as rules of thumb rather than specific guidelines which *should* be followed (Nielsen, 2005). Therefore, the heuristics should be applied subjectively, and be used to identify aspects which are considered to have a negative impact on the specific software application.

As this evaluation method relies upon the evaluator's subjective judgment, it would allow for the evaluator to take the RBI methodology into consideration while assessing the software's usability. Identification of the aspects that would have a negative and positive influence on the software usability could therefore only be achieved by applying the evaluators' subjective judgment. Through the use of the software application it would be possible for the evaluator to compare the software's conformance to the specific heuristics and possibly identify the RBI software application's strengths and weaknesses.

A drawback with heuristics evaluation is that the evaluators may overlook usability problems if the software is highly domain-dependent (Nielsen, n.d. D). As the evaluator in this specific project would have knowledge of the domain, this would not be an issue. That the usability evaluator may overlook problems if the software is highly domain-dependent was one of the main reasons for not carrying out a conventional usability evaluation.

Nielsen (Nielsen, n.d. E) states that the heuristics evaluation is best carried out using at least three to five evaluators for it to be an effective method because use of fewer would not be effective in finding all the usability problems. Since the goal of this project was not to identify *all* the usability problems, but rather carry out a usability evaluation to identify the software strengths and weaknesses, fewer evaluators would suffice.

Nielsen also advocates that even if the application of the heuristics may not lead to apparent solutions to the identified problems, it would be possible to recommend improvements as the problems are identified with reference to a specific usability principle (Nielsen, n.d. E). To explain this in other words, as the usability problems are identified through their non-compliance or nonconformity to a specific usability principle, it should therefore be easier to identify the correct actions to improve the problem. For the application of this method in relation to Orbit Offshore, this meant that it would be possible to identify specific aspects which should be emphasized in future software developments. Nielsen also states many of the usability problems have obvious solutions when they have been identified (Nielsen, n.d. E).

In this usability study the heuristics would be used not only for identifying usability problems but also for identifying the factors that can have a positive impact on the software usability. Even if this was not an advantage identified by Nielsen or any other usability experts, the author believes that this should be possible through the experience of using the software application.

4. Application of the Proposed Framework for the Usability Study of Orbit Offshore

4.1 Introduction

In this chapter the different activities related to the usability study of Orbit Offshore will be presented. The chapter is built around the application of the proposed framework, therefore each of the steps will be presented in the sequence in which they were defined in the previous chapter.

4.2 Step 1: Planning the Project and Allocation of Resources

As mentioned earlier, the goal of the project was to carry out a usability evaluation of the current version of Orbit Offshore. The necessary resources were identified and the author was granted complete access to the software. A short training course on the use of the software was also given.

The most beneficial usability evaluation method was identified and a framework was developed for the application of the method and the related activities. Based upon the chosen approach, the software was evaluated.

A number of DNV employees who took part in the development of the software and experienced users were made available for interviews in order to understand design philosophy and their on-job experiences on the use of the software.

4.3 Step 2: Identification of Organizational Requirements

The organizational requirements presented below were identified from the Orbit Offshore strategy document which was written when the software was first developed. Even though these requirements were formulated more than seven years ago, they should still apply to the current application, as well as future RBI software developments. The four organizational requirements are quoted and explained below.

- *"ORBIT must be continually at the forefront of RBI methodology and technology"*

To allow for DNV to provide world class RBI service to their clients, the software should use the latest technology and RBI methodology to ensure such a service.

- *"ORBIT must allow for the most efficient working process"*

This requirement more or less speaks for itself. The software should allow for the user to carry out the RBI analysis in the most efficient way possible. Efficient here refers to the time spent and user effort needed to carry out an RBI analysis with the use of Orbit Offshore.

- *"ORBIT must present the inspection planning results in the manner required by clients"*

RBI analysis is a service which DNV provides to their clients. In order to fulfil the customer requirements and expectations for such a service, the results from the RBI analysis must be presented in accordance with client specifications. The results in question are time to risk limit for all parts, an inspection plan containing those tags/parts which should be inspected, inspection timings and recommended inspection method.

- *"ORBIT must support the business objectives of DNV"*

Orbit should support DNV's business objective of being a leading international provider of services for managing risk. It can do so by allowing DNV to offer its clients the highest possible quality of service.

4.4 Step 3: Identification of the User

In order to evaluate the software application's usability and to indentify it strengths and weaknesses, it was important to identify the potential user of the software application. The reason for this was that the strengths and weaknesses are related to a user's computer skill level, or knowledge of the RBI.

The potential user of the software was defined as a person with average computer skills, meaning that the user had general knowledge of the operating system under which the software runs and was able to orientate him/herself with the use of multiple windows, navigation through standard application menus and other basic software interactions.

The user was assumed to have a good working knowledge of commonly used Microsoft Office applications like Word, Excel and Outlook. Furthermore, it was assumed that the user would have higher technical education and that he or she was familiar with RBI methodology and working process.

4.5 Step 4: Evaluation of the Existing Software

While using Orbit Offshore, each of the ten heuristic rules presented by Nielsen was applied to the software, in order to evaluate the software's conformity to the specific heuristic rule. The evaluation was carried out by using Orbit Offshore to redo the steps of a previously conducted RBI analysis. The author was given a short introduction to the use of Orbit Offshore prior to the evaluation, but aside from that no training was given in the use of the software. The analysis was therefore carried out with the aid of the help document and the already completed RBI analysis project files as guidance.

Opening two software sessions, and opening the complete analysis in one and the ongoing in the other application window, made it possible to compare the two with ease. The latter turned out to be a good way of learning how to use the program, and made it possible for the author to carry out the steps of the analysis and compare the correctness of what had been done to the already completed analysis. It is important to emphasize that the author did not carry out a complete RBI analysis as this would take far too much time, but rather carried out the different steps in the program with fewer parts.

4.5.1 Application of Nielsen Ten Heuristics

The findings are presented and explained with relevant examples as well as their impact on usability. After the initial presentation of the results, the findings have been summarized by discussing them in relation to the previously mentioned usability attributes as defined by Nielsen.

4.5.1a Nielsen's Heuristic Rule 1: Visibility of System Status

System status can be explained as the state at which the software is currently, for example if the software is busy working on a task, or if it is idle and ready for user input. Such states should be presented in a clear manner allowing for the user to immediately identify the software status.
In Orbit Offshore it was found that the visibility of system status needs to be improved. Lack of such feedback may cause confusion regarding the proper completion of the task in progress. This can be explained with an example. When the different calculations for CoF, PoF and risk have been completed in the software, no feedback of the successful completion of the task is given. If required data is not available the software gives the most conservative estimation (example, PoF = 1). A better presentation would be to display a notification message informing the user about the missing data. Such information would allow the user to identify the state the system is in, and if corrective measures need to be carried out before proceeding to the next stage of the working process.

As another example, in the current version, the date of last calculation can be retrieved from the calculation report (CalcReport) associated with every tag. A better method would be one in which every time the data is updated, markers identifying the subsequent calculations should appear. These markers should stay till the associated calculations have not been updated.

4.5.1b Nielsen's Heuristic Rule 2: Match Between System and the Real World

To make it easy for the user to work with the software application, the working process of the software should be aligned with the working process in real life. It has been found that this is the case for Orbit Offshore. The software application has a window where all the steps needed to carry out an RBI analysis have been identified. The steps and their sequence was found to be aligned with the steps mentioned in DNV Recommended Practice RP-G101 and should therefore be intuitive and logical for a user familiar with the RBI methodology.

Secondly, the forms provided in the software should match those commonly used by engineers in real life, which is the case for Orbit Offshore. An example of this is the screening form to be used when carrying out the screening session together with clients' experts. It has been developed to allow for easy and quick input of data that is required for screening.

Thirdly, the language used in the software must be the one that the end user is familiar with. Generally this was found to be the case with the current Orbit Offshore. The language and concepts are aligned with the domain of the program which, in this case, is the RBI methodology. Unfortunately, there are some field names within the software tables which use terms generally used by programmers instead of the terms used by maintenance engineers.

A problem which has been identified with the software is that the screening process and the detailed analysis are disjointed. It has been found that when the screening is completed, the results cannot be transferred into the following detailed analysis. In other words, the screening and detailed analysis is carried out independently. Once the screening has been completed, to carry out the subsequent detailed analysis, the user has to identify the parts that are required for further analysis and then input the relevant parts and data for the next stage. This is something which is unnecessary and obviously needs be done in a different way.

Based upon the above findings, the software can be said to be to following the conventions for carrying out an RBI analysis. A user who is familiar with the RBI methodology and working process should easily recognize and be comfortable with the working process in Orbit Offshore. On the other hand, there is some scope for improvement which can have an impact on the software's usability.

4.5.1c Nielsen's Heuristic Rule 3: User Control and Freedom

User control and freedom can have a significant impact on user satisfaction. If the user gets an impression that he or she is not in control of the software then it might be a cause for user frustration and subsequently have a negative impact on the users' attitude towards the use of the software. While Orbit Offshore provides good user control and freedom, it has been found that there are some areas where it requires improvements.

A function which has been found missing in the software is the possibility to undo and redo actions. This is a function which is available in almost every software application, and therefore the user is accustomed to the use of this function. When working in large tables a user might accidentally delete or wrongly input data. If the former should happen and data is deleted in Orbit Offshore, the user has to manually restore the data, for which the user may have to use previous versions of the document. At times this can be a time-consuming task. Therefore, when working with large amounts of data, it is essential to have the option to undo or redo steps.

Another function that would be beneficial is the option to exit the software without saving changes to the project files. There could be instances where the user just wants to check the project results or print reports, and therefore does not want to risk saving any unintentional changes. The current version of the software saves all changes when exiting without regard to the user's wish.

4.5.1d Nielsen's Heuristic Rule 4: Consistency and Standards

The relation between icons used on buttons, and the functions they carry out is important for the usability of the software. If the user does not understand what action or function the button carries out, he or she will hesitate to use it. The choice of icons should therefore be based upon similar use in other relevant programs which the user is likely to use in his line of work. This will contribute positively in relation to user satisfaction as the user would more easily recognize the correct "buttons" and would not have to search for the action or function he is looking for. This will also help new users of the software to more easily recognize actions and functions and will ease the learning and familiarizing process.

It is important that icons are updated and are comparable to the existing icons used in other programs at the given time. If this is not the case, new users will not immediately recognize the functions or commands attached to the icon used. Experienced users may forget over time and will therefore be more familiar with the icons and layout that is recently used. If, for example, Orbit continues to use old icons and all other programs that the user utilizes have a different set of icons, these differences could result in confusion for the user.

As Orbit Offshore is based on Access, most of its icons are consistent with other programs in the Microsoft suite, and will therefore be familiar for most users. An overview of the software buttons is shown on the next page.

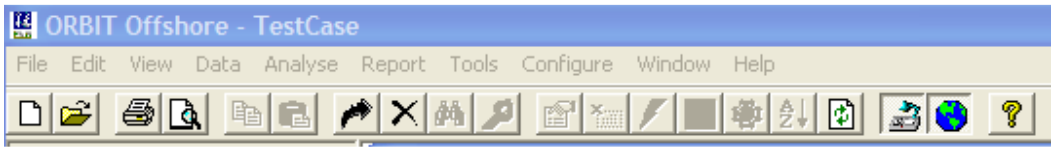


Figure 4.1. Icons used in Orbit Offshore

Apart from the familiar icons, there are some icons that may not be immediately familiar to a new user. For example, the purpose of the "World" button, shown in Figure 4.1, is to toggle on/off the explorer field which allows for filtering of parts data, and of the "Desktop" button to toggle on/off the overview of the working process. Unfortunately, the purpose of "Desktop" and "World" buttons may not be evident to new users.

The organization of Orbit's menus is similar to what is found in other programs based upon the Microsoft Windows architecture. The groupings within the menus are logical and make it easy for the user to find the functions or actions he might be looking for. An exception to the latter is that the user is not presented with the option to carry out calculations unless the parts table which contains all the parts information is selected. This may have been done intentionally to allow the calculations to be performed on filtered groups of specific tags. On the negative side, the lack to choice of carrying out the calculations without selecting the table gives the user the impression that he is not in control, which will have an impact on user satisfaction.

4.5.1e Nielsen's Heuristic Rule 5: Error Prevention

Prevention of errors is especially important in an RBI analysis because the reliability of the final result depends upon the reliability of the data used in the calculations. Since the number of tags in an analysis can range from 2000 to 30 000, it is nearly impossible for the user to manually evaluate the reliability of each and every single tag data. Therefore, to get a reliable result there must be some measures to prevent errors arising due to missing or incorrectly filled in data. As the software uses tables/spreadsheets as the primary interface for input and evaluation of data, it is difficult to find the exact field which is missing data. To make it easier to resolve the problem, data fields where input is required could be highlighted in a red colour, making it more obvious for the user where error correction is needed.

One of the major problems found with the software is the lack of fault checking to prevent errors. The lack of error prevention can be explained with an example from the probability of failure (PoF) calculation. If required data - either temperature or pressure - is missing for a tag, correct calculations cannot be carried out for that specific item. The reason for this is that without the data it is not possible to use the degradation model to calculate the probability of failure. Instead of informing the user about the missing data either before or after the calculation has been completed, the software allocates the tag with PoF equal to one. Due to the lack of message, the user will not expect any error to have occurred during the calculation. This problem could easily be resolved if the user is presented with a dialogue box informing about the missing data, and giving him the choice of correcting the data or continuing with the calculations.

The software should also prevent the user from inputting incorrect or faulty information if possible. Currently, Orbit Offshore works on the line list provided by the clients, as such it is assumed that the client has provided an error-free line list. An improvement in the software can be made by incorporating the expected limits of the data. In case the data does not fall within the set limits, then the data can be highlighted for manual checking.

A more efficient error prevention feature in the new version may save considerable time. It would also increase user satisfaction and software efficiency and thereby also the overall usability of the software application.

4.5.1f Nielsen's Heuristic Rule 6: Recognition Rather than Recall

Generally software should provide an informative interface allowing for recollection of steps and actions rather than making the user recall the exact working process. For RBI software this is certainly an important aspect as there are many steps and related actions that have to be carried out, from project start to reporting of results.

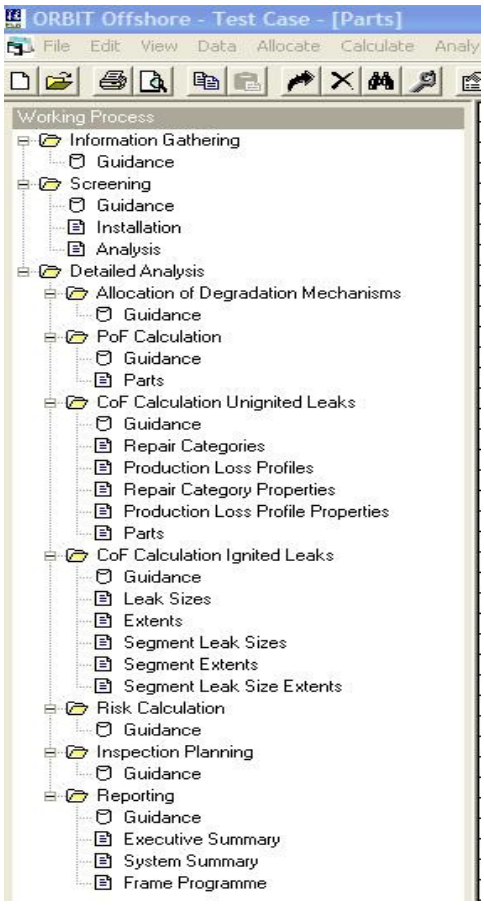


Figure 4.2. Overview of the RBI analysis process

Orbit Offshore has a good feature to guide and remind the user of steps and actions that are necessary for the RBI analysis. The feature is a navigational window presenting the working process as a tree structure (Figure 4.2). The tree structure consists of the three stages of information: gathering, screening and detailed analysis. The detailed analysis part is divided into sub-stages for the calculation of PoF, CoF, risk, inspection planning and reporting. Within each stage or sub-stage there is a list of tables and dialogues which is required in order to carry out the specific task or calculation. Each of the listed items doubles as a shortcut and opens their respective tables and dialogues, which makes it easy to navigate through the different stages of the RBI analysis.

In addition to the tables listed in the working process window, there might be a need to configure one or more background tables. These background tables consist of predefined values which are used as input in the parts table. Example of such background table is the materials table which consists of physical property values for the most common parts materials. These data may have to be updated to reflect the specific in the project. It can be argued whether these background tables should be presented as part of the working process or not, as these do not require editing in every project. On the other hand displaying all the steps may help the user to more easily identify all the tables which have to be edited to carry out the specific analysis, and would therefore be a better aid.

Another feature that may be useful to aid in recollection of the working process is related to the choice of using data from an existing QRA as an input to the analysis. If data from an external QRA is available, this information can be used directly as input for consequence of

ignited leaks without the need to carry out the specific calculations. Available QRA data has to be inputted into a specific table, and the user has to choose *not* to carry out unnecessary calculations through the use of a dropdown menu within the same table. In other words, there is no easily accessible option to allow for input of QRA data without carrying out the unnecessary calculations, an option that obviously should be available. This is something which might be difficult for the user to recollect while carrying out such an analysis.

Another feature that is wanting in the Orbit Offshore is a system indicator showing how far in the working process the user has come. In order to carry out the calculations, the required data has to be filled into the right tables, and background tables used in the calculations may have to be configured. Dependent on the extent of the analysis, such tasks may take several days to complete for both CoF and PoF. Throughout this process it can therefore be difficult to monitor the tables that have been completed. This may lead to uncertainty about the work progression and whether the system is ready to carry out further calculations.

If an ongoing project is opened, Orbit Offshore will open the project file and show a standard opening screen, rather than presenting the last dialogue or table used. So, every time the project is reopened, the user has to recollect where he left off during the last software session before he can continue to work. The tree structure used in Orbit Offshore is also collapsed, which makes it even more difficult for the user to identify which part of the analysis as well as the specific table he was working on. As there are many tables the user has to work with, it can be difficult to recollect tables that have been completed. Therefore, it may be necessary for the user to manually check table content to confirm his progress. This could be avoided if there was a system status showing which tables and calculations that have been completed. Ideally, there could be a function allowing the user to "check off" the completed tables, thereby making the progression more transparent. An example of how this might work is presented in the figures below.

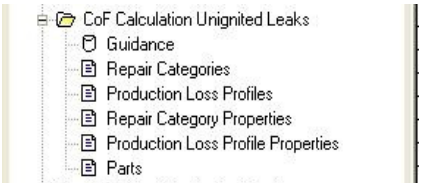


Figure 4.3. Example of the current working process without feedback on progression

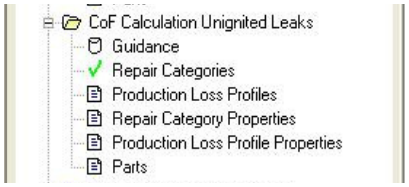


Figure 4.4. Working process with feedback on progression

Overall the software provides good guidance for its use, but there is still room for further improvements (discussed later in the thesis).

4.5.1g Nielsen's Heuristic Rule 7: Flexibility and Efficiency of Use

Software efficiency is an important aspect of usability. Efficiency can be explained as the resources needed to carry out the task with accuracy and completeness (ISO 9241-11, 1998). It has been found that the working process in Orbit Offshore is not optimal, resulting in slower progress. This part of the evaluation shall identify aspects which can have an impact on the overall efficiency of the program.

A deficiency in Orbit Offshore is its inability to import the line list independently of Access. Access is also used to organize the database tables using queries to input data directly into the right tables. This can have an impact on the efficiency during the screening and detailed

analysis. This could be resolved by implementing easy data importation possibilities into Orbit Offshore (discussed further later in this thesis).

As previously mentioned, the screening session is a way of assessing the parts, systems or segments that have potentially high risk, and therefore require detailed analysis. The screening part of Orbit Offshore is intended to be used while conducting the manual screening session together with the client personnel, allowing the user to input information directly into a form used for the screening analysis. Because of the vast amount of systems and subparts which needs to be assessed, this type of manual input is very time-consuming and not very efficient. Therefore, today it is a common practice for the engineer responsible for carrying out the RBI analysis to conduct a preliminary screening based upon information found in the line list and P&IDs provided by the client. The line list is a digital document, often an Excel spreadsheet, which can contain large amounts of detailed information about the installation parts and systems.

After the preliminary assessment has been carried out, corrections and missing information is identified, and the line list needs to be updated accordingly. If required data is not available digitally in the line list, such information must be collected from other sources, mainly paper-based; therefore the information may have to be filled into the tables manually. Orbit Offshore was found not to be optimized for manual input of data, or to work efficiently with the data already available in the tables.

An example of the use of a dropdown menu is presented the figure 4.5. When working with the parts table spreadsheet, dropdown dialogues have to be used where available. It is not even possible to speed up the selection process by typing the first letters of the desired choice; instead, the dropdown menu has to be selected and the appropriate selection must be confirmed by double-clicking. This, combined with the fact that it cannot be carried out for multiple parts at the same time, makes this type of data processing difficult and inefficient. Dropdown menus are helpful for novice users but in this case it would even slow down the working pace for such a user. The limited possibilities for an experienced user to utilize shortcuts and hotkeys to speed up data processing, results in Access being today the only viable option for effective use of the software.

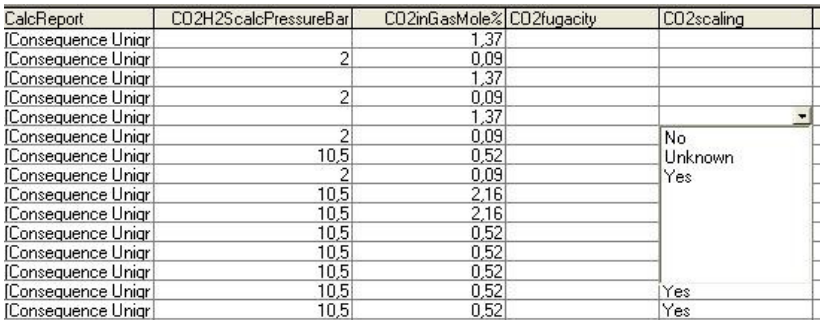


Figure 4.5. Example of the use of dropdown dialogue

After the screening is completed, the results are fed into the Orbit Offshore database through the use of Microsoft Access for detailed analysis involving the calculation of PoF, CoF and risk calculations. Orbit Offshore has screening and detailed analysis as separate modules; hence, results from the previous screening assessment cannot be used directly for the detailed assessment. In order to import data into the parts table, where all required information about parts and systems needed for calculations is stored, it is again necessary to use Access and

queries. The program was originally designed around this method for importing data, but this is not an efficient method to import data.

Evaluation of results is an important aspect when carrying out an RBI analysis. Often this has to be carried out manually using an expert's knowledge and judgment. In Orbit Offshore the results can be evaluated either in tabular form, or in a separate window more suitable for the task. The results can be evaluated by selecting an item from the part table and then opening the "part analysis" window from the menu. In the part analysis window the user can evaluate the results from the different calculations. If errors have occurred, then at this stage the user has a possibility to apply his expert knowledge and re-evaluate the results. Unfortunately, it is difficult to use the latter method for evaluating multiple results efficiently, as it is not possible to scroll through the parts continuously. In other words, the user must leave the current window and reopen another in order to evaluate the next part.

Another limitation of the software regarding flexibility and efficiency is related to the limited reporting function currently available in Orbit Offshore. The software has a number of pre-made report structures for the user to choose from, but neither of the pre-defined reports can be adjusted with respect to the content or layout. Therefore, it is not possible to adjust the report to a specific project or as required by a specific client. In order to create a report that meets the specific client requirements, Access has to be used to tailor the report.

The installation data supplied by the client may not always be available in the same units as Orbit Offshore uses in its calculations. This is a problem in the software as it is not flexible in relation to using whatever units the data is provided in. An example of this is the input of safety risk limit which should be given in potential loss of life (PLL). However, the customer safety risk limit is not always provided as PLL, but may be defined as fatal accident rate (FAR). If the latter is the case, the user has to do some minor calculation before he is able to input the correct data. These calculations may be easy to carry out, but it slows down the working process and the user spends effort here which could be of better use elsewhere. A solution could be that the user is able to select the desired denomination which corresponds to the data available, such that the user would then have the choice of either FAR or PLL and fill in the correct data accordingly. This is found to be relevant for other input data as well, such as temperature (°C, F), pressure (psi, bar) and geometric dimensions (mm, inch) used in the calculations.

Based upon the findings above, the software can be said to have potential for an improvement when it comes to the efficiency and flexibility of its use.

4.5.1h Nielsen's Heuristic Rule 8: Aesthetic and Minimalist Design

The need for aesthetic design can be argued as Orbit Offshore is not a software application that needs to appeal to a large user group, but the design should be functional and allow for the user to easily perform tasks necessary to carry out RBI analysis.

In Figure 4.6 an overview of the software layout is presented. The overall software design may be considered to be minimalistic since the layout consists only of the minimum features required to carry out the RBI analysis. The layout is focused around the effective use of tables and interaction with them. Sometimes the expression "less is more" is a good rule, and this applies to Orbit Offshore to some extent.

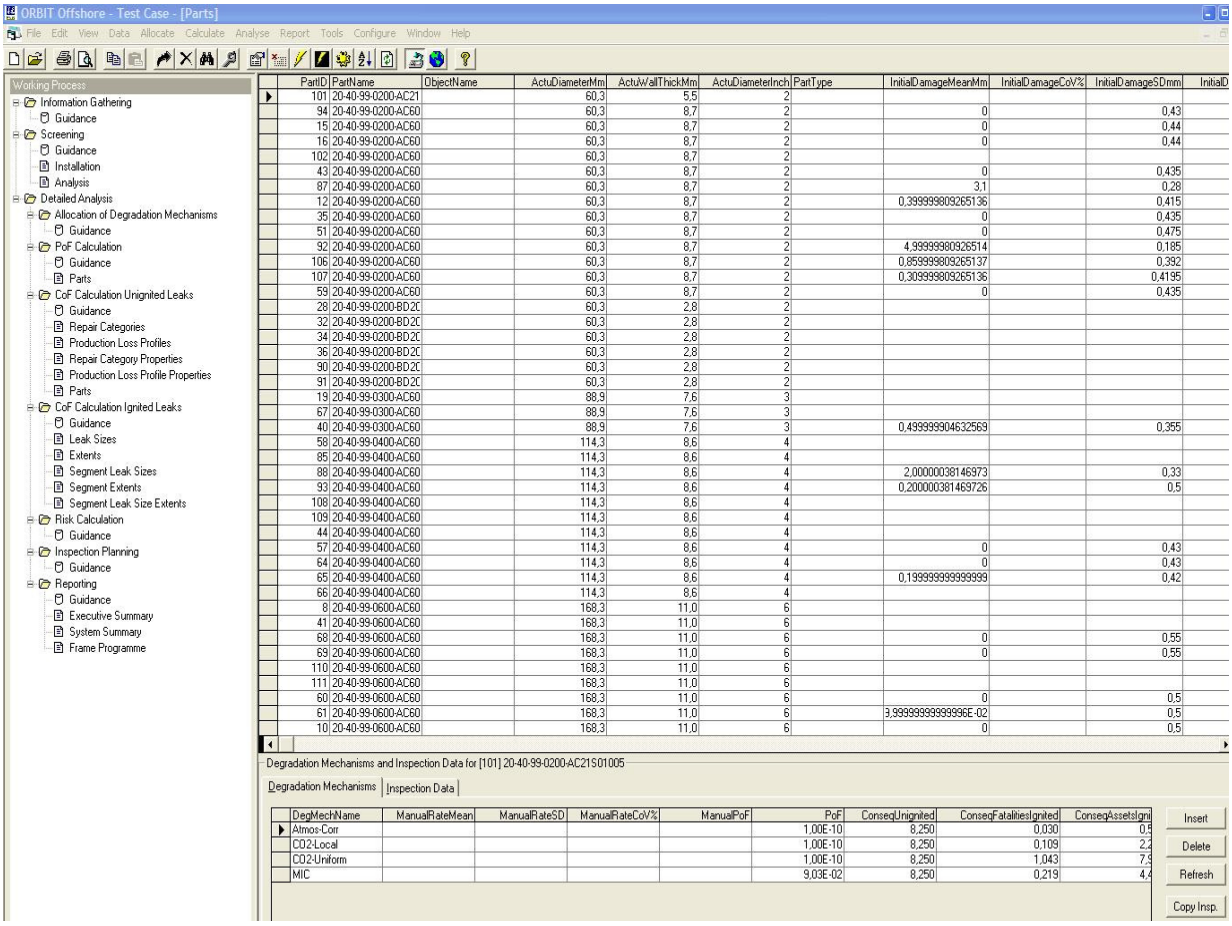


Figure 4.6. Overview of the user interface in Orbit Offshore

In spite of its overall good design, Orbit Offshore has some weaknesses. One of the problems with the current software is that its design resolution is low, i.e. the amount of information which can be displayed at a time is limited. When the latter is combined with the use of large tables, it makes it difficult to get an overview of all the fields in some of the tables, and therefore the scroll bars have to be used extensively. This can be troublesome when reviewing or editing table data fields. A possible solution could be to divide the tables containing many columns into several smaller tables to ease the experience of working with the tables.

4.5.1i Nielsen's Heuristic Rule 9: Help Users Recognize, Diagnose, and Recover from Errors

Good software must help the user to recognize, diagnose and recover from errors encountered during its use. Error messages generated must be as precise and informative as possible to allow for the user to identify and recover from the error with ease. It has been observed that in Orbit Offshore some of the error and attention messages provided by the software are not clear. As an example, a message encountered during an application of the software asked: "Do you want to save the data?" The dialogue gave no feedback on what data had not been saved and therefore made it difficult to establish the right action. There are two possible reasons for this; either desired changes to the data had been carried out by the user, or undesired changes had been made without the user's knowledge or intent. So the user had to decide whether to keep potentially incorrect data or to risk losing correctly applied data. This illustrates the reason for providing unambiguous messages.

As previously mentioned, the program does not give any feedback on missing data in relation to the PoF calculation. Again, take the example of the PoF calculation which can only be carried out if sufficient data is available. If a part defined in the parts table does not have the required data, for example lacking temperature or pressure, it is not possible for the software to proceed with the calculations as data needed for the degradation model is missing. Because of this, the software allocates this part with a PoF equal to one. As there is no mechanism to highlight the tags that have been allocated the pre-defined value, the user has to manually identify them from the results. This can be done either by checking that all required information is available in the parts table, or by checking the calculation details for each item in a separate part analysis window. The same was found to be the case for the consequence of failure calculations; here the lack of data may result in some parts having an over-conservative estimation of the consequence of failure.

The above discussion highlights the need for improving the features that can aid a user to recognize, diagnose and recover from errors.

4.5.1j Nielsen's Heuristic Rule 10: Help and Documentation

Help and documentation is an important aid for new as well as experienced user. Ideally the software should provide all the necessary information to make the help document redundant, but for technical programs this may not always be possible. Since Orbit Offshore is primarily based upon user interaction with spreadsheets there are limited possibilities to implement help functions within the tables themselves, therefore a complementarily help document has been provided. As seen in Figure 4.7, the working process is divided into three major parts: information gathering, screening and detailed analysis. The detailed analysis is further divided into several subparts which have to be carried out to complete the RBI analysis. In every stage and sub-stage of the working process, there is an item called "Guidance", which is a shortcut that automatically opens the help document. The thought behind it is that the user should be able to look up information for each of the specific steps. Instead, it has been found that all of the shortcuts lead to the front page of the help document, and the user therefore must manually find the information relevant for each of the steps. The idea is good, and if properly implemented would help the user find specific information about the task he is currently working on.

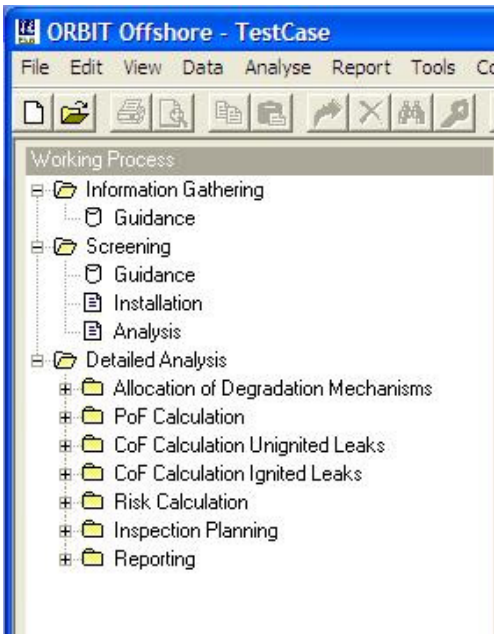


Figure 4.7. Overview of the RBI analysis process

The help document contains an extensive guide on how to carry out the different stages of the overall working process. For each specific stage the document highlights the tables that have to be used, as well as what actions need to be carried out before the user can move on to the next step in the working process. A helpful feature is that all the tables used in the software are reproduced in the help file together with an explanation of their use and the input that is required. Therefore, if a user is uncertain of the information that is needed in order to carry

out each of the calculations he can always refer to the help document to resolve the problem. The help document also contains an explanation of the functions that are available and the location where they can be found in the menu. Additionally, since working with the tables is a significant part of the software, there is a guide on working efficiently with the tables.

Overall, the guidance document in Orbit Offshore is found to be a good aid for both experienced and non-experienced users. For a non-experienced user it is possible to use the document as a step-by-step guide in order to conduct a full analysis, while an experienced user should not have any problem in finding answers to the problems through the use of the search function.

4.5.2 Usability of Orbit Offshore

Earlier in the thesis, usability has been defined as a quality attribute of a software application. Nielsen states that usability consists of five attributes: learnability, memorability, errors, efficiency and satisfaction. To sum up the heuristics evaluation, the five usability attributes will be discussed in light of the evaluation findings, together with the previously mentioned organizational requirements and the defined user. This will be a way of summarizing the findings and concluding on the current degree of usability of Orbit Offshore. The evaluation of the results is a qualitative evaluation based upon the findings and the author's subjective opinion.

4.5.2a Learnability

Learnability is a measure of how easy the software application is to learn, and how easy it is for the users to carry out a basic task the first time they use it (Nielsen, 2005). The first time users of the current Orbit Offshore will need training before they are capable of solving tasks efficiently in the software; this is mainly because of the highly technical discipline of RBI which it is based upon. So, to able to start working with the software, the user must be familiar with the RBI methodology before he or she is capable of carrying out software tasks.

The software has a layout which guides the user through the different stages of the RBI analysis, and highlights the tables that would be used at the different stages. If the user is in need of help he can use the help document which is easily accessible, and contains detailed information about the software working process. The menu in Orbit Offshore is also logical and intuitive to use.

Overall, the software has some features that are helpful for a new user who is trying to learn how to use the software efficiently. But there are some drawbacks as well. One of them is that not all actions in the RBI analysis are so easy to carry out, for example, the use QRA input into the RBI analysis. There is another factor which may have a negative impact on learnability, and that is the need for the use of additional software, Access, to carry out the RBI analysis efficiently. From the point of view of the potential new user who was defined earlier, this is software that he or she would not be familiar with which will have a significant effect on the software's learnability. Learning to use both of the programs will take a significantly longer time than just learning to master one of them.

Considering the fact that Orbit Offshore is technical software designed to carry out a specialized task, it is to be expected that new users have to be given some training. It has many good features which should help the user to quickly learn to use the software efficiently, but there are some weaknesses that need to be corrected.

4.5.2b Memorability

Memorability is a measure of how easy it is to recollect the use of the software application. This is important for users who use an application infrequently. Software with high memorability would be appreciated by infrequent users who need to easily regain their proficiency and not learn to use it all over again (Nielsen, 2005).

The nice overview of the working process implemented in Orbit Offshore should make it easy for an infrequent user to recognize the tasks and actions which should be carried out. The same window can be used to navigate between the required tables which should help the user to recollect the overall software working process. The software also contains a detailed and easily accessible help document which should be a good aid to regaining software proficiency quickly.

The aspect which might have the most significant impact on the software's memorability is the need for additional software to carry out the RBI analysis. Use of Access might be the most difficult part to quickly regain proficiency in as it is an application which is seldom used in relation to other task the users might take on. Since Orbit Offshore lacks some functions in order to work efficiently with the data, it will probably take some time to regain the skill level. Additionally, not all tables and input data are easy to utilize; for example, the use of QRA data in the calculations, may also have an impact on the software memorability.

Overall, the software fares fairly well when it comes to helping the user in recollection of its use, mainly due to a clear working process and the detailed help document.

4.5.2c Error(s)

Error(s) is a measure of how many errors the user would commit, the severity of the errors, and how easily the user can recover from them (Nielsen, useit.com, 1). In the evaluation it has been found that the software's error handling needs improvement. There is no feedback to the user about lack of data or on wrong input of data. Fault checking has been left up to the user him/herself, and users have to actively resolve the errors on their own. The verification and evaluation of the results are difficult to carry out as the software does not offer an optimal solution to carry out such activities. The current practice is to let the users verify their own input data and results, and then a colleague does verification. This makes such an analysis time-consuming. In general, there would not be the need for such extensive evaluations if sufficient error handling were available. The software also does not contain "undo" functions and, combined with the fact that all changes are saved when exiting the software, makes the software difficult to recover from unintended changes.

Based upon the above summary, it is safe to say that the software application's error handling could have been better.

4.5.2d Efficiency

Efficiency is a measure of how quickly the user can carry out the intended task using the software application. Once the user has learned to use the software, it should allow the user to carry out the task in a time-efficient manner (Nielsen, 2005).

The software needs to be improved when it comes to providing an efficient working process for RBI analysis. The first aspect that has significant impact on the software efficiency is the

time required to carry out the screening by manual input, because manual input of data in the detailed analysis is difficult and time-consuming.

Secondly, the software has limited possibilities for tailoring the reports to client specifications. Thirdly, the users require Access to work with data transfer and organization of tables. So, to work efficiently with Orbit offshore, an additional software application has to be used in all steps of the analysis, thereby lowering the overall efficiency of the software.

Based on the above examples, it can be stated that the software needs to be updated to fulfil the DNV requirement to allow for the most efficient working process.

4.5.2e Satisfaction

Satisfaction is a measure of how pleasant the software is to use. The software application should solve the given task in a way that the user finds pleasing and satisfactory (Nielsen, 2005).

The lack of undo functions and the fact that mistakes made while working might not be picked up by the software's error handling makes it inconvenient to use. The need to continuously evaluate the correctness of actions and the RBI analysis results to ensure that they are fault free has a significant impact on user satisfaction. The need for additional software, in this case Microsoft Access, to be able to carry out the RBI analysis efficiently also lowers the satisfaction.

User satisfaction is a highly subjective measure, but from the author's point of view the current Orbit Offshore scores low on user satisfaction. The main problem is that the program is not optimized for conducting RBI analysis as it is done today. Users describe the use of the software as a challenging and repetitive task of inputting data rather than an efficient tool which allows them to use their expert knowledge to carry out the RBI analysis. From the above discussion, it can be stated that overall user satisfaction is low.

4.5.2f Level of Usability of the Current Version of Orbit Offshore

Based upon the above usability summary, where each of the five usability quality measures has been discussed, the conclusion must be that the current version of Orbit Offshore needs to be improved in terms of software usability.

4.6 Step 5: Recommendations based upon Findings from the Usability Evaluation

The extensive usability study of the existing software has highlighted the deficiencies in the software. During the course of study a number of ideas have been generated that may help to improve the usability of the software. This section shall deal with some specific suggestions and recommendations that may help to improve the usability of the next version of the software.

- The overview of the working process in Orbit Offshore is one of the good features of the software and is a great help to users. In future developments this feature should be improved further. This may be done by introducing an even more detailed overview of the working process, where all the steps and actions that are required to be carried out are included.
- The working process should be made as intuitive as possible where each of the steps of the RBI analysis is carried out in a logical sequence. An idea could be for the user to define the project at startup and select the information that is available for the analysis. The software could then be based upon the initial information provided by the user and adapt the working process to include the necessary steps. For example, if the user checks off an option that data from a previously conducted QRA is available, then the application should prompt the user for this data when it is needed. The software should prompt for the data in an order which is intuitive for the user, and that will be most beneficial for the overall RBI working process.
- One of the deficiencies with the current software application is the possibility of direct import of data from other applications or previous analysis. In future developments it should be possible to import data directly from the client's inspection management software or such like. A survey could be carried out to identify the software the clients are currently using and the way the information from such programs can be imported into the RBI software application. Import of data should be conducted in an accurate and efficient way and if possible without user interaction. The imported data could be used directly into screening and from there to the subsequent detail analysis.
- When the imported data are available within the software, there should be functions and actions available to allow for the user to work efficiently and effectively with the data. Examples of such functions are (a) advanced filtering in order to apply actions to multiple components at the same time, (b) buttons and hotkeys for frequently used actions, and (c) built-in unit conversion tools.
- The software should focus on the information that is necessary to carry out the calculations, and the data that will increase the accuracy of the calculation results. This would make it easier to carry out the analysis as it would make it obvious to the user which is the required information and to disregard the unavailable data if it is not required. If such an approach is taken, it should be possible for the user to evaluate the benefits of adding more data to improve the accuracy of the calculated results.
- Philosophically Orbit Offshore is *not* a "push button" software application where the data is imported and the necessary calculations are carried out without any user interaction or evaluation. Instead, it is a tool to aid engineers to carry out tasks efficiently, but after careful evaluation of all stages of the analysis. The software should provide suitable solutions to allow for such evaluations, and make it possible for the user to take advantage

of his expert knowledge and judgment in relation to the analysis. Hence, it must be possible for the user to easily evaluate specific part results and to allow for manual manipulations. This would allow for the user to do a more thorough evaluation of the parts which are identified to have high risk after calculations, which might require a more accurate end result.

- As the end result presented for the client is the report, it should conform to client expectations and requirements. The software should allow for easy adjustment of report content and layout to ensure that the report can be tailored to meet any client specification.
- An RBI analysis consists of many steps and therefore it can be difficult for the user to remember what tasks have been carried out. The software should make it easy for the user to obtain the status about the progress of the overall analysis, and to identify which task and actions have been carried out. If it is difficult to implement automatic features for the status update of the different steps, then a status button should be incorporated to allow manual input of the status.
- The software should also document and save the status of the project at the end of each session. When the project is opened the next time the user should be informed about the progress of the project and the last step he was working on. Such a feature would improve efficiency and software satisfaction as the user does not have to look for or remember where he left off.
- When working with such a data-intensive analysis as RBI, there should be advanced fault diagnostics features to allow the user to recognize errors and to aid the user to carry out the necessary corrective measures. An important focus for future developments is how to implement such error prevention measures to highlight important missing or incorrect data. The error handling abilities of the software should be one of the main focus areas, as undiscovered errors may have a negative impact on the reliability of the end results.
- The possibility to undo actions should be available as this may help the user recover from incorrect actions while working with the software application. This should also be available from a consistency standpoint, as it is frequently available in other software applications and the user might be accustomed to the use of the function.
- The current software already has a good help document, but the future software applications should have more advanced help functions available to guide and help the user to establish the right input or action. Throughout the working process, the help document and information about the current task should be easily accessible for the user. If detailed information and description of the working process is easily accessible, it would help the user to learn faster and to resolve problems more quickly. A detailed and well structured help document may also reduce the resources needed for software training.
- The current software application is a stand-alone application but for future development it would be beneficial to develop a software application which can exchange information with a central database. By doing so it is possible to implement a range of new helpful features.
- This would allow for project files to be stored centrally, such that multiple users can access the same project should that be necessary. Such network communication would also allow for update of information stored in the user application. This can be used, for example, to update pre-defined background tables or degradation models used as input in the calculations.

- There should also be provision for storing project-specific data for future use. For example, if special materials are used in the RBI analysis for an installation, such data may be stored in the central database and retrieved for another related analysis. Similarly, there should also be provision for storing documenting/reporting-specific layouts and requirements as pre-defined templates based upon company requirements.
- Network-based application can open new possibilities, like the possibility for multiple users located in geographically distant locations to work simultaneously on different parts of the same RBI analysis. This feature should come with a "change log" for the users to track the changes made.
- There is a lot of information required to carry out the RBI analysis and such information is gathered from many different sources, like layout drawings and P&IDs. It would be beneficial if such information could be stored either in Orbit Offshore project files or in a central database for further use. If possible, parts and system numbers from CAD-drawings could be linked to the respective parts and systems in the specific Orbit Offshore project file, allowing for even easier user access to installation layout information.

The list of suggestions mentioned is an incomplete list of the features that need to be considered while developing the new version. Hopefully, incorporation of these suggestions along with those from a more extensive usability study will have a positive impact on the usability of future RBI software applications.

4.7 Step 6: Identification of Relevant Design Recommendations from Standards and Publications

When designing a new software application, it is important to take advantage of design recommendations available in different standards and publications as these can provide important input to the development and design process. Instead of listing design recommendations from standards which are easily accessible for software developers, a literature survey has been carried out to find design recommendations for technical software. No design recommendations for RBI software were found, but an article providing such recommendations for condition based monitoring (CBM) software was found. Condition based monitoring software is used to monitor and evaluate machinery health predicatively (Higgs et al., 2006).

In an attempt to establish some guidelines and design rules for CBM software applications, Higgs et al. (2006) conducted a survey in which CBM solution providers and CBM practitioners were asked to contribute. Based on the result of the survey, Higgs et al. presented a list with a total of 40 rules and guidelines for the design and development of CBM software applications (see appendix 4 and 5). While many of these are exclusive for such applications, there are some recommendations that can be applied to software applications in general. Based upon all the suggestions presented by Higgs et al., the relevant points will be presented below together with how these would apply to RBI software.

- RBI - as well as all other software applications - should be designed and developed by using quality programming techniques and standards. It is important that the choice of programming and platform makes the application capable of conducting the intended task. The current Orbit Offshore is a good example of why this is important. Today's RBI analysis has become so data-intensive that the current software is starting to struggle with the magnitude of information. Future software developments should, therefore, take into

consideration not only the current requirements but also the requirements that may arise in the future.

- When designing the RBI software application, the RBI methodology should be taken into consideration. This is important in order to ensure task focus and ultimately a high degree of usability. The usability study should take advantage of user experiences and recommendations when developing the software; this would make it easier to develop software that complies with the end user's requirements and needs. In other words, the application should be designed with the functionality necessary for an efficient RBI analysis working process.
- Maintenance and upgrades should be considered at the design and development stages and provisions made to avoid major rework and changes in graphical user interface. If such considerations are not taken into account, such updates may have a significant impact on software usability. If the product is to be sold commercially, after-sales support must be evaluated as an input to the usability effort. Resources invested in usability efforts in the development process will probably reduce the resources needed to provide after-sales support. So, a good idea is to find a middle ground between these two software aspects.
- When designing the software, opportunities and possibilities made possible by the use of network-based information exchange must be considered. The possibilities for integration with client software must also be considered. For example, RBI requires and generates an extensive amount of data; hence, there should be possibilities to import data and export results directly into the clients' inspection planning or enterprise resource planning software (ERP).
- As with all software, GUI design rules should be followed in the RBI software design phase. Use standards and proven guidelines when designing the graphical user interface to ensure a high degree of usability.
- The screen structure and the information classification must allow for the most beneficial presentation of information. Easy navigation techniques should be provided to allow the user to easily move between the windows and dialogues needed to carry out RBI analysis and to evaluate the subsequent results. Functions and actions must be made easily accessible as they are needed to carry out the different tasks.
- When appropriate, the RBI software structure must be designed to incorporate the use of windows and tables. Tables are beneficial when a large amount of data is to be processed or presented for the user, as is the case for RBI software. The possibility to show multiple windows at the same time allows for presentation of different parts of related information, and may give the user a better overview. An example of this could be the possibility to show navigation windows, data windows and graphical presentations of results at the same time. If possible, graphical and 3-D drawings must be used to illustrate the components and systems in question as this can help in understanding and taking right decisions.
- The choice of text size and font type should be considered and found appropriate for use in the software. An evaluation of other programs that the intended user would utilize could be beneficial to avoid inconsistencies. The icons used must be self-explanatory and familiar for the intended user, and as far as possible platform conventions and standards must be followed.
- The language used in the software has to be understandable for the intended user, and should give him/her a clear understanding of the kind of input needed. Therefore,

technical terms that may not be familiar to the intended user should be avoided. For Orbit Offshore it would be natural that the language and technical terms be aligned with those used in the RP-G101 as this is a document likely to be used for employee training and introduction to RBI methodology.

- An easy way of helping the user with information input is the use of dropdown menus. Especially for users who do not use the software on a daily basis, a list of choices may help them conduct the analysis as they may recognize the right input when they are presented with the possible selections. On the other hand, extensive use of such menus should be avoided as it may impair the task efficiency for an experienced user, which was found to be the case in the current Orbit Offshore.
- Security is important; therefore username and password authentication of the user should be implemented. This is something which applies to all software where information should not be readily available for all personnel. While access to the information may not be the critical scenario, an unauthorized intended or unintended change of the information or the results is. Therefore, to maintain the integrity of the information and results, the use of the software should be restricted to the intended users, especially if DNV wants to make future software network-based.
- Software which is used to process large amounts of data should have automatic fault diagnostics. For RBI, the previous statement is certainly the case and automated fault diagnostics should be available to prevent wrong, or lack of, input.
- If the software has different intended users who will utilize the program to carry out different tasks, it could be beneficial to have different user interfaces between the user groups. The user interfaces should then be tailored to emphasize the task which is to be carried out by the intended user group. For RBI software this could, for example, be that the client was a separate user group which only had access to results and reports. Another possibility could be a software interface which allowed for the client to upload available information to be used in an analysis, as for example drawings and the line list.

The above-mentioned recommendations are an example of identified guidelines and design rules which have been applied to a specific software application, in this case an RBI analysis tool. From the above, it can be stated that such recommendations may provide practical and important input to a software development. Due to the latter, the use of standards and publications should always be considered as a valuable input to the software design and development.

5. Discussion and Conclusion

5.1 Introduction

In the following chapter the project discussion and conclusion will be presented. The discussion addresses the adopted approach, and the results in general as the specific results have already been discussed in previous sections.

5.2 Discussion

Usability was initially found to be a concept that was difficult to adopt for a non-software or usability engineer. One of the problems was that usability definitions were defined differently between authors and standards, and the content varied along with it. As previously mentioned, Goodwin (1987) stated that usability too often is defined in abstract terms that give little guidance, which from the author's recent experience must be said to be true.

The most challenging part of the project was to identify the most applicable usability evaluation method as there were many to choose from, but their advantages and drawbacks - as well as how to carry them out - were difficult to identify. In general, the usability effort was difficult to plan as little guidance was found on how to carry out the type of usability study that was intended for this project. As there little guidance was found in the literature, it was decided to develop a framework for the usability study of Orbit Offshore.

The proposed framework was found to be very helpful. The choice of defining a potential new user made it possible to evaluate the software application's usability in relation to what computer skills such a user might have. An example of results identified based upon the evaluation criteria, was the use of Access to work with importation of data and its impact on the software learnability and user satisfaction. While this specific example probably was an obvious one, which likely could be identified without the evaluation criteria, the author would still claim it to be an important input to such a usability evaluation.

The organizational requirements made it possible to classify the software's usability performance in relation to the company's requirements. For example, it was found that the software did not offer the most efficient working process, which was one of the requirements set by the company prior to the software development. Based upon the results related to the organizational requirements, the author would claim that this also was an important input to the usability evaluation. Also included in the framework was a survey of publications and standards. This provided important input to the recommendations for future developments. By identifying relevant design rules and guidelines from publications, it was possible to adapt these to RBI software and present suggestions for future software developments.

The framework made it easier to plan the activities and identify the resources needed to complete them. Overall, the proposed framework was found to include the necessary activities in a logical sequence, which made it a suitable approach for the usability study of Orbit Offshore.

The author will claim that the choice of using a usability inspection method to evaluate Orbit Offshore was the right decision. The use of Nielsen heuristics provided good support for an evaluator who was not educated within the field of usability or software engineering. But not all the rules were as easy for a first-time usability evaluator to apply. For example, rule four of Nielsen's heuristics states that the software should follow platform conventions; for a non-usability or software engineer such problems are difficult to identify due to lack of formal education within the field of software engineering. On the other hand, such problems were anticipated and were one of the reasons why the usability effort was mainly focused around the software application's technical domain.

It can be debated whether this was a viable method for identifying the software's strengths and weaknesses, as heuristics evaluation is mainly used to identify usability problems. As the chosen method was a subjective method for assessing the software usability, this made it possible for the evaluator to assess which aspects had a positive impact on the usability based on his subjective judgment. Therefore, it can be claimed that if the evaluator is not merely focused on identifying weaknesses, it can be possible to identify software strengths as well. In this project it was probably easier to identify positive aspects than is common for such evaluations. The evaluator was a new user and had to rely upon the software's built-in guidance and help in order to use the software; this made it possible to identify what had a positive influence on the software's usability. These two previously mentioned aspects are important and have a significant impact on usability of a software application. On the other hand, based upon the presented results, it must be stated that it was easier to find usability problems rather than aspects which had a positive impact on usability.

Further, it can also be questioned whether it is beneficial for a non-expert in usability to carry out such an evaluation. However, it can be argued that as the evaluator was not an educated usability or software engineer, that made it easier to focus on the specific usability problems in relation to the technical domain rather than the "traditional" usability problems. As mentioned when discussing the evaluation method adopted in chapter 3.3, evaluators may overlook usability problems if the software is highly domain-dependent (Nielsen n.d. D). In this case the software was closely linked to the RBI methodology, and therefore it would be safe to say that it would be an advantage if the evaluator had knowledge about the software domain. And as there are probably not many usability or software engineers with knowledge about the RBI methodology, it can be stated that the project and the approach of having an evaluator with knowledge of the domain to carry out this evaluation can be defended. Based upon some of the evaluation findings, it would also be probable that some of the findings would not have been picked up by an evaluator who was not familiar with the RBI methodology, which further confirms the validity of the usability approach.

From the usability study several problems were identified, but there were also identified aspects which had a positive impact on the software's usability. Based upon the evaluation results, it was possible to suggest some specific recommendations for future developments, what to avoid and what to implement in future RBI software. From the evaluation and subsequent survey of publications, more than 30 points were presented which should be considered in relation to future software developments. The latter should indicate that the approach at least had some amount of success.

It must be said that these findings were based upon subjective judgment in relation to the ten heuristics, and therefore the findings may indicate problems in relation to the specific user group of the evaluator. As the evaluator was not an experienced user of the software

application, some of the findings are likely to be usability problems which might only apply to a new or novice user. Therefore, it could be beneficial to carry out a usability study to identify specifically what problems these experienced users might have with the software. Such usability studies may be carried out by using the proposed framework or by adapting it to the specific goal of the usability effort.

From the above discussion it can be claimed that the application of the proposed framework and the completed usability study was a valid approach for the evaluation of Orbit Offshore.

5.3 Conclusion

The goal was to get an in-depth understanding of the RBI methodology and, based upon that knowledge, carry out a usability study to identify domain-specific usability problems. From the previous discussion it can be stated that the application of the usability evaluation method and usability study as a whole was successful in identifying such usability problems.

On the other hand, even if it was possible to identify a range of usability problems it must be emphasized that this usability study will not be a substitute for a complete usability study carried out by multiple evaluators. But the goal of the project was not to identify all usability problems, but rather indentify its strengths and weaknesses which the project to a large extent accomplished.

The proposed framework was also found to be suited for carrying out the usability study of Orbit Offshore. So if DNV desires to carry out an additional usability study of the current software or to carry out usability studies of future software, the framework can be applied directly or tailored to reflect the specific goal of the usability activity.

Based upon the above discussion and the results presented, the conclusion must be that the aim of project has been accomplished.

References

American petroleum institute, 2002, API Recommended practice 580 – Risk based inspection First edition, Washington: API

Det Norske Veritas, 2009, Recommended practice RP-G101 - Risk based inspection of offshore topsides mechanical equipment, Høvik: DNV

Faulkner Xristine, 2000, Usability Engineering
Palgrave Macmillan
ISBN-10: 0333773217

Folmer, E. & Bosch J., 2004. Architecting for usability: a survey [Online]
The Journal of systems and software, 70, pp. 61-78.

Goodwin, Nancy, 1987, Usability and Functionality [Online]
Communication of ACM, volume 30, number 3, pp.229-233
Available at: http://mis.njit.edu/ullman/cis465/Articles/Goodwin.pdf
[Accessed 17 March 2005].

Higgs, P.A., Parkin, R. & Jackson, M., 2006. Guidelines and rules for condition based maintenance software GUI design.
International Journal of COMADEM, 9 (4), pp. 14-22

Lauesen, Søren 1997, Usability Engineering in Industrial Practice
Published in Howard et al. (eds.): Human-Computer Interaction
Interact'97, Chapman & Hall, 1997, pp. 15-22.

Nielsen Jakob, 2005, Heuristics 10 rules [Online]
Available at: http://www.useit.com/papers/heuristic/heuristic_list.html
[Accessed 10 March 2009]

Nielsen Jakob, n.d. A, Usability 101 [Online]
Available at: http://www.useit.com/alertbox/20030825.html
[Accessed 2 March 2009]

Nielsen Jakob, n.d. B, Useit.com, iterative design [Online]
Available at: http://www.useit.com/papers/iterative_design/
[Accessed 10 March 2009]

Nielsen Jakob, n.d. C, Heuristic Evaluation [Online]
Available at: http://www.useit.com/papers/heuristic/
[Accessed 10 March 2009]

Nielsen Jakob, n.d. D, Characteristics of Usability Problems Found by Heuristic Evaluation [Online]
Available at: http://www.useit.com/papers/heuristic/usability_problems.html
[Accessed 10 March 2009]

Nielsen Jakob, n.d. E , How to Conduct a Heuristic Evaluation [Online]
Available at: http://www.useit.com/papers/heuristic/heuristic_evaluation.html
[Accessed 10 March 2009]

Norsk Standard, 1998, NS-EN ISO 9241-11 *Ergonomic requirements for office work with visual display terminals (VDTs)* Part 11: Guidance on usability (ISO 9241-11:1998)

Schaffer, Eric, 2004, *Institutionalization of Usability: A Step-by-Step Guide*
Addison Wesley Longman Publishing Co., Inc. Redwood City, CA, USA
ISBN:032117934X

Usability.gov, n.d. What is user-centered design? [Online]
Available at: http://www.usability.gov/basics/usercntrd.html
[Accessed 29 March 2009].

Appendix

Appendix 1:
Overview over the Orbit Offshore working process

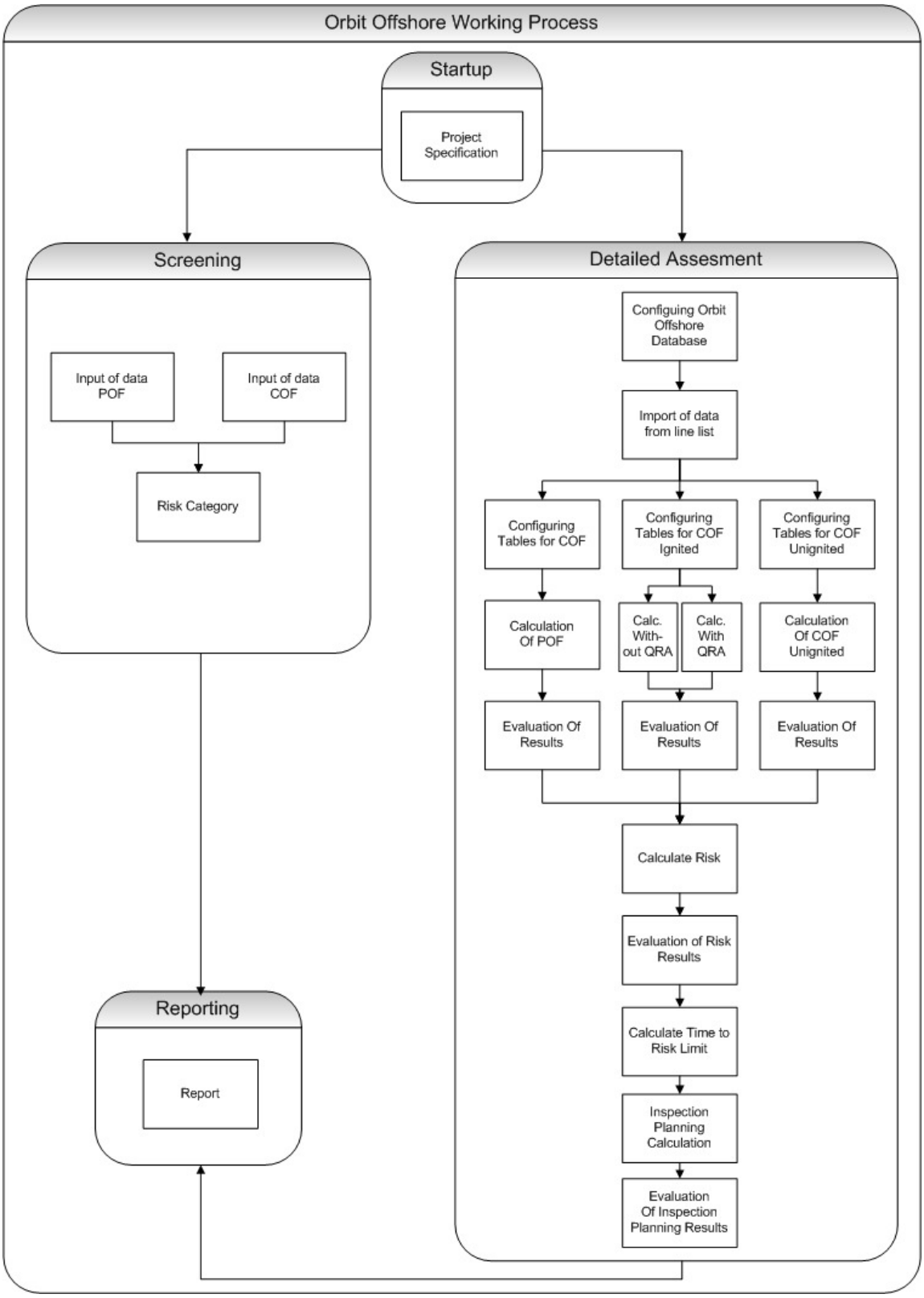
Appendix 2:
Overview over the proposed RP-G101 working process

Appendix 3:
Overview of the Schaffer method

Appendix 4:
Overview over Higgs et al. rules and guidelines Part 1

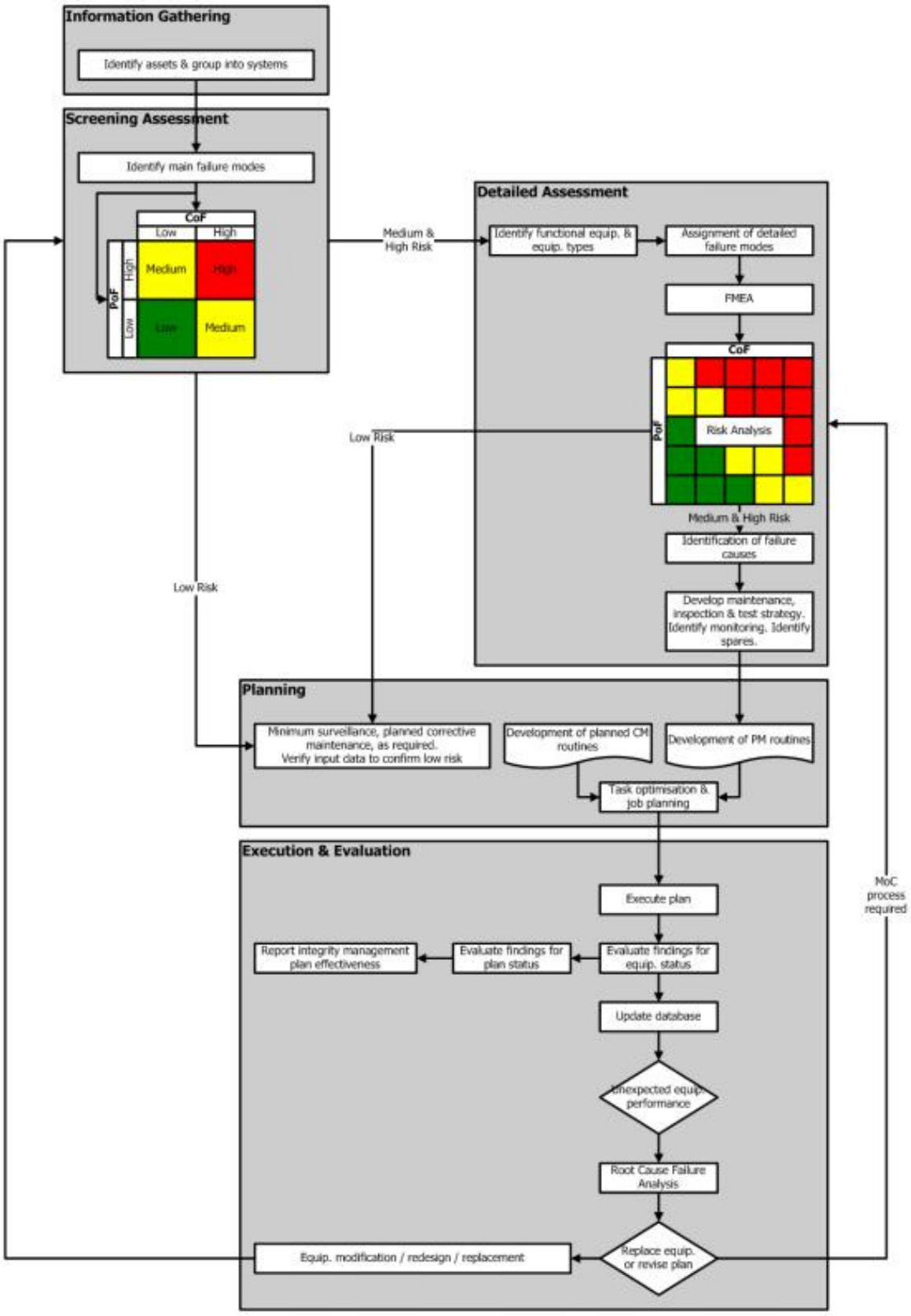
Appendix 5:
Overview over Higgs et al. rules and guidelines Part 2

Appendix 1



Overview over the Orbit Offshore working process

Appendix 2



Overview over the proposed RP-G101 working process

Appendix 3

Overview of the Schaffer method

Source: Human Factor
http://www.humanfactors.com/downloads/SWMchart.pdf

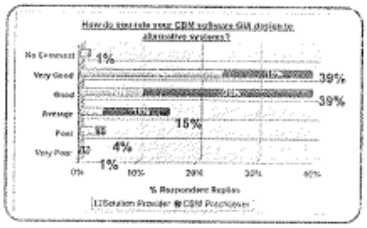
Appendix 4

Two of the design considerations listed in Table 5, items 5 and 6, are the same or associated with items 6 and 7, found in Table 4. Both considerations concern software tailor-ability issues.

Designing CBM software to be, 'out of box user friendly', received the highest number of additional comments. Software considered out of box user friendly would be assumed to adhere to many GUI design rules. CBM software designed to integrate with other computer applications and operate on more than one operating system, received two additional comments. Software capable of integrating with other computer applications, can be assumed to incorporate file and data transfer and sharing capabilities. Software capable of operating on more than one operating system would be assumed to be programmed in a multi platform operate-able language. Considerations 2 and 3, Table 5, do not concern CBM GUI design as such, but rather the actual sales and marketing effort put in by the CBM solution provider, and the after sales support and maintenance offered, following the sale. Broadly speaking, both of these considerations affect the saleability and success of products universally.

3.9. How do respondents rate their existing CBM software's/s' GUI.

An equal majority of respondents (39%) rate their CBM software's GUI design as being good or very good in comparison to alternative systems on the market. Only a small percentage of respondents (5%) considered their CBM GUIs to be poor or very poor compared to other software. Solution providers have a greater tendency to rate their CBM GUIs designs, higher than CBM practitioners. Such a result must be anticipated, because rarely does a sales person (e.g., a CBM solution provider) consider their own merchandise negatively, compared to a competitors merchandise.



3.10. Comparison of survey results

Although not presented graphically or numerically for all the results, opinions from CBM solution providers and CBM practitioners did not vary to a great extent for the entire survey. A noticeable few exceptions included:

- More CBM solution providers (74%) place a high or very high level of importance on the use of colour coding to reflect machinery condition, compared to 54% of CBM practitioners.
- More CBM solution providers (72%) believe tailoring the design of CBM GUIs to customer requirements, has a high or very high level of influence over CBM software design decisions, compared to 56% of CBM practitioners.
- More CBM solution providers (77%) agree or strongly agree, the reliability and quality of programming affects the saleability and success of CBM software, compared to 58% of CBM practitioners.

The lower percentage vote from CBM practitioners for the three bulleted cases above, is suggested as being attributable to a higher level of ignorance amongst this group, towards the complexities of GUI design and Human Computer Interaction.

Sections 3.6., 3.7. and 3.8., provided a measure of the importance and value several CBM software design considerations, issues relevant to GUI design rules, and various CBM system software control features, have on CBM software design and CBM system success. Each result is grouped together and presented below, categorised highest to lowest according to its perceived level of importance or value:

3.10.1. Proposed CBM software design and development rules and guidelines

Considered to be very high in importance or value

1. Design CBM software with a selection of data analysis capabilities, suited to the CBM technique being employed.
2. Design CBM software with correct interpretability of CBM information, leading to accurate maintenance decisions in mind.
3. Incorporate a data archiving feature as an option in CBM software.
4. Design CBM software with look, feel and interpretability of reports (refer to item 5 below) in mind.
5. Design CBM software to incorporate maintenance and malfunction report generation capabilities, so CBM information can be presented using performance reports.
6. Design CBM software to present condition parameters graphically, i.e. using charts and graphs.

Overview over Higgs et al. rules and guidelines Part 1

Source: Higgs et al

Higgs, P.A., Parkin, R. & Jackson, M., 2006. Guidelines and rules for condition based maintenance software GUI design. *International Journal of COMADEM, 9 (4), pp. 14-22*

Appendix 5

7. Design CBM software to incorporate password and user name authentication.

8. Design CBM software using reliable, quality programming techniques and standards.

Considered to be high in importance or value

9. Consider screen structure and information classification whilst designing CBM software GUIs.

10. Design CBM software to incorporate visually appealing GUI screens.

11. Use appropriate language that is legible to the intended audience in the CBM software GUI.

12. Design CBM software GUIs to incorporate drop down menus.

13. Provide easy to use navigation techniques in the CBM software's GUIs.

14. Design the structure of CBM software to incorporate windows.

15. Take into account the CBM technique being employed when designing the CBM software.

16. Design the CBM software to satisfy the intended audiences requirements.

17. Design the CBM software with the intended functionality of the CBM system in mind.

18. Take into account networking and integration opportunities and requirements whilst designing the CBM software.

19. Anticipate having to tailor CBM software to users/ customers requirements.

20. Design CBM software GUIs to make use of tables where appropriate.

21. Design CBM software GUIs to incorporate colour coding to reflect machinery condition.

22. Provide users with the ability to control the appearance and functionality of CBM GUI control features.

Considered to be average in importance or value

23. Whilst designing CBM software follow recognised GUI design rules.

24. Take into account environmental issues, i.e. location of displays / monitors.

25. Use text size and font type appropriately in CBM software GUIs.

26. Use icons and icon design appropriately in CBM software GUIs.

Additional issues suggested by respondents

27. Provide software control features enabling data to be exported or imported between a CBM software application and other software applications.

28. Design CBM software GUIs to be user friendly logical and easy to operate, e.g. out of box user friendly, easy to use navigation techniques.

29. Design an urgent method of communicating machinery defects and deviations from acceptable trends into CBM software GUIs and other media.

30. Design CBM software to automatically distinguish and represent machinery condition according to the criticality before anticipated failure. Take into account the level of importance the machine being monitored represents if it fails and is unable to function.

31. Design the CBM software to receive and process data from multiple different types of CBM techniques, and multiple machines.

32. Design CBM software to enable users to access CBM information through Internet web pages.

33. Provide automated data screening and automated fault diagnostics into CBM software.

34. Consider the marketing and sales strategies anticipated at the design and development stages of a CBM software application.

35. Consider the after sales support, including maintenance, and software upgrades, at the design and development stages of a CBM software application.

36. Incorporate an error protection feature for back stepping through user actions within the CBM software's GUIs.

37. Use 3D representation of machinery and machinery condition in CBM software GUIs.

38. Provide users with an ability to make a selection from a group of GUIs. This selection might include user interfaces designed specifically for a particular operation or designed for a specific user group.

39. Incorporate a feature within the CBM software for automatic machinery identification verification.

40. Design the CBM system to operate through a single central database.

A certain degree of association and repetition can be detected amongst the 40 items above. This occurs because some items form part or need to be incorporated in

Overview over Higgs et al. rules and guidelines Part 2

Source: Higgs et al.
Higgs, P.A., Parkin, R. & Jackson, M., 2006. Guidelines and rules for condition based maintenance software GUI design. International Journal of COMADEM, 9 (4), pp. 14-22