VIRTUAL WOCS

Merging industrial control-systems with a virtual 3D-environment

HØGSKOLEN i Buskerud

nebb

Operator training in a virtual 3D-environment

Leif Henning Larsen

Project leader, implementation

Dejan Vukobratovic

Analysis, risk analysis, web

Torjus Engell

Test, economy, web

Jan Hansen

Requirements, design, documentation

vWOCS

FINAL DOCUMENT

V1.0 27.05.2011

Leif Henning Larsen

Jan Hansen

Dejan Vukobratovic

Torjus Engell

External supervisor

Internal supervisor

Table of Contents

Table of figures

Table of tables

1 Introduction

1.1 SHORT DESCRIPTION OF THE PROJECT

This project is the work of following students at Department of Technology, Høgskolen i Buskerud (HiBu University College), Kongsberg:

- Leif H. Larsen
- Torjus Engell
- Jan Hansen
- Dejan Vukobratovic

"The main purpose of the project is to enhance the Workover Control System (WOCS) by integrating it with a VR (Virtual Reality) application called Source. Source is a game engine produced by Valve, a corporation that made many astonishing game titles such as Half-Life and Portal. For this project, we will modify the Source engine to fit our needs for the project requirements.

Personnel can use Virtual WOCS for training purposes. By simulating a ship or an oil rig, the personnel no longer have to travel far out to deep waters. This also includes emergency scenarios, such as equipment on fire."

- Virtual WOCS blog

FIGURE 1 - CONNECTING THE TWO SYSTEMS

Our job is to connect two pieces of software with each other using an interface that we will build. This will allow them to communicate with each other, so that actions in Source will simulate the same action in System Platform, and vice versa, in real time. Figure 1 explains this project at its simplest form.

1.2 DEFINITIONS, ACRONYMS AND ABBREVIATION

This is a short list of definitions, acronyms and abbreviations. Please refer to the attachment "Definitions, Acronyms and Abbreviations" for more information.

WOCS, Christmas tree, VR, EyeSim, ROV, API, UML, HiBu, Use case diagram, Javadoc, Doxygen, Unified Process, Scrum, UML, VR, Dropbox, HPU, System Platform, Artificial Intelligence, HMI, Stack, Source, Valve, HPU, Socket, Server, Client, String, Thread, Port, Listen, Bind, Array, UML, Use case diagram, Galaxy, OPC, Unity 3D, Crytek

2 Vision Document

2.1 INTRODUCTION

2.1.1 PURPOSE OF THE DOCUMENT

This document is meant to describe a project idea. It will cover the following statements:

- Who has prepared the proposal for the project
- The purpose and the goal of the project
- If the project can be solved in several parts
- A description of our understanding of the project

The project description has to be short and gladly contain sketches and drawings which promote the general understanding of the verbal presentation. The purpose is to provide the reader with a fundamental understanding of the system that will be built. This document will be the basis for a process of making the project approved. It will also describe the fundamental questions "why" and "what" in connection with the project, and be the basis for everything that concerns the progress of the project.

2.1.2 AUTHOR

Dejan Vukobratovic

2.1.3 RESPONSIBLE

Dejan Vukobratovic

2.2 PRODUCT PROPERTIES

Nebb Engineering is in need of a system that simulates their existing WOCS system in 3D, and makes it possible to control elements in WOCS using VR equipment.

Nebb Engineering is a company which works within the areas of process, energy and environment to offer concept studies and engineering services, including complete electrical and automation systems. Nebb Engineering is also involved in subsea oil installations and safety systems for ships and drilling rigs.

As of this day, Nebb uses a system for controlling subsea installations called WOCS (Workover Control System). This makes it possible to control Christmas trees by providing possibilities for opening/closing valves with help of computers. The WOCS system is integrated in a container that is placed on a ship or a drilling rig. Inside the container, there are computers and other equipment necessary for operations.

Nebb wants to simulate this WOCS system in interactive 3D, so that operations that are usually carried out offshore can also be done on land with the aid of VR equipment. This makes it possible to train workers without exposing them for dangerous situations. In addition, the system will make it possible to train on handling emergencies, for such as fires etc.

The VR system to be used is called EyeSim, by Invensis. With the use of 3D goggles and glove, one can operate interactive elements, such as valves. It should be possible to bring up details like valve type, pressure, serial number and more.

In addition to the simulation of WOCS in 3D, activities in one of the systems must influence the same activities in the other system, and the other way around of course. This means that the purpose of this project idea is to complete an interface between the two software systems.

2.3 OVERVIEW

2.3.1 SUMMARY

- A virtual system in full 3D.
- Must simulate a working environment on ship/drilling rigs.
- Must provide possibilities for:
  - Operating elements through VR equipment (goggles and glove)
  - Bring up detailed information (valve type, pressure and more)
- Must be able to influence and be influenced by the WOCS system.

2.3.2 CONDITIONS

Development licenses for EyeSim will be obtained when the internal censor (HiBu) has sent an email to Invensys, and when the students have signed a contract. This way, Invensys secures their software and makes sure it will only be used for this project.

2.3.3 EXPENSES

Project costs will be determined by:

- Food service in connection with project presentations and meetings
- 3D equipment (goggles and glove)
- Development licenses from Invensys
- Help and support by Invensys technical team
- Equipment for display of the product (monitors from 3D Perception)
- Shipping of this equipment to the place where the project is being developed
- Costs in connection with food and social activities by schedule
- Number of working hours

2.3.4 POTENTIAL

The result of this project is meant to be a system that provides an interface between WOCS and EyeSim software. Nebb has not been involved with VR before, and this is the first time they wish to try it. We will be able to work with groundbreaking and market leading technology. Project's outcome will be used for training of personnel without exposing them for dangerous situations offshore.

2.3.5 PRIORITY

The project can be split up in several parts:

2.3.5.1 PRIORITY A

Priority A will be to simulate a WOCS container system in full 3D with help of EyeSim. The chosen system functions in WOCS will also be simulated in EyeSim.

2.3.5.2 PRIORITY B

Priority B will be to make the interface between WOCS and EyeSim, so that both systems can influence and be influenced by each other.

2.3.5.3 PRIORITY C

Priority C will be to extend the product (if possible) to cover the underwater systems as well, operated by ROV's.

3 Pre-study report

3.1 PURPOSE OF THIS DOCUMENT

The purpose of this document is that the group members will get a basic knowledge about the assignment, and to create a time schedule for the work we are to do throughout the fall.

3.2 AUTHOR

Leif Henning Larsen

3.3 IN CHARGE OF ACTIVITY

Project group

3.4 ASSIGNMENT

This suggestion of assignment for this project is given by Nebb Engineering AS.

Nebb Engineering AS is a company which has been in the engineering industry a long time. Nebb works within the fields of process, energy and environment and offers concept studies as well as engineering services. To mention a few they offer complete electrical and automated systems. Nebb also works with subsea oil installations and safety systems for ships and drilling rigs.

Nebb are in the need of a system to simulate their existing WOCS in 3D, and which makes it possible to control elements in the WOCS using VR equipment.

As of today Nebb uses a system called WOCS to control and monitoring subsea installations. Using this system they can control Christmas trees where they have the opportunity to open/close valves by using computers. The system is integrated in a container which is placed on ships and drilling rigs. Inside this container there are computers and other equipment which is required to run the installation.

The idea of the assignment is to simulate the WOCS with interactive 3D, so that operations which you usually would to offshore also can be done onshore with VR equipment. This will make it possible to test different installations without using a lot of money and time in real life. You could also use this simulation to train operators onshore, without exposing them to dangerous situations offshore. It could also be used to practice emergency situations, such as fire. In other words, there are a lot of possibilities.

The VR system which is supposed to be used is Invensys EyeSim. By using 3D goggles and some kind of controller (PlayStation controller for instance) one can operate interactive elements, such as valves. This is just what Nebb wants.

In addition to the simulation in full 3D operations in one system should affect the same operations in the original WOCS system. This will also go the other way; operations in the WOCS should affect the same operations in EyeSim. The complete idea of this project is to create an interface between these two systems.

We can write a list of main objectives of the project:

- Implement parts of a WOCS in interactive 3D
- Implement chosen WOCS operations in 3D
- Connect WOCS and EyeSim through an interface, which we will create, so that the systems will affect each other.

This list is the main objectives of the project. We will later split this in smaller, more detailed objectives in the requirement specification.

Our thoughts about the analysis of the project (is it possible to do?) can be found in the chapter "General aspects around the project".

3.5 WORKING METHODS

This is a big project, and it will obviously require a great deal of planning as for how we are to work our way around it. In the following we will evaluate different project methods which are natural to consider for this project.

3.5.1 SPIRAL MODEL

The spiral model combines the waterfall model and prototyping (we will not go into detail about those). This model is thought to be used in big, expensive projects. Working with this model requires thorough risk analysis for each round.

To explain how this model works, we can sum it to the following;

1. Gather system requirements and define these as detailed as possible.
2. Analyze system requirements and risks.
3. Create a temporary design of the system.
4. Create a prototype of the system based on the design.
5. Create a new prototype, using the same points as above.

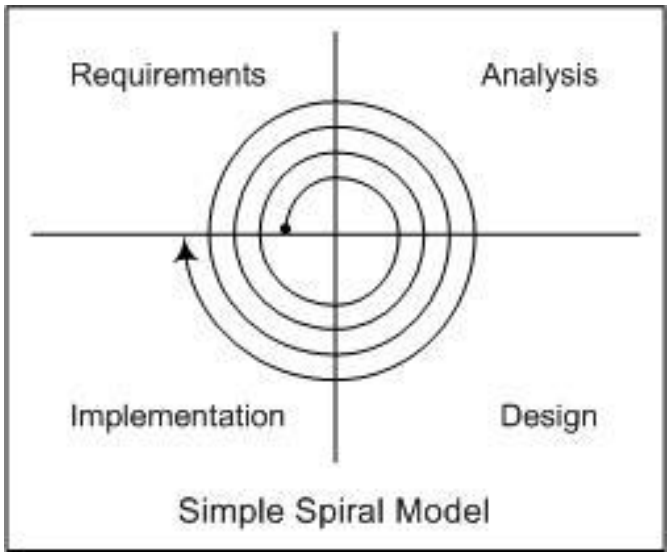

FIGURE 2 - SPIRAL MODEL

3.5.1.1 ADVANTAGES

- Errors are eliminated early
- Requirements are executed by importance, so vital parts of the system are first done.

3.5.1.2 DISADVANTAGES

- Details must be elaborated early
- One has to be very good at risk analysis.

3.5.2 EXTREME PROGRAMMING

Another project model we have considered is called extreme programming. This is a project model which aims to satisfy the customers' requirements/needs. This model has the possibilities to change requirements late in the development process, without causing too much trouble.

Communication and simplicity are emphasized, and one is supposed to start testing from day 1. This gives the customer the possibility to give feedback which can be handled quickly.

There are three phases behind extreme programming. First you go through a planning stage, where you find the customers' requirements, and agree on deadlines. Further on you have a design phase. Everything is kept simple, and there are often restructuring. Optimization is done at last. The last phase is the test phase. Every parts of code need to have relevant tests, and very part of the code must succeed those tests before one can continue.

3.5.2.1 ADVANTAGES

- Start early with development.
- Changes in late stage of development are possible.

3.5.2.2 DISADVANTAGES

- A lot of time can be wasted with restructuring.
- Can incorporate bad software design.

3.5.3 UNIFIED PROCESS

Unified process is an iterative process method, which consists of four phases.

3.5.3.1 INCEPTION

This is the first and shortest phase. Throughout this phase one should be able to outline system requirements, identify risks associated to the project, and consider possible solutions to the project.

3.5.3.2 ELABORATION

This phase aims to find the final way of develop the product. If you use UML use cases-, class- and package diagrams will be developed through this phase. At the end of this phase one are expected to deliver a good plan for the construction phase.

3.5.3.3 CONSTRUCTION

This is the biggest phase in the project. The objectives within this phase are to develop the product based on what you did in the elaboration phase. One will usually have many iterations, where you develop different parts of the product in each iteration.

3.5.3.4 TRANSITION

This is the last phase of the project, and is the phase where you release the product. You will get feedback of the product. Based on this feedback you will consider if you need a lot of iterations to change the product and correct errors.

3.5.3.5 ADVANTAGES

- Structured
- Use-case based iterations

3.5.3.6 DISADVANTAGES

- Can be complicated to change requirements towards the end.

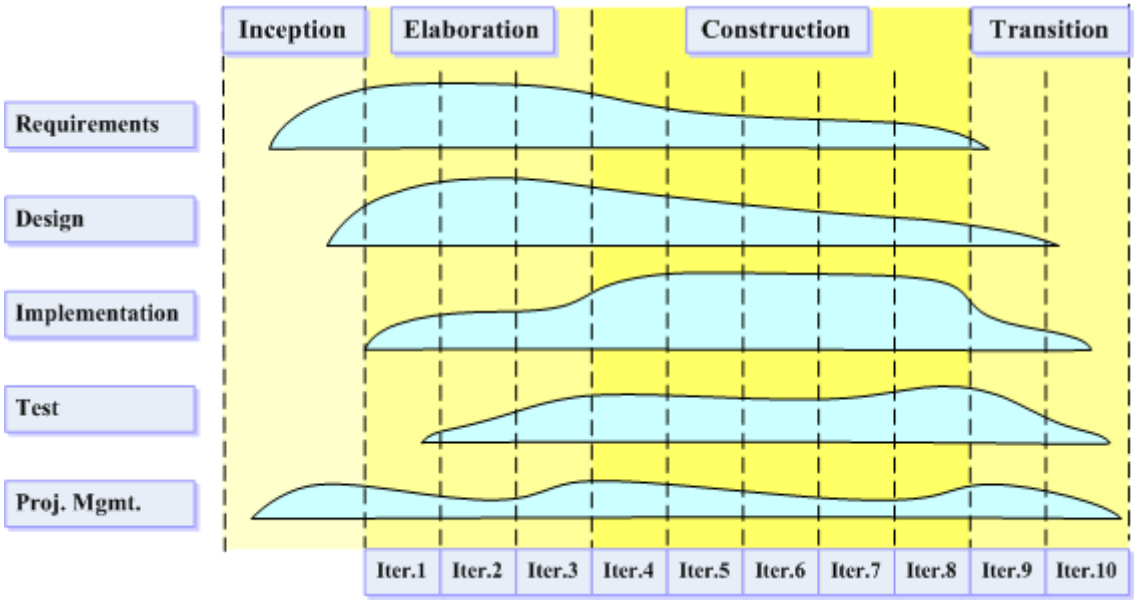

FIGURE 3 - UNIFIED PROCESS MODEL

3.5.4 OUR CHOICE

After some discussion around which method that was best suited for this type of project we have decided to use unified process, with some elements from other working methods. We chose this method because it consists iterative phases. As figure 3 illustrates it is easy to create project activities, and outline the progress from this.

In the elaboration phase we will use UML diagrams, as we have had some experience with this type of planning from school. We feel quite familiar with the software used to create these diagrams, and the possibilities it offers. UML will make it possible to explain the projects technical side on a structured way. We will also get a good start on the construction phase if we use this tool.

To reach the main objectives we will also incorporate ordinary scrum-meetings. This is short meetings at the start of the day to go through what the status of the project is, and what the plans of the day are. These meetings will give us an idea of what needs to be prioritized that day.

3.6 GENERAL ASPECTS AROUND THE PROJECT

3.6.1 ANALYSIS OF THE PROJECT – IS IT POSSIBLE TO SOLVE IT?

Based on the project description, working methods and our own programming experience it is very likely that we can solve it. However, there are several uncertainties.

Firstly the size of a WOCS system can be problematic. Nebb operates with huge installations, which could affect the project. However, after a meeting with Nebb in start of October, we came to an agreement that Nebb will select some operations they would like us to work with. This is good since we can determine an approximate size of the project.

Another uncertainty is the technical. We still do not know which programming language EyeSim is developed in, and WOCS is script based. If we are to get stuck, we would have to call Invensys support team to get help with EyeSim. This could quickly use a huge amount of time if a bigger problem were to occur.

More uncertainties will probably occur when we develop the requirements, but those two points above can affect the project plan quite a lot.

We do feel that it is possible to solve a project of this size. When we get the proper licenses we can take a look at the technical side, and we can then determine requirements which are realistic.

As a project group we have great understanding that the project has a big scope, and that it is quite exiting. We also understand that there are a lot that can go wrong, and that this will require much more work than we are aware of.

Now that we have started to work with the project we would not consider changing project. We feel that Nebb are good as employers, and we hope that we can get a close and god cooperation with them.

3.6.2 3D MODELS

We are supposed to simulate a work environment from oil rigs/ships in a virtual system in 3D. Since it does not exist for the system we will need different 3D models to make it realistic. It takes a great amount of time to create these models by professionals, so it is unrealistic to create these ourselves. It will probably not be realistic to purchase models, since this can be quite expensive. Nebb will therefore need to give us the models we need.

3.7 TIME SCHEDULE

By the fall of 2010 each of the four project group members should use approximately 100 hours each. This will be distributed on requirements specification, test specification, maybe user manual (API) and project plan. The fall of 2010 will be the start of the project, and it is natural to let it be iteration 1.

3.7.1 REQUIREMENTS SPECIFICATION

This pre-study report is to be done within the first week of October. When it is done we will meet the employer to start to find requirements. The requirements specification is the next task on the list. It is a relatively big activity, which will require a lot of work in the start. Later during the fall we will focus more on the project plan.

We will try to reveal as many requirements as possible, as correctly as possible during the fall, so we will have monthly meetings with the employer to make sure we are on the correct track.

3.7.2 TEST SPECIFICATION

We will start to write the test specification a little after the requirements specification. This will be a document which describes which test strategies we will use and how we are to test the different requirements.

This phase will be executed in parallel with the requirements, as the communication between those is important.

3.7.3 USER MANUAL

As of today we are not sure if a user manual is required at this stage. The alternative is to write this document at the spring. In case we need to write it now, we will do this in the end of November.

3.7.4 PROJECT PLAN

The project plan will obviously be crucial for the further work. Therefore it is important that we do a good job writing this. When the requirements starts to come in place we need to start to plan iterations and activities, so we should start with the project plan in November.

3.7.5 FIRST PRESENTATION

The first presentation will be held within the first week of January 2011. This is what we work towards the entire fall, and all the mentioned products will be delivered at this presentation. The presentation itself will deal with project description, and some about the requirements/tests and project plan.

When we have finished the exams in December we should start to prepare for the presentation. By then we have approximately 2-3 weeks to prepare the presentation.

4 Project plan

4.1 PURPOSE OF THIS DOCUMENT

"The purpose of the project plan is to gather all relevant information, which is required to guide the project. It describes how the development progresses, and are used to control the project.

The project manager uses this document to keep track of the resources needed and the members of the group use this document to get an understanding of what to do. "

-Project Handbook (HiBu)

4.2 AUTHOR

Leif Henning Larsen

4.3 IN CHARGE OF ACTIVITY

Leif Henning Larsen

4.4 PROJECT OVERVIEW

4.4.1 PROJECT SCOPE

The main purpose of the project is to enhance the Workover Control System (WOCS) by integrating it with a VR (Virtual Reality) application called Source. . Source is a game engine produced by Valve, a corporation that made many astonishing game titles such as Half-Life and Portal. Personnel can use Virtual WOCS for training purposes. By simulating a ship or an oil rig, the personnel no longer have to travel far out to deep waters. This also includes emergency scenarios, such as equipment on fire.

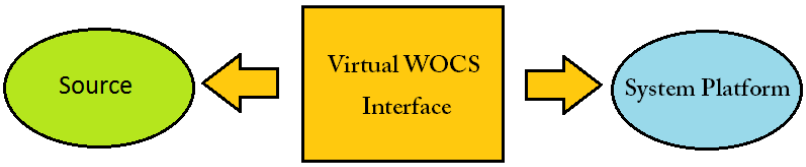

FIGURE 4 - CONNECTING THE TWO SYSTEMS

Our job is to make sure that these two systems can communicate, in such a way that it is possible to actually do personnel training onshore.

To make sure the scope does not get to big, we have selected a few minor “missions” involving some start up procedures and some alarm procedures.

4.4.2 PROJECT GROUP

The project group consists of the following members.

Name:	Initials:
Leif Henning Larsen	LHL
Torjus Engell	TE
Dejan Vukobratovic	DV
Jan Hansen	JH

TABLE 1 - MEMBERS OF PROJECT GROUP

4.4.3 PROJECT DOCUMENTATION

For a complete list of all official documents to be written, please refer to the attachment "List of documents".

We will throughout the project produce quite a lot of documents. Some of the documents are official, and shall be delivered in print to both HiBu and the employer.

As well as these documents we are going to write a pre-study report, we have to make sure that we write proper meeting requests/commentary, we will need to keep track of the time we use. These kinds of documents will not be delivered in paper format, but will be present in a digital format.

4.5 TECHNICAL PROCESS PLAN

4.5.1 PROCESS MODEL

These are the project methods and tools we will be using.

- Unified Process (UP)
- Unified Modeling Language (UML)
- Scrum

The main project method is UP, but we will gather some elements from scrum, such as daily internal meetings.

4.5.2 DOCUMENTATION AND CODE STANDARDS

During the project we will produce a lot of documents and code. It is therefore of great importance that these documents and all the code follow the same standards. This will make it easier for everybody to follow the process, and to see what we have been doing.

4.5.2.1 CODE

All the code we produce follows a specific coding standard.

Please refer to the document "Code standard" for the current code standard. [9]

4.5.2.2 DOCUMENTATION

All code must be well documented, and we can achieve this by following the standards from Javadoc. For this we will use something called "Doxygen", which can generate Javadoc documentation automatically.

All official documents must follow the same template, according to the document standard. Please refer to the attachment "Dokumentstandard".

4.6 PROJECT ORGANIZATION

4.6.1 ROLES AND RESPONSIBILITIES

- LHL : Project manager, implementation
- TE : Test, economy, web
- JH : Requirements, design, documentation
- DV : Analysis, risk, web

For a detailed explanation of every responsibility, please refer to the attachment "Responsibilities".

4.6.2 EXTERNAL INTERFACES

4.6.2.1 SUPERVISORS AND CENSORS

Name	Initials	Role
Espen B. Davidsen	EBD	External supervisor/censor
Alexander Risøy	AR	External supervisor
Olaf Hallan Graven	OHG	Internal censor
Karoline Moholth	KM	Internal supervisor

TABLE 2 - SUPERVISORS AND CENSORS

4.6.2.2 CONTACT INFORMATION

Contact information for external resources, as well as project group members, please refer to the attachment "Contact information".

4.6.2.3 INTERFACE TOWARDS THE SURROUNDINGS

In general the project group will have a web site, which will contain a blog where we will present updates and our progress. There will be held three presentations, where everyone is welcome.

4.6.2.3.1 INTERFACE TOWARDS EMPLOYER

We will have meetings with the external supervisors occasionally when needed. Otherwise we will stay in contact through email and phone calls. After our second presentation we will start to use one day a week at Nebbs office, in Asker.

4.6.2.3.2 INTERFACE TOWARDS HIBU

There will be weekly meetings with our internal supervisor. Before our presentations we will deliver our documentation. Both internal censor and supervisor will attend the presentations.

4.7 PROJECT MANAGEMENT

4.7.1 ACTIVITIES

The categories for the activities are as follows:

Category	
1xx	Administration and project control
2xx	Requirements
3xx	Design
4xx	Implementation
5xx	Test
6xx	Research

TABLE 3 - ACTIVITY CATEGORIES

For a complete list of activities, please refer to the attachment "Activities".

4.7.2 TIME SCHEDULE

4.7.2.1 GANTT CHARTS

We will create Gantt charts to keep track of what we should do, and how long each activity should last. Please refer to the attachment "Overordnet V2".

We will go into further detail about the iterations, where we will give a more detailed view on the time schedule for each activity. This will be done ahead of each iteration.

4.7.2.2 MILESTONES

- 1. Presentation: 07.01.2011
- 2. Presentation: 15.03.2011
- Industrial Gaming presentation: 07.04.2011
- 3. Presentation: 09.06.2011

4.7.2.3 ITERATIONS

We will work after the Unified Process method, which implies that the project will be split up in phases and iteration. All iterations are planned to be three weeks, however, due to the fact that we are to have our second presentation March 15th we have to extend the fourth iteration by a week. Also because of our lengthened due date, we have room for an iteration more, so the last two iterations, considered as transition, will be two weeks each.

When a new iteration starts we will have to produce a plan for that iteration, which describes how we will spend the time we have. It will also need to contain some information about how many hours per activity within that iteration we have.

At the end of each iteration we will have to produce an iteration report to compare the planned time consumption against the achieved time consumption.

We will need to have a working prototype of the product for the presentation at the Industrial Gaming conference. We should according to the plan have started on the construction phase by this time.

4.7.3 RESOURCES

4.7.3.1 HUMAN RESOURCES

We have four persons which will spend approximately 500 hours each.

4.7.3.2 SOFTWARE

- Microsoft Word
- Microsoft Excel
- Microsoft Project
- Source Engine
- System Platform
- 3D Studio Max
- AutoCAD

4.7.3.3 HARDWARE

- Gamepad controller for movement in VR
- Microsoft Kinect

4.7.3.4 DROPBOX

The project will use dropbox as a cooperation platform. By doing this we will all have the latest versions of each document, since it can be used as a synchronization tool.

4.7.4 BUDGET

The employer is required to cover all costs in the budget according to the contract. For the entire budget, please refer to the attachment "Budget".

4.7.5 CONTROL PLAN

4.7.5.1 MEETINGS

We will have meetings with the employer if needed, mainly to update them on the progress, and to see that there are no misunderstandings.

There will also be weekly meetings with our internal supervisor.

4.7.5.2 PLANS AND REPORTS

We will write plans and reports during the project, which will make it easier to keep control of the progress.

4.7.5.3 TIMESHEETS

Each member of the group will have to keep track of the hours they use, by writing a timesheet. This should be detailed enough to know what we were doing at a given date.

5 Risk analysis

5.1 PURPOSE OF THIS DOCUMENT

The main purpose of this risk analysis document is to improve our project quality. In addition, it helps identify risks associated with the project and describe which source of action should be taken. This document will cover several risk factors. Among them are risk factors related to the system, risk factors where the origin is an outsider, and finally the human risk factors.

5.2 AUTHOR

Dejan Vukobratovic

Jan Hansen

5.3 IN CHARGE OF ACTIVITY

Dejan Vukobratovic

5.4 IDENTIFYING THE RISK

5.4.1 RISK ACKNOWLEDGEMENT AND IDENTIFICATION

We believe that it is important to face problems that might occur during the project runtime. Therefore, it is important to identify and acknowledge the reality of a risk. To deny existence of risks would be bad, as something might show up later and cause trouble. In order to deal with this problem, we wrote this document so that it can help us solve troubles by creating good risk evaluations. We will map the risk factor, identify the weak spots where they might occur, and look at how we can improve our work so that hopefully none of them might spawn.

The most common questions in the risk subject are "Is there a risk?" and "How likely is this risk to occur?". Based on these questions, our risk evaluation will identify the risks by how much threat they pose to our project. The following graph explains why it is extremely important to understand the risks and what consequences they might bring.

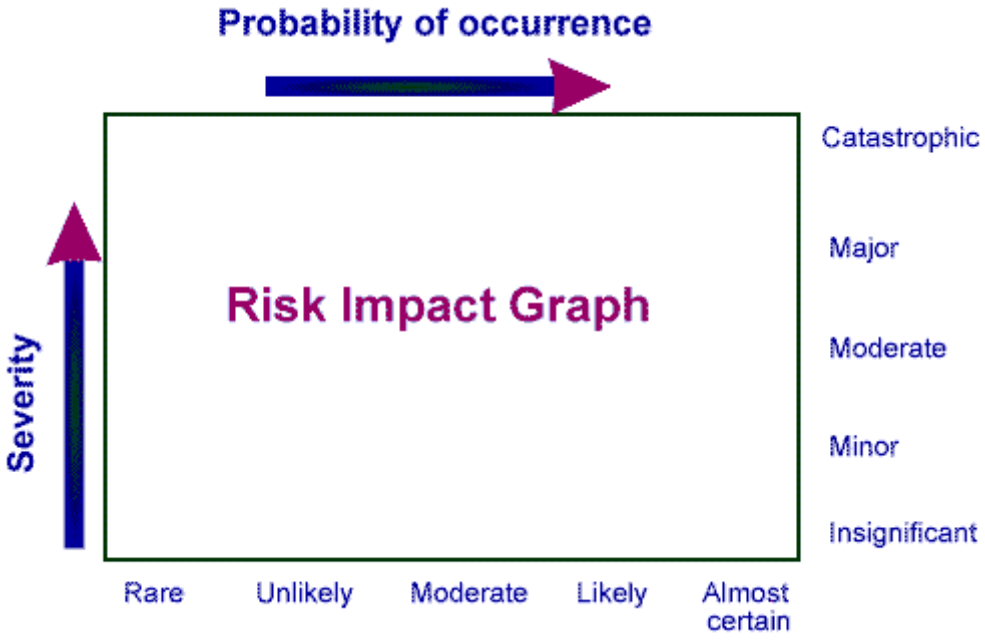

FIGURE 5 - RISK IMPACT GRAPH

The higher the chances are for a risk to occur (probability), and the bigger consequence (severity), the higher the risk will be. Based on this observance, we can conclude that a risk is the sum of probability of occurrence and the severity.

In order to handle this risk problem, we need to pay attention to our risk management. We believe that we probably won't cover all risks involved; however we will do our best to identify the most threats and find a way to deal with them so that it might save us time later. Some risks might be quite small and therefore not takes within the analysis. We have put the risks into tables where they are identified by their probability of occurrence and how severe they are. This is a direct approach from figure 2, "Risk Graph Impact", above. Therefore, risks with high consequences or high severities are the ones we will have to take extra care of to make sure we have done what could be done in order to prevent them from occurring.

This document will be updated throughout the whole project runtime. Some risks might come up later, and they will be included as well. Some risks might not even happen because sometimes we might we lucky enough and build our system without stepping into the problem areas. Like mentioned before, this document will help us improve to keep a good quality of the system we are building. We will follow the risk analysis and take the necessary actions identified below if the corresponding risks occur.

5.5 THE RISKS

5.5.1 RISK LIST

Risks related to requirements

- R1: A-requirements not met
- R2: B-requirements not met
- R3: C-requirements not met

Technical and system related risks

- R4: EyeSim license not received
- R5: Bad performance
- R6: PC problems

Human risks

- R7: Breakup
- R8: Low motivation
- R9: Lack of knowledge
- R10: Disagreement
- R11: Illness and injuries

Outsiders and other risks

- R12: Late delivery of hardware
- R13: Burglars at school
- R14: Classroom not assigned/available
- R15: Nebb goes bankrupt

5.5.2 RISK TABLE

All risks we have identified will be put in the following table:

Risk name:	*(risk name)*	Risk ID:	*(IDnum)*
Probability of occurrence:	*(high, middle, low)*	Level of severity:	*(high, middle, low)*
Prevention:	*(description of how to prevent the risk from occurring)*		
Consequence:	*(description of the risk consequence)*		

TABLE 4 - RISK TABLE TEMPLATE

Risks will be detailed with name and their ID number, consisting of the letter **R** and a number **X** (range is not specified at the moment, but we doubt we will go further than 99). For example, there can be a risk called **R54**. All risks are defined under their own title of category (for example requirements), suggesting they belong in the requirements sector.

5.5.3 RISKS RELATED TO REQUIREMENTS

Risk name:	**A-requirements not met**	Risk ID:	**R1**
Probability of occurrence:	Middle	Level of severity:	High
Prevention:	Group members should be more involved in the project. Try to understand the technology we use and make sure we know how to operate it.		
Consequence:	Not meeting the A-requirements would indicate that we have not implemented the functional description from Nebb, meaning that we would not meet the expectations of the outcome product.		

TABLE 5 - R1

Risk name:	**B-requirements not met**	Risk ID:	**R2**
Probability of occurrence:	Middle	Level of severity:	Middle
Prevention:	Group members should be more involved in the project. Try to understand the technology we use and make sure we know how to operate it.		
Consequence:	Not meeting the B-requirements would indicate that we have not implemented some of the functions described by Nebb, meaning that we would not meet the expectations of the outcome product. However, these consequences are not as important as the ones associated with A-requirements.		

TABLE 6 - R2

Risk name:	**C-requirements not met**	Risk ID:	**R3**
Probability of occurrence:	Middle	Level of severity:	Low
Prevention:	Group members should be more involved in the project. Try to understand the technology we use and make sure we know how to operate it.		
Consequence:	Not meeting the C-requirements would indicate that we have not implemented some of the functions described by Nebb. These requirements are not as important for the final product though, so there is no immediate panic associated to this risk.		

TABLE 7 - R3

5.5.4 TECHNICAL AND SYSTEM RELATED RISKS

Risk name:	**EyeSim license not received**	Risk ID:	**R4**
Probability of occurrence:	High	Level of severity:	High
Prevention:	Keep good contact with Invensys and Nebb. Making sure we show enthusiasm and that we really want to work with EyeSim and that we depend on it.		
Consequence:	If EyeSim falls off, we will have to find another graphical tool to represent the WOCS system in. There are other options such as the UI-engine and the Source Engine.		

TABLE 8 - R4

Risk name:	**Bad performance**	Risk ID:	**R5**
Probability of occurrence:	Middle	Level of severity:	Middle
Prevention:	Making sure we meet the system requirements and recommended settings.		
Consequence:	EyeSim and System Platform might run bad on our laptops since they are optimized for running on server based platforms. This might result in a major loss of time if appropriate hardware isn't acquired.		

TABLE 9 - R5

Risk name:	**PC problems**	Risk ID:	**R6**
Probability of occurrence:	Middle	Level of severity:	Middle
Prevention:	Group members should take care of their PC's. Regular system scan and antivirus updates should be performed.		
Consequence:	PC problems can quickly evolve into a major time sink.		

TABLE 10 - R6

5.5.5 HUMAN RISKS

Risk name:	**Breakup**	Risk ID:	**R7**
Probability of occurrence:	Middle	Level of severity:	Middle
Prevention:	Group members should keep a professional attitude while working on this project. Keep work and private time separated.		
Consequence:	Breakups can make group members unmotivated and distracted. This can have a serious impact on the work that has to be done.		

TABLE 11 - R7

Risk name:	**Low motivation**	Risk ID:	**R8**
Probability of occurrence:	Middle	Level of severity:	Low
Prevention:	Group members should be more involved in the project. We all want a good final grade, and we need to work to make that happen. By organizing celebrations of milestones with going out every now and then, it will help a lot with motivation.		
Consequence:	Low motivation can keep group members distracted and not focused on the project.		

TABLE 12 - R8

Risk name:	**Lack of knowledge**	Risk ID:	**R9**
Probability of occurrence:	High	Level of severity:	Middle
Prevention:	Everyone should read the documentation based on the systems we will work with. Make sure that we understand what we are doing.		
Consequence:	Lack of knowledge might be a consequence occurring often because we will learn about Source and System Platform throughout the project. This might bring time sinks yet again, because group members will need to adapt to the new hardware and software used, and learn how to operate on it.		

TABLE 13 - R9

Risk name:	**Disagreement**	Risk ID:	**R10**
Probability of occurrence:	Middle	Level of severity:	Middle
Prevention:	Democracy, milestones, celebration, clean rules.		
Consequence:	Disagreements will lead to arguments. It is time consuming and can in a worst case scenario lead to one of the group members leaving the group and taking all his work with him or destroy others.		

TABLE 14 - R10

Risk name:	**Illness and injuries**	Risk ID:	**R11**
Probability of occurrence:	Low	Level of severity:	High
Prevention:	Take precaution on slippery surface (especially during the winter). Since most of us drive around 100 km each day to/from school, make sure we don't exceed speed limits etc. unnecessarily. Everyone should eat a healthy and varied diet, to make sure we have enough energy to work with the project.		
Consequence:	Injury/illness can lead to a group member being away for a while. This means other group members will have to step up and do that persons work. This leads yet again to stress and time loss.		

TABLE 15 - R11

5.5.6 OUTSIDERS AND OTHER RISKS

Risk name:	**Late delivery of hardware**	Risk ID:	**R12**
Probability of occurrence:	Low	Level of severity:	High
Prevention:	Make sure we have the presentation hardware (screens and controllers) at least 1 day before presentations occur.		
Consequence:	Consequences are that a good presentation demo will not be shown in case equipment does not arrive.		

TABLE 16 - R12

Risk name:	**Burglars at school**	Risk ID:	**R13**
Probability of occurrence:	Middle	Level of severity:	Low
Prevention:	Group members should make sure the working room at school is locked when we are not there. All high-cost equipment and PC's should be taken home over night.		
Consequence:	During the last 3 years, we can recall a few projector thefts at school. They obviously like to steal equipment, and if they stole our PC's or other hardware such as the projector, we would have a few very bad weeks at school.		

TABLE 17 - R13

Risk name:	**Classroom not assigned/available**	Risk ID:	**R14**
Probability of occurrence:	Low	Level of severity:	High
Prevention:	Group members should keep time dates of reservations in order. Keep good contact with the ones responsible to assign project groups a classroom.		
Consequence:	Not being able to attend presentations due to rooms not being available is a major concern.		

TABLE 18 - R14

Risk name:	**Nebb goes bankrupt**	Risk ID:	**R15**
Probability of occurrence:	Low	Level of severity:	High
Prevention:	There is nothing we can do to prevent this.		
Consequence:	There is a high chance that the project will be discontinued.		

TABLE 19 - R15

5.5.7 RISKS WHICH OCCURRED

5.5.7.1 R4 – EYESIM LICENSE NOT RECEIVED

During the middle of March 2011, we received a note from the Italian team behind EYESIM telling us that they had no time to help us with our project after all. This proved to be a severe setback because we had prepared all 3D models for EYESIM. After realizing the problem, we had to choose a game engine as a replacement and the choice quickly fell on Source engine, which was used to develop great titles such as Half-Life 2 and Portal.

However, new problems quickly surfaced. One of the major ones was that Source has a limit of 10000 polygons for each model. Since we drew most of our 3D models in CAD software, they had a much larger polygon count (50000+). We had to redesign and rebuild the models which took a huge chunk of time. However, we managed to do what we planned within the iteration in March.

At the same time, it was good to finally start on the interface implementation in Source. We managed to pull ourselves together and made it for the 2nd presentation. For this risk, we have to say it was good to evaluate backup plans in case it went wrong. Source engine turned out to be a wise choice after all, due to its great community and tutorials. However, the 3D modeling part took three times as much time as planned.

5.5.7.2 R8 – LACK OF MOTIVATION

This one did not occur too often. One time we took a day off for a trip to Sweden. There we bought a lot of different stuff, such as drinks and food and clothing. It was really great to relieve our minds from the hard work before the 1st presentation. Some other times when motivation was not on top, we took early days off and continued working later in evenings at home, or assigned a longer day some other time.

5.5.7.3 R9 – LACK OF KNOWLEDGE

We did not encounter this that often, but it is still worthy of mentioning. The most notable occurrence was when we started using Source. We had no clue of how to start so we had some extensive research and tutorial reading to do. However, this had to be done since there was no way back to EYESIM.

5.5.7.4 R10 – DISAGREEMENT

When it comes to disagreement, this is something which probably happens for everyone.

When four people work on the same project, there are usually different views of how one should approach a situation. This is where the problems were at hand for us. Coming to agreement between different approaches is something we have spent time on quite a lot. Apart from that we had no severe occasions where we had fights or verbal outbursts at each other.

5.5.7.5 R11 – ILLNESS AND INJURIES

All of us have had some absence. Nothing extraordinary though, and no one was absent for a long time. This also meant that people did not have to work extra to cover up for the absent ones, since we worked on documentation from home when we were absent.

6 Requirement specification

6.1 PURPOSE OF THIS DOCUMENT

"A requirements specification is a complete description of the behavior of a system to be developed. This document contains requirements description of accurately defined properties or constraints the system must fulfill. In addition, it contains a set of use cases (UML) that describe several interactions the users have with the system. Use case modeling is a part of functional requirements, but there can also be examples of non-functional requirements. Non-functional requirements are requirements that put constraints on the system (often by quality standards that affect general system performance). It is crucial that the requirements specification document specifies briefly what the system will do without specifying how it will be done."

-Wikipedia, Project Handbook (HiBu)

Requirements specification document will be updated as the project is in progress. Some things might change, and that is why this document is described as dynamic. The list of requirements in this document is defined from a set of functions we received from the employer (Nebb). Requirements will help us define the system behavior and also help us understand the project tasks correctly. This document contains all requirements we define throughout the project runtime. By utilizing Use-case diagrams, we will show how they can meet the requirements. Only functional requirements are to be met in Use-cases.

6.2 AUTHORS

- Jan Hansen
- Dejan Vukobratovic

6.3 IN CHARGE OF ACTIVITY

- Jan Hansen

6.4 REQUIREMENTS

6.4.1 REQUIREMENTS LIST

6.4.1.1 CATEGORY A

6.4.1.1.1 FUNCTIONAL

- A1-F: Activate Emergency Shutdown (ESD)
- A2-F: Initialization of ESD alarm
- A3-F: Evacuation to designated zones
- A4-F: Hydraulic Power Unit (HPU) - Start pump
- A5-F: Clogged Filter Alarm (CFA)
- A6-F: Update/warn Human Machine Interface (HMI)
- A7-F: Augmented Reality alarm threshold

6.4.1.1.2 NON-FUNCTIONAL

- A8: Communication

6.4.1.2 CATEGORY B

6.4.1.2.1 FUNCTIONAL

- B1-F: Regulate pressure
- B2-F: WOCS start-up procedure - Purge Container
- B5-F: Real time info on valves in Source

6.4.1.2.2 NON-FUNCTIONAL

- B3: Models/animation file type(This requirement is deprecated because EyeSim was not used for implementing the virtual world)
- B4: Development of an API for the system

6.4.1.3 CATEGORY C

6.4.1.3.1 FUNCTIONAL

- C1-F: Raise/lower stack
- C2-F: Artificial Intelligence - Movement of crew
- C3-F: Artificial Intelligence - Abilities of crew
- C4-F: Kinect Integration

6.4.2 REQUIREMENT DETAILS

An introduction to how requirements are divided into three distinct groups, and what the reason for this is, can be found in the attachment "Introduction to Requirements". Please refer to this file.

The environment to be simulated is a WOCS Container installation on an oil rig. It is basically a hydraulic system to operate vents that control the flow of oil in a Christmas tree. The core of the system is installed in a container stationed on the oil rig. This container contains everything needed to open and close the vents with hydraulic pressure. The requirements are based on such an installation.

All requirements will be set up in tables to provide a good overview. We have used the table template below for all requirements. Each table is described by the requirement ID and the name of requirement. The ID will be detailed as for example **B3-F**, where **B** is the requirements category (either A, B or C), **3** is the number of the requirement, and **F** identifies that the requirement is functional. Non-functional requirements will not have the last letter **F** in their ID.

Requirement details					
Category:	*(A, B, C)*	ID:	*(IDnum)*	Origin:	*(how the requirement was identified)*
Description:	*(description of the requirement)*				
Date:	*(dd.mm.yyyy)*		Functional/non-functional:		*(functional, non-functional)*
Meets Use-case:	*(Use-case ID)*		Hardware/Software Implementation:		*(hardware, software)*
Requirement state:	*(approved, denied, suggested)*				
Comments:					

TABLE 20 - REQUIREMENT TABLE TEMPLATE

6.4.3 REQUIREMENTS OF CATEGORY A

6.4.3.1 A1-F: ACTIVATE EMERGENCY SHUTDOWN (ESD)

Requirement details					
Category:	A	ID:	A1-F	Origin:	Function description from employer
Description:	Activates the emergency shutdown procedure, where the stack is disconnected from the rig. Effects must be visible on both System Platform and Source. Alarms should run. This action can be triggered at all times!				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC1		Hardware/Software Implementation:		Software and hardware
Requirement state:	*approved*				
Comments:	Alarms should run until turned off. ***Update: Changed alarm run time from 30 second to "until turned off". Functionality in System Platform has its own predefined alarm sequence so we can't manipulate the alarm run time ourselves.***				

TABLE 21 - REQUIREMENT A1-F

6.4.3.2 A2-F: INITIALIZATION OF ESD ALARM

Requirement details					
Category:	A	ID:	A2-F	Origin:	Function description from employer
Description:	Initialization of ESD alarm should be implemented in both System Platform and Source. In Source simulation, there should be a VR-button for sounding the alarm. Alarms should be visible in both systems as well.				
Date:	26.04.2011		Functional/non-functional:		Functional
Meets Use-case:	UC2		Hardware/Software Implementation:		Software and hardware
Requirement state:	*approved*				
Comments:	The hardware bit is a warning light that should flash (connected through USB) when the alarm is running. As requirement A1-F, this requirement should also run for at least 30 seconds, or until turned off. The hardware bit is deprecated because the flashing light is implemented in Source.				

TABLE 22 - REQUIREMENT A2-F

6.4.3.3 A3-F: EVACUATION TO DESIGNATED ZONES

Requirement details					
Category:	A	ID:	A3-F	Origin:	Function description from employer
Description:	When the alarm is on, workers on the rig should immediately evacuate to the nearest designated zone.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC3		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:	The player (worker on the oilrig in Source) has a 2 minute limit to evacuate to the designated zone.				

TABLE 23 - REQUIREMENT A3-F

6.4.3.4 A4-F: HYDRAULIC POWER UNIT (HPU) - START PUMP

Requirement details					
Category:	A	ID:	A4-F	Origin:	Function description from employer
Description:	Start the pump that transfers hydraulic oil from return tank to supply tank. This action will be implemented in Source.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC4		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:	This start-up sequence should take 2 minutes maximum.				

TABLE 24 - REQUIREMENT A4-F

6.4.3.5 A5-F: CLOGGED FILTER ALARM (CFA)

Requirement details					
Category:	A	ID:	A5-F	Origin:	Function description from employer
Description:	When an oil filter is clogged, it should issue an alarm so the operator can change it. This filter alarm should be visible in both systems.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC5		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 25 - REQUIREMENT A5-F

6.4.3.6 A6-F: UPDATE/WARN HUMAN MACHINE INTERFACE (HMI)

Requirement details					
Category:	A	ID:	A6-F	Origin:	Function description from employer
Description:	All manual override of valve operations in Source should update in the HMI on System Platform.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC6		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 26 - REQUIREMENT A6-F

6.4.3.7 A7-F: AUGMENTED REALITY ALARM THRESHOLD

Requirement details					
Category:	A	ID:	A7-F	Origin:	Function description from employer
Description:	Valves should have a configurable pressure limit in Source. The operator should be able to set threshold values through both systems.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC7		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 27 - REQUIREMENT A7-F

6.4.3.8 A8: COMMUNICATION

Requirement details					
Category:	A	ID:	A8	Origin:	Torjus
Description:	A stable communication between Source and System Platform must be established.				
Date:	24.02.2011		Functional/non-functional:		Non-Functional
Meets Use-case:	"Figure 2 - Main use case diagram"		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 28 - REQUIREMENT A8

6.4.4 REQUIREMENTS OF CATEGORY B

6.4.4.1 B1-F: REGULATE PRESSURE

Requirement details					
Category:	B	ID:	B1-F	Origin:	Function description from employer
Description:	The system operator should be able to set and regulate the supply pressure of tanks.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC8		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 29 - REQUIREMENT B1-F

6.4.4.2 B2-F: WOCS START-UP PROCEDURE - PURGE CONTAINER

Requirement details					
Category:	B	ID:	B2-F	Origin:	Function description from employer
Description:	The system operator should be able to purge the WOCS container by applying overpressure.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC8		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 30 - REQUIREMENT B2-F

6.4.4.3 B3: MODEL/ANIMATION FILE TYPE

Requirement details					
Category:	B	ID:	B3	Origin:	Function description from employer
Description:	The model and animation file types should be of any Autodesk standard.				
Date:	02.12.2010		Functional/non-functional:		Non-functional
Meets Use-case:	*N/A*		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:	Licenses for Autodesk software are distributed for free (student versions only). This requirement is deprecated because Source was used for implementing the virtual world.				

TABLE 31 - REQUIREMENT B3

6.4.4.4 B4: DEVELOPMENT OF AN API FOR THE SYSTEM

Requirement details					
Category:	B	ID:	B4	Origin:	Function description from employer
Description:	An API should be developed for the system.				
Date:	02.12.2010		Functional/non-functional:		Non-functional
Meets Use-case:	*N/A*		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 32 - REQUIREMENT B4

6.4.4.5 B5-F: REAL TIME INFO ON VALVES IN SOURCE

Requirement details					
Category:	B	ID:	B5-F	Origin:	Function description from employer
Description:	There should be an option to show info about the state of valves in Source. The state of a valve is its pressure and open/closed status.				
Date:	26.04.2011		Functional/non-functional:		Functional
Meets Use-case:	UC6		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 33 - REQUIREMENT B5-F

6.4.5 REQUIREMENTS OF CATEGORY C

6.4.5.1 C1-F: RAISE/LOWER STACK

Requirement details					
Category:	C	ID:	C1-F	Origin:	Function description from employer
Description:	The stack in Source should be controllable. It should have options for raising and lowering.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	UC9		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:	This requirement can be up-scaled to provide a visual "wow" feeling of the rig simulation in Source. ***Update: Removed time measurement since this is of no real value as long as it can be operated so it moves up or down.***				

TABLE 34 - REQUIREMENT C1-F

6.4.5.2 C2-F: ARTIFICIAL INTELLIGENCE - MOVEMENT OF CREW

Requirement details					
Category:	C	ID:	C2-F	Origin:	Function description from employer
Description:	Crew members controlled by AI should be able to move around the rig in a specified route (work cycle).				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	*N/A*		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:	This concerns Source system only. Using nodes as waypoints.				

TABLE 35 - REQUIREMENT C2-F

6.4.5.3 C3-F: ARTIFICIAL INTELLIGENCE - ABILITIES OF CREW

Requirement details					
Category:	C	ID:	C3-F	Origin:	Function description from employer
Description:	Crew members controlled by AI should be able to execute a simple task, such as opening/closing valves and pushing alarm buttons.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	*N/A*		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:	This concerns Source system only. Each AI controlled crew member should execute a simple action – turning the valves assigned to them.				

TABLE 36 - REQUIREMENT C3-F

6.4.5.4 C4-F: KINECT INTEGRATION

Requirement details					
Category:	C	ID:	C4-F	Origin:	Function description from employer
Description:	By connecting a kinect controller to the system it should be able to control valves and movement of the player.				
Date:	02.12.2010		Functional/non-functional:		Functional
Meets Use-case:	*N/A*		Hardware/Software Implementation:		Software
Requirement state:	*approved*				
Comments:					

TABLE 37 - REQUIREMENT C4-F

6.5 ANALYSIS OF THE SYSTEM

6.5.1 SYSTEM IN GENERAL

By utilizing this basic Use-case diagram, we have illustrated the following system based on the task definition given by Nebb:

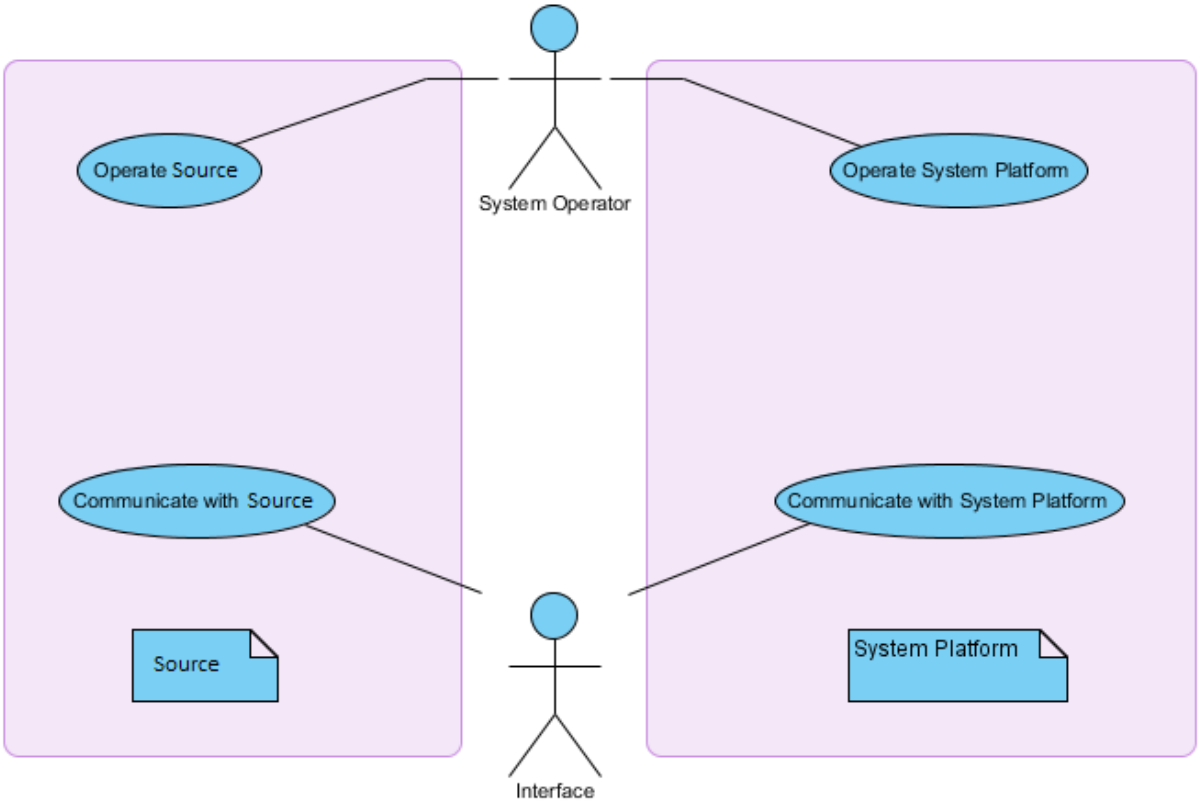

FIGURE 6 - MAIN USE CASE SYSTEM DIAGRAM

The task definition from Nebb describes that they want to integrate System Platform and Source together so that they can simulate both systems simultaneously. We will do this by making an interface system that can communicate between the two pieces of software. What happens in one system, is supposed to happen in the other system as well.
Figure 2 illustrates the basic operations of the system and how the interface is meant to be. System Operator is a human user outside the system, who controls both Source and System Platform. The interface is integrated in the system itself. It is a communication system between Source and System Platform. Because the interface needs to be explained in terms of HOW it works, we will be covering it in the design and implementation phases of the project.

6.5.2 USE-CASE DIAGRAM EXTENDED

We will focus on describing what the operator and system can do.

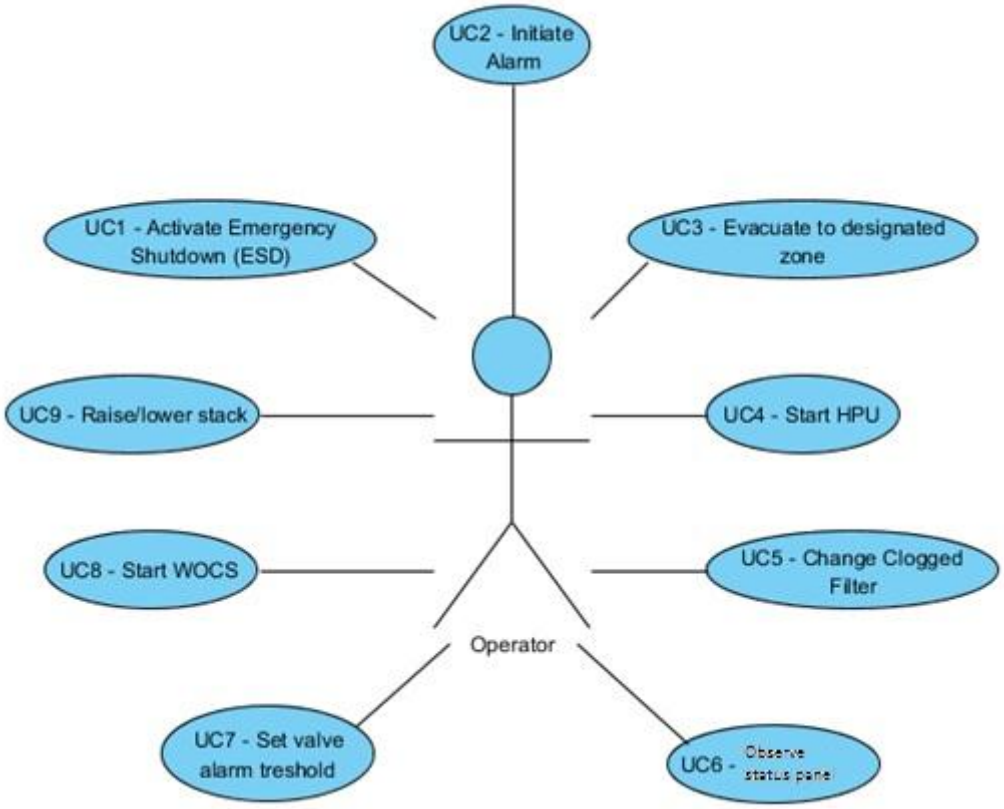

FIGURE 7 - EXTENDED USE CASE DIAGRAM (OPERATOR)

6.5.3 DESCRIPTION OF ACTORS

6.5.3.1 OPERATOR

The operator is the actor taking care of operative actions. He/she is the user of the system and can operate both Source and System Platform.

6.5.3.2 INTERFACE

The interface is responsible for handling communication between Source and System Platform.

6.5.4 USE-CASES IN DETAIL

6.5.4.1 ACTOR – OPERATOR

6.5.4.1.1 UC1 – ACTIVATE EMERGENCY SHUTDOWN

Description:	➢ Operator activates the emergency shutdown procedure
Meeting specification:	➢ A1-F
Predecessors:	➢ The simulated environment requires the trainee to activate the emergency shutdown
Event description:	1. Operator activates emergency shutdown procedure 2. An alarm is started
Post conditions:	➢ The alarm is running
Optional:	➢

TABLE 38 - UC1

6.5.4.1.2 UC2 – INITIATE ALARM

Description:	➢ Manually initiate an alarm.
Meeting specification:	➢ A2-F
Predecessors:	➢ An accident has happened in the simulated environment.
Event description:	1. Activate alarm by pushing the virtual button. 2. An alarm starts.
Post conditions:	➢ The alarm is running.
Optional:	➢

TABLE 39 - UC2

6.5.4.1.3 UC3 – EVACUATE TO DESIGNATED ZONE

Description:	➢ Evacuate to the designated zone.
Meeting specification:	➢ A3-F
Predecessors:	➢ The emergency shutdown has been activated. ➢ The emergency shutdown alarm is running.
Event description:	1. Evacuate to the nearest zone.
Post conditions:	➢ You have evacuated and mission is ended.
Optional:	➢

TABLE 40 - UC3

6.5.4.1.4 UC4-START HPU

Description:	➢ Start the pump that transfers hydraulic oil from return tank to supply tank.
Meeting specification:	➢ A4-F
Predecessors:	➢ Your mission is to start the HPU.
Event description:	1. Activate the HPU by pushing the virtual button. 2. HPU is started.
Post conditions:	➢ HPU is up and running.
Optional:	➢

TABLE 41 - UC4

6.5.4.1.5 UC5 – CHANGE CLOGGED FILTER

Description:	➢ Operator changes the clogged filter.
Meeting specification:	➢ A5-F
Predecessors:	➢ A filter is clogged. ➢ An alarm is shown in HMI. ➢ Your mission is to change the filter.
Event description:	1. Vent the filter. 2. Take out the clogged filter. 3. Set in new filter. 4. Resume pressure.
Post conditions:	➢ Pressure level is normal. ➢ System is running normally.
Optional:	➢

TABLE 42 - UC5

6.5.4.1.6 UC6 – OBSERVE STATUS PANEL

Description:	➢ Operator observes the status panel in Source.
Meeting specification:	➢ A6-F
Predecessors:	➢ The operator is ready to observe the HMI window in Source.
Event description:	1. The operator activates the HMI. 2. The HMI shows a virtual window with information.
Post conditions:	➢ The operator observes the status panel.
Optional:	➢

TABLE 43 - UC6

6.5.4.1.7 UC7 – SET VALVE ALARM THRESHOLD

Description:	➢ The operator sets a new valve alarm threshold.
Meeting specification:	➢ A7-F
Predecessors:	➢ The system is running normally.
Event description:	1. The operator sets a new value for the alarm threshold. 2. The new value has been set.
Post conditions:	➢ The new value shows up correctly in HMI.
Optional:	➢

TABLE 44 - UC7

6.5.4.1.8 UC8 – START WOCS

Description:	➢ The operator starts the Workover Control System.
Meeting specification:	➢ B1-F,B2-F
Predecessors:	➢ WOCS is not running.
Event description:	1. The operator sets a pressure value. 2. The new value is regulated. 3. An overpressure is applied. 4. The tank is purged.
Post conditions:	➢ WOCS is running.
Optional:	➢

TABLE 45 - UC8

6.5.4.1.9 UC9 – RAISE/LOWER STACK

Description:	➢ Operator raises or lowers the stack.
Meeting specification:	➢ C1-F
Predecessors:	➢ The stack is ready to be raised or lowered.
Event description:	1. The operator pushes the button for raising/lowering the stack. 2. The stack raises/lowers itself.
Post conditions:	➢ The stack is in its proper place.
Optional:	➢

TABLE 46 - UC9

7 Test specification

7.1 PURPOSE OF THIS DOCUMENT

The purpose of this document is to have predefined ways to test the requirements which are written down in the Requirement specification. From the requirement specification we find graded requirements from most important, the fundamentals of the program for actually getting it up and running graded A, till C requirements that operate as extra features. This document gives us the guidelines we need when we are about to test these requirements so the testing methods aren't swayed by the problems that emerge.

7.2 AUTHOR

Torjus Engell

7.3 IN CHARGE OF ACTIVITY

Torjus Engell

7.4 SHORT ABOUT THE TESTING METHOD

For full description about which test we are performing and how they work in practice, please read the "Test strategy" – document provided as attachment.[2]

Since we have access to most of the internal data while doing this project, both a "black box and white box testing – approach" sounds like the logical way to go. Together with unit testing of each function, regression tests when we add a new function, with internal testing of the entire system in between. This to ensure the old functions working properly with the new and the entire system is cooperating.

Further down this document, is a table created for testing. This will be the standard for every test that is to be done. When creating a test document, we have to be able to trace it back to the person who was responsible for the testing of a particular requirement. Together with the traceability issue, we need to know about what system has been tested at which time. A test report will also contain what values we tested and the errors which a cured during the testing period. The person, who has written the software, will not be the one to test it. This is because of the psyche of humans, not wanting to find errors in their own work

Short description, what to do prior to performing a test:

1. Find and identify the part of the code you are to test, followed by the test code corresponding to the tests.
2. Create a test report with the correct test code, name of the tester, date etc.

You are now ready to start testing.

7.5 TEST SPECIFICATION TEMPLATE

Every test will have an ID attached to it. The ID will be detailed as for example **T3**, where **T** is the letter for test and **3** is the number of the test. All tests are based on their requirements, identified in the upper left corner of the table.

Test details					
Req. ID:	*(A,B,C)*	Name	*(Test name)*	Test ID:	*(Test ID)*
Test description:	*(Test description)*				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Name of tester)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• *(Test approach)*				
Expected results/errors:	• *(Expected errors and results)*				
Variable inputs:	*(Inputs)*				
Errors:	*(Encountered errors)*				
Outcome:	*(What happened?)*				

TABLE 47 - TEST TABLE TEMPLATE

7.6 THE TESTS

7.6.1 MAIN OBJECTIVES

7.6.1.1 ACTIVATE EMERGENCY SHUTDOWN (ESD)

Test details					
Req. ID:	A1-F	Name	Activate emergency shutdown	Test ID:	T1
Test description:	Activates the emergency shutdown procedure, where the stack is disconnected from the rig. Effects must be visible on both System Platform and Source. Alarms should run. This action can be triggered at all times!				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Run the ESD from Source. • Is alarm on in both applications? • Is the stack disconnected from the rig? • Run the ESD from System Platform. • Is alarm on in both applications? • Is the stack disconnected from the rig?				
Expected results/errors:	• The ESD cannot be activated at a certain time. • The alarm is functioning when turned on in System Platform, but not when turned on in Source. • The alarm is functioning when turned on in Source, but not when turned on in System Platform. • The alarm does not make a sound/is not visible.. • The stack beneath the rig is not disconnected.				
Variable inputs:	N/A				
Errors:	*(Encountered errors)*				
Outcome:	The ESD can be activated at any given time. *(Pass/fail)* The alarm sounds and is visible on both systems simultaneously. *(Pass/fail)* Stack is disconnected from the rig. *(Pass/fail)*				

TABLE 48 - TEST T1

7.6.1.2 INITIALIZATION OF ALARMS

Test details					
Req. ID:	A2-F	Name	Initialization of alarms	Test ID:	T2
Test description:	Initialize alarms in Source. Initialize alarms in System Platform.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Run the alarm in Source by pushing a VR button for manual alarm initialization. • Is the alarm on and visible in both System Platform and Source? • Manually run the alarm in System Platform. • Is the alarm on and visible in both System Platform and Source?				
Expected results/errors:	• The VR button is not operational. • The alarm does not run at all.				
Variable inputs:	N/A				
Errors:	*(Encountered errors)*				
Outcome:	The alarm sounds and is visible on both systems. *(Pass/fail)* The VR-button is operational. *(Pass/fail)* The button in System Platform is sending alarm command to source. (*Pass/fail)*				

TABLE 49 - TEST T2

7.6.1.3 EVACUATION TO DESIGNATED ZONE

Test details					
Req. ID:	A3-F	Name	Evacuation to designated zone	Test ID:	T3
Test description:	After the ESD alarm is initiated, the operator moves to the designated safety zone. Alarm can be turned off ONLY after the operator has entered the safe zone.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Run ESD in any of the systems. • Move to the nearest designated zone.				
Expected results/errors:	• The user can turn off the alarm without moving to safe zone • The alarm doesn't run				
Variable inputs:	N/A				
Errors:	*(Encountered errors)*				
Outcome:	The operator turn the alarm on, moves to the safe zone, and get's the opportunity to turn the alarm off. (*pass/fail)*				

TABLE 50 - TEST T3

7.6.1.4 HYDRAULIC POWER UNIT (HPU)

Test details					
Req. ID:	A4-F	Name	Hydraulic Power Unit start-up	Test ID:	T4
Test description:	Activate the HPU from Source.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run source and System Platform. • Start the pump in Source. • Does the pump start, and is it visible in System Platform?				
Expected results/errors:	• The HPU will not activate from Source. • System Platform will not register the HPU activation				
Variable inputs:	N/A				
Errors:	*(Encountered errors)*				
Outcome:	The HPU is activated and registered in System platform. *(Pass/fail)*				

TABLE 51 - TEST T4

7.6.1.5 CLOGGED FILTER ALARM (CFA)

Test details					
Req. ID:	A5-F	Name	Clogged Filter Alarm (CFA)	Test ID:	T5
Test description:	Clog the filter, so that an alarm can be issued visually by an indicator in both systems.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Manually clog the filter. • Is the CFA indicator visible in both systems?				
Expected results/errors:	• The alert will not be visible if the filter is clogged, either in system platform or source • The alert will be visible even if the filter is not clogged.				
Variable inputs:	A function (in code) that marks the filter as clogged. Written in both systems.				
Errors:	*(Encountered errors)*				
Outcome:	Alarm on when the filter is clogged. *(Pass/fail)* The CFA is visible on both systems. *(Pass/fail)*				

TABLE 52 - TEST T5

7.6.1.6 UPDATE/WARN HUMAN MACHINE INTERFACE

Test details					
Req. ID:	A6-F	Name	Update/warn HMI	Test ID:	T6
Test description:	Turn valves on/off in Source manually (by operators vision) and see if the status indicators update in real-time in System Platform.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run source and System Platform. • Manually turn a valve on/off in Source. • Is the valve status indicator in System Platform (HMI) updated? *NOTE: Time delay has not yet been accounted for!*				
Expected results/errors:	• The valves turned on/off have no effect at all. • The HMI in System Platform will not register the changes done on the valve.				
Variable inputs:	Interface modifications (on/off functions).				
Errors:	*(Encountered errors)*				
Outcome:	The HMI in System Platform registers the changes done to any valve and displays this. *(Pass/fail)*				

TABLE 53 - TEST T6

7.6.1.7 AUGMENTED REALITY ALARM THRESHOLD

Test details					
Req. ID:	A7-F	Name	Augmented Reality Alarm Threshold	Test ID:	T7
Test description:	The pressure limit/threshold on any valve can be manually configured through both systems. Test this on both systems.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Change the pressure limit on a valve on any of the systems. • Is the new pressure limit registered on both systems? • Test again on the system that has not yet been tested.				
Expected results/errors:	• None of the systems register the change in pressure limit. • Only one of the systems registers the change in pressure limit. • The pressure change is registered but not taken further by the interface.				
Variable inputs:	Interface modifications (on/off functions for valves).				
Errors:	*(Encountered errors)*				
Outcome:	The pressure change is registered on both systems and shows correctly. *(Pass/fail)*				

TABLE 54 - TEST T7

7.6.1.8 COMMUNICATION

Test details					
Req. ID:	A8	Name	Communication	Test ID:	T13
Test description:	Communication between a server and two clients. Information sent to one client, will be received by the other.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Name of tester)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Start server and wait for connection • Start a client and connect to server • Send a number of commands to server to check the connection between the two. • Started another client and connected. • Send commands from both clients to see if the server will return open/close valve from both.				
Expected results/errors:	• The server will not accept connections • The server will not parse the incoming data • The server will not accept two clients connected at once • The client will not connect • The client will not be able to send data				
Variable inputs:	*Changes done in EYESIM that are needed to be registered also in System Platform, or the other way around.*				
Errors:	*(Encountered errors)*				
Outcome:	*(What happened?)*				

TABLE 55 - TEST T13

7.6.2 SECONDARY OBJECTIVES

7.6.2.1 REGULATING PRESSURE

Test details					
Req. ID:	B1-F	Name	Regulating pressure	Test ID:	T8
Test description:	The system operator should be able to set and regulate the supply pressure of tanks.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Apply a steady increasing pressure to the supply tank.				
Expected results/errors:	• The pressure cannot be regulated. • The WOCS system does not start at all.				
Variable inputs:					
Errors:	*(Encountered errors)*				
Outcome:	Supply tank pressure can be regulated and the WOCS is successfully started up. *(Pass/fail)*				

TABLE 56 - TEST T8

7.6.2.2 WOCS START-UP PROCEDURE

Test details					
Req. ID:	B2-F	Name	WOCS start-up procedure	Test ID:	T9
Test description:	The operator applies overpressure to the WOCS through Source in order to purge the tanks.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Start the WOCS system on System Platform. • Start WOCS startup. • Purge container.				
Expected results/errors:	• Purging is not applied. • The systems do not accept overpressure to be applied. • Indicator shows wrong state.				
Variable inputs:	Overpressure (on/off).				
Errors:	*(Encountered errors)*				
Outcome:	The WOCS system continues the start up procedure as normal. *(Pass/fail)* Both systems register the event and show the correct state of the WOCS system after start-up. *(Pass/fail)*				

TABLE 57 - TEST T9

7.6.2.3 B5-F: REAL TIME INFO ON VALVES IN SOURCE

Test details					
Req. ID:	B5-F	Name	Real Time info on valves in source	Test ID:	T14
Test description:	The user will point its sight on a valve which will catch information from source and get the value and the state of the valve (open/close) plus name of valve				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Start source and system platform • Connect to server. • Move the character to a valve • Point at the valve				
Expected results/errors:	• The user will not get any description about the valve at all • The user will only get the name of the valve • The user will only get the values • The user will not get the state of the valve • The user will get the wrong values				
Variable inputs:	Variables sent from System Platform State of valve (Open/close) Name of valve				
Errors:	*(Encountered errors)*				
Outcome:	All the values will be shown *(Pass/fail)*				

TABLE 58 - TEST T14

7.6.3 THIRD OBJECTIVE

7.6.3.1 RAISE/LOWER STACK

Test details					
Req. ID:	C1-F	Name	Raise/lower stack	Test ID:	T10
Test description:	The operator proceeds to the button panel for stack controller. By pressing buttons for lowering and raising the stack, check if functionality is normal.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source and System Platform. • Raise the stack from System Platform. • Lower the stack from Source by VR-buttons. • Raise the stack from System Platform by VR-buttons.				
Expected results/errors:	• Nothing happens when controls for raising/lowering the stack are applied, in both systems.				
Variable inputs:	N/A				
Errors:	*(Encountered errors)*				
Outcome:	The stack will raise and lower from both systems. *(Pass/fail)*				

TABLE 59 - TEST T10

7.6.3.2 ARTIFICIAL INTELLIGENCE – MOVEMENT OF CREW

Test details					
Req. ID:	C2-F	Name	AI – Movement of crew	Test ID:	T11
Test description:	The operator starts Source, and observes if the AI controlled crew members walk the predefined paths (their walking routes).				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source. • *(The predefined walking routes should have been installed)* • Observe if the crew members are following their predefined routes correctly.				
Expected results/errors:	• No AI crew members are visible. • The AI crew members are not moving. • Source stops working when the AI is initialized				
Variable inputs:	Path made by waypoints (preinstalled).				
Errors:	*(Encountered errors)*				
Outcome:	AI controlled crew is visible. *(Pass/fail)* AI controlled crew is not moving. *(Pass/fail)*				

TABLE 60 - TEST T11

7.6.3.3 ARTIFICIAL INTELLIGENCE – ABILITIES OF CREW

Test details					
Req. ID:	C3-F	Name	AI – Abilities of crew	Test ID:	T12
Test description:	The operator starts Source, and observes if the AI controlled crew members do their assigned tasks (open/close valves and pushing alarms) while on their predefined walking route.				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source. • *(The predefined walking routes should have been installed)* • *(The tasks should have been assigned to specific AI controlled crew members)* • Observe if the crew members are executing their specific tasks, such as turning valves and pushing alarm buttons.				
Expected results/errors:	• The AI crew members are not doing anything (standing still). • The AI crew members are not doing the correct task. • The AI crew members do not go back to their paths when a task is done.				
Variable inputs:	Path made by waypoints (preinstalled). Variable tasks for AI crew (preinstalled).				
Errors:	*(Encountered errors)*				
Outcome:	AI controlled crew is doing assigned tasks. *(Pass/fail)* AI controlled crew goes back to their predefined path when a task is completed. *(Pass/fail)*				

TABLE 61 - TEST T12

7.6.3.4 C4-F: KINECT INTEGRATION

Test details					
Req. ID:	C4-F	Name	Kinect integration	Test ID:	T15
Test description:	By connection a kinect to the system, it should be able to control valves and movement of the player				
Test date:	*(dd.mm.yyyy)*	Tester:	*(Testers will be assigned later in the project)*		
Testing method:	*(Testing approach, black box/white box)*				
Test approach:	• Run Source • Make hand gestures so the player will move				
Expected results/errors:	• No movement will be registered. • Only some movement is registered.				
Variable inputs:	Different gestures made by the user				
Errors:	*(Encountered errors)*				
Outcome:	The player will move in game. *(Pass/fail)*				

TABLE 62 - TEST T15

7.6.4 TEST OBJECTIVES

Since we have access to most of the internal data while doing this project, both a “black box and white box testing – approach” sounds like the logical way to go. Together with unit testing of each function, regression tests when we add a new function, with internal testing of the entire system in between. This to ensure the old functions working properly with the new and the entire system is cooperating.

When creating a test document, we have to be able to trace it back to the person who was responsible for the testing of a particular requirement. Together with the traceability issue, we need to know about what system has been tested at which time. A test report will also contain what values we tested and the errors which a cured during the testing period. The person, who has written the software, will not be the one to test it. This is because of the psyche of humans, not wanting to find errors in their own work

7.6.4.1 TIME USE

Taking in to consideration that we only have about 5 months to finish the project, testing and programming have to be done simultaneously, we therefore chose to spend up to three hours a week testing some software.

7.7 TEST CASES

In order to determine whether the use cases described in Requirements Specification are working correctly or not, we need to set a specific list of conditions which will be tested by a tester. This testing method is also known as test cases. They will help us keep the same quality as the rest of the project. We have covered each use case with a corresponding test case in a table template which can be seen below (table 2).

Steps	Description	Expected Result	Expected Errors	Comments
1.				
2.				

TABLE 63 - TEST CASE TABLE TEMPLATE

7.7.1 TEST CASES IN DETAIL

7.7.1.1 TEST CASE 1 – ACTIVATE EMERGENCY SHUTDOWN

Use case origin: UC1 – Activate Emergency Shutdown

Description: This test case will test whether an emergency shutdown has been executed after the system operator has initiated the shutdown.

Predecessors: Source and System Platform must be preinstalled. The simulated environment requires the trainee to activate the emergency shutdown in any of the systems while executing the test.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Activate the ESD from any of the systems.	ESD starts and the stack is disconnected from the rig. The alarm is hearable and visible.	Nothing happens. Stack doesn't disconnect, but the alarm is on (or vice versa).	
2.	Go back to step 1 and repeat the process from the software not yet tested.			

TABLE 64 - TEST CASE 1

7.7.1.2 TEST CASE 2 – INITIATE ALARM

Use case origin: UC2 – Initiate Alarm

Description: This test case will test whether an alarm is ON or OFF after it was initiated.

Predecessors: Source and System Platform must be preinstalled. The simulated environment requires the trainee to activate an alarm in Source while executing the test.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Activate an alarm by pushing a virtual button in Source.	The alarm is visible and hearable in both systems.	Nothing happens when alarm is initiated.	There could be situations where alarm is visible and not hearable (or vice versa).
2.	Go back to step 1 and repeat the process for all other manual alarms on the rig.			

TABLE 65 - TEST CASE 2

7.7.1.3 TEST CASE 3 – EVACUATE TO DESIGNATED ZONE

Use case origin: UC3 – Evacuate to designated zone

Description: This test case will test if the operator has evacuated to the nearest safety zone.

Predecessors: Source and System Platform must be preinstalled. The simulated environment requires the trainee to activate ESD in any of the systems while executing the test.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Activate the ESD from any of the systems.	ESD starts and the stack is disconnected from the rig. The alarm is hearable and visible.	Nothing happens. Stack doesn't disconnect, but the alarm is on (or vice versa).	
2.	Move to the nearest safety zone.	Operator has evacuated and the mission is complete.	Something happens that hinders the operator from getting into the zone.	There can be multiple zones.
3.	Go back to step 1 and repeat until all safety zones have been tested.			

TABLE 66 - TEST CASE 3

7.7.1.4 TEST CASE 4 – START HPU

Use case origin: UC4 – Start HPU

Description: This test case will test if the HPU has been started.

Predecessors: Source and System Platform must be preinstalled.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Start Source and System Platform	Source and System Platform start up on their own PC's		Source runs on its own PC, while System Platform runs on its own PC and a virtual machine on that same PC
2.	Start the mission: "HPU start-up"	A task list shows up instructing the user to advance to the next step in the start-up progress	Nothing happens. Task list bugs	
3.	Purge container	Container purged, and the filter clog message appears	Task list bug.	
4.	Change filter	Filter changed, next step is to activate valve 19NSCircPump	Nothing happens in System Platform	
5.	Turn 19NSCircPump ON	When valve is turned on, next valve 19NSCircPumpAuto is to be activated	Nothing happens in System Platform	
6.	Turn 19NSCircPumpAuto ON	This is the last item in the sequence of activation before HPU mission is finished	Nothing happens in System Platform	

TABLE 67 - TEST CASE 4

7.7.1.5 TEST CASE 5 – CHANGE CLOGGED FILTER

Use case origin: UC5 – Change clogged filter

Description: This test case will test if the operator has successfully changed the clogged filter.

Predecessors: Source and System Platform must be preinstalled. A filter must be clogged.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Change the filter.	There is no hydraulic fluid in the pipe. CFA is shown on System Platform.	Filter cannot be changed.	
2.	Put filter in its default position and resume operation.	Everything runs normally. The task list continues.	The clogged filter alarm still shows a clogged filter.	

TABLE 68 - TEST CASE 5

7.7.1.6 TEST CASE 6 – OBSERVE STATUS PANEL

Use case origin: UC6 – Observe status panel

Description: This test case will test if the operator has successfully activated a HMI window inside Source.

Predecessors: Source and System Platform must be preinstalled. A mission in Source has to be started before the test can be executed.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Activate status panel for the observed object in Source.	A virtual window shows up with information about the observed object.	Nothing happens. Flickering window.	There can be many unforeseen faults here.

TABLE 69 - TEST CASE 6

7.7.1.7 TEST CASE 7 – SET VALVE ALARM THRESHOLD

Use case origin: UC7 – Set valve alarm threshold

Description: This test case will test if a new value for alarm threshold has been set on a valve.

Predecessors: Source and System Platform must be preinstalled. A mission in Source has to be started before the test can be executed.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Operator sets a new value for the alarm threshold.	A virtual window shows up with information about the new value and states it has been set.	Nothing happens. Flickering window. Old value still shows.	There can be many unforeseen faults here.

TABLE 70 - TEST CASE 7

7.7.1.8 TEST CASE 8 – START WOCS

Use case origin: UC8 – Start WOCS

Description: This test case will test if the WOCS system has been started up.

Predecessors: Source and System Platform must be preinstalled.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Operator sets a pressure value.	A virtual window shows up with information about the new value and states it has been set.	Nothing happens. Flickering window. Old value still shows.	
2.	New value is regulated by the operator.	The new value gets adjusted to what the operator wishes for.	Old value still shows. New value not being able to set.	
3.	An overpressure is applied by the operator.	Tank is purged and ready for use.	No overpressure available.	

TABLE 71 - TEST CASE 8

7.7.1.9 TEST CASE 9 – RAISE/LOWER STACK

Use case origin: UC9 – Raise/lower stack

Description: This test case will test if the stack has been raised and lowered after appropriate commands have been executed.

Predecessors: Source and System Platform must be preinstalled.

Steps	Description	Expected Result	Expected Errors	Comments
1.	Operator pushes the button for lowering the stack.	Stack lowers itself beneath the rig.	Nothing happens.	
2.	Go back to step 1 and repeat the process, this time with raising the stack.			

TABLE 72 - TEST CASE 9

8 Test strategy

8.1 PURPOSE OF THIS DOCUMENT

The purpose of this document is to define some testing methods we will use during the different stages of this project, also to give the reader insight in our testing methods.

8.2 AUTHOR

Torjus Engell

8.3 IN CHARGE OF ACTIVITY

Torjus Engell

8.4 IDENTIFYING TEST TYPES

There are different kinds of testing types. We have found some test strategies we thought were useful and choose to use them as testing methods.

8.4.1 BLACK BOX TESTING

What is a Black Box Testing Strategy?

Black Box Testing is not a type of testing; it instead is a testing strategy, which does not need any knowledge of internal design or code etc. As the name "black box" suggests, no knowledge of internal logic or code structure is required. The types of testing under this strategy are totally based/focused on the testing for requirements and functionality of the work product/software application. Black box testing is sometimes also called as "Opaque Testing", "Functional/Behavioral Testing" and "Closed Box Testing".

The base of the Black box testing strategy lies in the selection of appropriate data as per functionality and testing it against the functional specifications in order to check for normal and abnormal behavior of the system. Now a days, it is becoming common to route the Testing work to a third party as the developer of the system knows too much of the internal logic and coding of the system, which makes it unfit to test the application by the developer.

In order to implement Black Box Testing Strategy, the tester is needed to be thorough with the requirement specifications of the system and as a user, should know, how the system should behave in response to the particular action.

Various testing types that fall under the Black Box Testing strategy are: functional testing, stress testing, recovery testing, volume testing, User Acceptance Testing (also known as UAT), system testing, Sanity or Smoke testing, load testing, Usability testing, Exploratory testing, ad-hoc testing, alpha testing, beta testing etc.

These testing types are again divided in two groups: a) Testing in which user plays a role of tester and b) User is not required.

8.4.2 WHITE BOX TESTING

White box testing strategy deals with the internal logic and structure of the code. White box testing is also called as glass, structural, open box or clear box testing. The tests written based on the white box testing strategy incorporate coverage of the code written, branches, paths, statements and internal logic of the code etc.

In order to implement white box testing, the tester has to deal with the code and hence is needed to possess knowledge of coding and logic i.e. internal working of the code. White box test also needs the tester to look into the code and find out which unit/statement/chunk of the code is malfunctioning.

Advantages of White box testing are:
i) As the knowledge of internal coding structure is prerequisite, it becomes very easy to find out which type of input/data can help in testing the application effectively.
Ii) the other advantage of white box testing is that it helps in optimizing the code
iii) it helps in removing the extra lines of code, which can bring in hidden defects.

Disadvantages of white box testing are:
i) As knowledge of code and internal structure is a prerequisite, a skilled tester is needed to carry out this type of testing, which increases the cost.
ii) And it is nearly impossible to look into every bit of code to find out hidden errors, which may create problems, resulting in failure of the application.

8.4.3 FUNCTIONAL TESTING

In this test, the software is tested for the functional requirements. The test are written in order to check if the application behaves as expected

8.4.4 AD-HOC

This is a test done without any formal Test Plan or Test case creation. The Ad-Hoc testing helps users learn the application prior to starting any other testing procedures, this helps in deciding duration and scopes to various testing.

8.4.5 VOLUME

Volume testing is done against the efficiency of the application. Huge amount of data is processed through the application being tested, in order to check the extreme limitations of the system.

8.4.6 STRESS

The application is tested against heavy load such as complex numerical values, large number of inputs, large number of queries etc. which checks for the stress/load the applications can withstand.

8.4.7 UNIT TESTING

The developer carries out unit testing in order to check if the particular module or unit of code is working fine. The Unit Testing comes at the very basic level as it is carried out as and when the unit of the code is developed or a particular functionality is built.

8.4.8 STATIC AND DYNAMIC TESTING

Static analysis involves going through the code in order to find out any possible defect in the code. Dynamic analysis involves executing the code and analyzing the output.

8.4.9 STATEMENT COVERAGE

In this type of testing the code is executed in such a manner that every statement of the application is executed at least once. It helps in assuring that all the statements execute without any side effect.

8.4.10 BRANCH COVERAGE

No software application can be written in a continuous mode of coding, at some point we need to branch out the code in order to perform a particular functionality. Branch coverage testing helps in validating of all the branches in the code and making sure that no branching leads to abnormal behavior of the application.

8.4.11 MUTATION TESTING

A kind of testing in which, the application is tested for the code that was modified after fixing a particular bug/defect. It also helps in finding out which code and which strategy of coding can help in developing the functionality effectively.

8.4.12 REGRESSION TESTING

Regression testing is done to ensure that something that was previously working, still works. The regression testing is doesn't focus on new functionality, but on functionality that was previously delivered and tested. As we add new functionality to the program, a test to ensure that noting previously delivered got broken during the merging. Defects of this type include:

- Existing uncaught defects that show up with the integration of new code
- Defects preciously fixed that reappear in the new release.
- Defects in previous functionality introduced as part of the creation of new functionality.

Regression tests are generally identified from previously run tests. Common ways to regression test include performing all previously run tests, focusing on the reemergence of previously found bugs and running a subset of previously run test focusing on critical functionality.

Besides all the testing types given above, there are some more types which fall under both Black box and White box testing strategies such as: Functional testing (which deals with the code in order to check its functional performance), Incremental integration testing (which deals with the testing of newly added code in the application), Performance and Load testing (which helps in finding out how the particular code manages resources and give performance) etc.

9 System analysis

9.1 PURPOSE OF THIS DOCUMENT

This document will cover the analysis part and extensions based on use-cases made for our system in the requirements specification document. All use-cases will be thoroughly defined. Further on, analysis class for the system will be made. This will help understand the connections between objects and their communication with each other. This document will also be a good stepping stone for the upcoming design phase of the project. Please see the document “Introduction to UML” for a brief explanation and introduction to UML diagrams.

9.2 AUTHOR

Dejan Vukobratovic

9.3 IN CHARGE OF ACTIVITY

Dejan Vukobratovic

9.4 ANALYSIS CLASSES

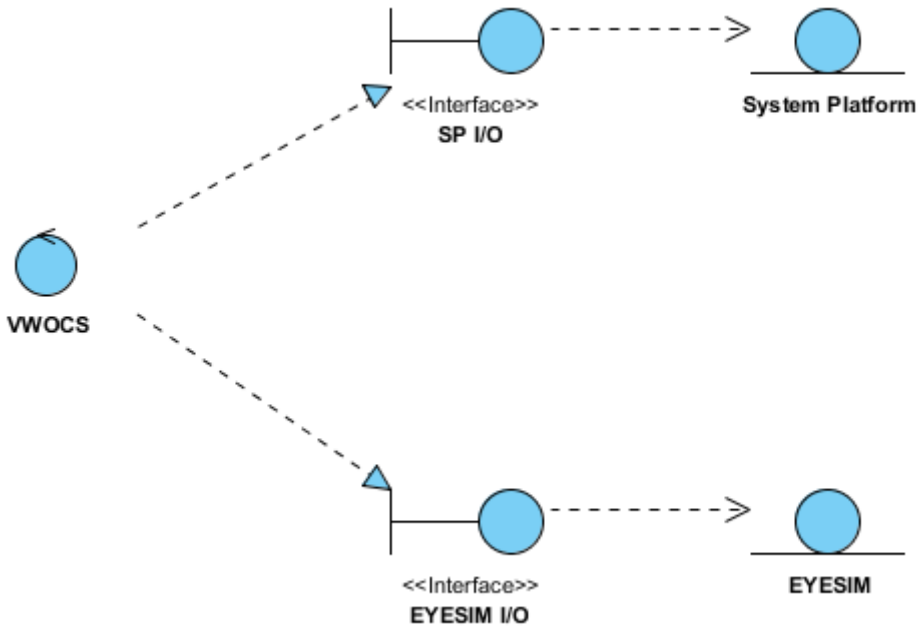

FIGURE 8 - ANALYSIS CLASS DIAGRAM

In order for EYESIM and System Platform to communicate with each other, we need to implement several interfaces. Figure 2 shows a rough overview of the system.

If unfamiliar with the symbols above, please take a short look in the technology document "Introduction to UML", the chapter about Analysis Classes.

System Platform and EYESIM are illustrated as databases where the data will be stored and changed frequently by both systems.

Further on, we have set up EYESIM I/O and SP I/O (System Platform) interfaces for handling the inputs and output for both systems.

In the left end we have the VWOCS control class (the interpreter), which represents the core of our planned interface. The interfaces above are meant to cooperate with each other through the VWOCS interpreter.

When it comes to functionality, let us look at the following example:

A valve V-21 has been closed in EYESIM. EYESIM I/O interface snatches this change and calls appropriate functions in VWOCS which in return tells the SP I/O to close off the same valve V-21 in System Platform.

9.5 USE-CASE ANALYSIS

Let's take a look on the functionality overview as described in the requirements document:

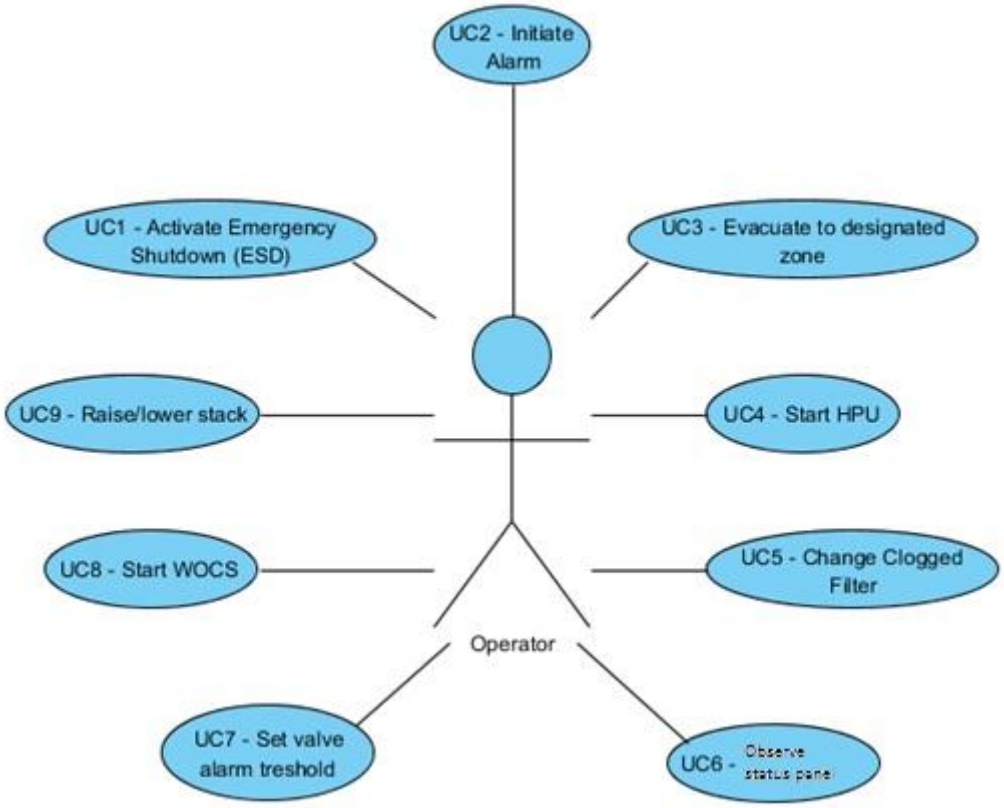

FIGURE 9 - EXTENDED USE CASE DIAGRAM (OPERATOR)

A system operator can do many things, illustrated as use-cases.

Based on the use-case diagram above, we will now define functionality for each use case a bit further than it was explained in the requirements document and introduce detailed steps of each use case and how they can be implemented.

Since Figure 2 shows that there won't be many classes involved in the interface, we will need to focus on function declarations for every little thing such as changing valve status and alarm status. This will be introduced further in the project, in the Design Document (class implementation and description) and Mission Description Document (procedures and descriptions of missions and objectives for trainees).

9.5.1 UC1 – ACTIVATE EMERGENCY SHUTDOWN (ESD)

This use-case illustrates system functionality allowing a trainee to activate the ESD at any given time during a mission. This will disconnect the stack from the rig and raise an alarm. The ESD can be activated from both EYESIM and System Platform, and likewise, give same alarm indications in both systems. In reality, this shutdown action is extremely expensive. However, in order to maximize crew security and minimize the risks of injury it is necessary to have the functionality in place.

Specific events which need to be handled exquisitely are stack and alarm. The stack disconnection needs to be modeled and animated in a way so that it looks great, visually.

The alarm solution has an interesting option. In addition to alarms in EYESIM and System Platform, we can put a real spinning beacon light connected to an USB port. This will be a great addition to our last presentation which will focus on the product itself.

9.5.2 UC2 – INITIATE ALARM

A trainee must be able to initiate an alarm manually when something unexpected happens. This is done by pressing one of the manual alarm buttons placed throughout the rig. As with UC1, the main focus here is of course crew safety and minimizing the occurrence of for example accidents, leaks and emergency.

The alarms should be visible in both systems of course. To make the training simulation somewhat real, we will need to make sounds for the alarm in EYESIM. Since the alarm can then be heard and seen, it will be easier for trainees to react to it.

9.5.3 UC3 – EVACUATION TO DESIGNATED ZONE

This use-case concerns EYESIM only. As with the two preceding use-cases, this one will focus on crew safety yet again. When an alarm has been raised all workers on the rig should evacuate to the nearest safety zone (both operator and AI-controlled crew). Leaks in pipes (high pressure) or eventual fires could easily threaten lives of men, so evacuating to designated zones is an important part of safety procedures.

There will be several designated zones on the rig (number is not decided yet, this will be done in the design and implementation phase of the project). Depending on the current position of the trainee on the rig, he will need to refer to the closest zone, and this applies to AI controlled crew as well.

9.5.4 UC4 – START HPU

This use-case illustrates functionality allowing a trainee to start the HPU which transfers hydraulic oil from return tank to supply tank for the WOCS system. This action should be available in both EYESIM and System Platform. Primarily this system function involves extensive control. However, we decided that we will simplify it by using simple buttons for start-up in combination with some valve controlling (in order to stabilize pressure).

9.5.5 UC5 - CHANGE CLOGGED FILTER

The filter itself is quite simple. Figure 4 shows a schematic of how the filter will look like in EYESIM.

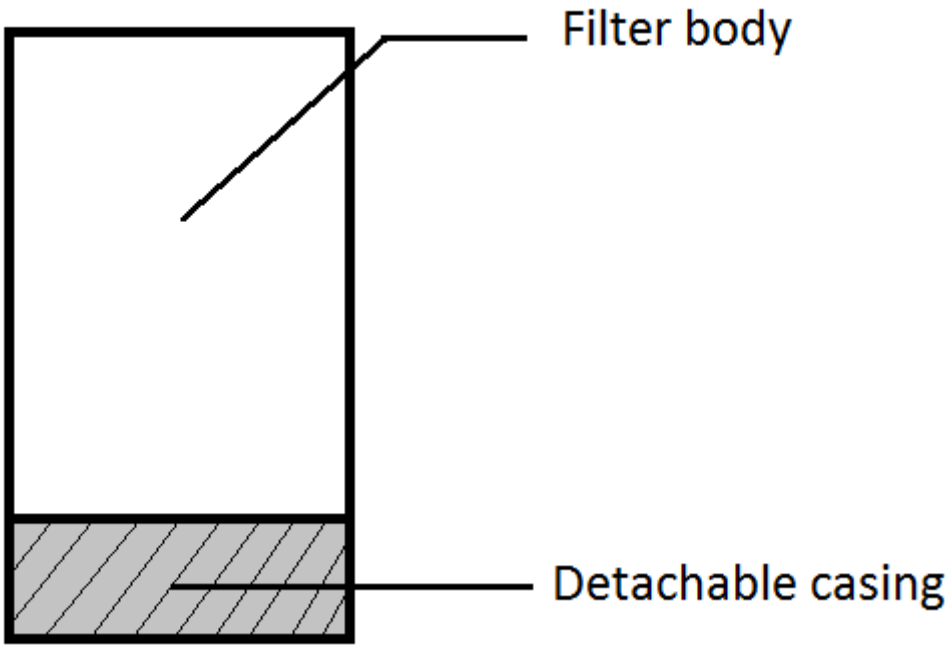

FIGURE 10 - FILTER

First we have the filter body, and in the end we have the casing. When detached, it reveals the filter which can then be changed when it is clogged. An alarm will show up when the filter is clogged which indicates that a trainee needs to change it. The alarm will then

9.5.6 UC6 - OBSERVE HMI

While system information is easily obtained in System Platform, EYESIM has introduced another dimension – the virtual window. The HMI in System Platform is designed to show and allow control of all physical processes in an industrial environment. We wish to take this a step further and allow the trainee to choose an object in EYESIM (a valve or a pipe/tank) and then bring up an “in-game” menu. This menu contains information about the valve such as name, type, tags, etc. In addition, instead of using manual gauges and meters that are modeled with moving parts such as the needle in a manometer, we will make a digital version which shows up in the HMI virtual window in EYESIM. This will allow all information to be displayed at the same time, at the same place (same window).

9.5.7 UC7 - SET VALVE ALARM THRESHOLD

Another valuable function is for a trainee to set alarm boundaries. If pressure in a tank is too high, it would be nice to hear and see an alarm which indicates this. Now, if the trainee can set the alarm threshold manually, the system would be configurable from EYESIM. This is what we will focus on within this functionality. We want a trainee to be able to set thresholds for valves primarily. If the pressure limit is exceeded, it will sound an alarm and the trainee can then shut off or relieve the supply by redirecting the flow. There are many possibilities here, and we will need to limit them and choose a handful we wish to implement.

9.5.8 UC8 – START WOCS

This is a complicated system function. The real WOCS start-up takes time and requires a lot of flow and pressure control. We wish to simplify this start-up by making it a push-button in combination with some flow control performed by the trainee. A detailed step-by-step description will be made in the mission document later.

9.5.9 UC9 – RAISE/LOWER STACK

As with UC8, this function is complicated as well. To simplify it, we will start with making two buttons, where one is for raising and the other for lowering the stack. Some sounds and animations will have to be made in order to produce a visual “WOW – factor”.

10 Design document

10.1 PURPOSE OF THIS DOCUMENT

Provide a full description of how the system will function and look. It has to be detailed enough so that it can be used in the development process.

10.2 AUTHOR

Jan Hansen

10.3 IN CHARGE OF ACTIVITY

Jan Hansen

10.4 DIAGRAMS

10.4.1

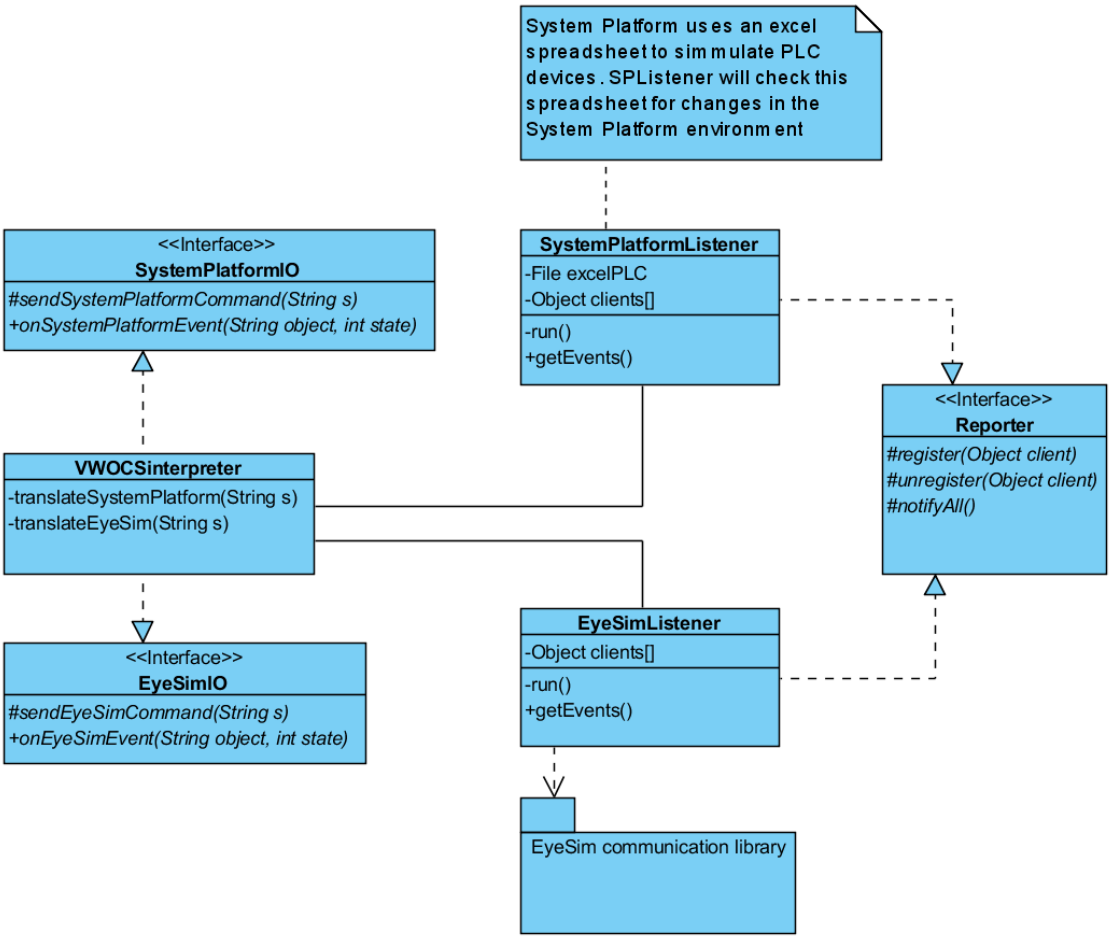

FIGURE 11 - DESIGN CLASS DIAGRAM

10.4.2 CLASS DESCRIPTIONS

10.4.2.1.1 VWOCSINTERPRETER

The class is to interpret events in EyeSim and send it forward to System Platform in a format it can understand. It will also interpret events in System Platform and forward them to EyeSim in the right format.

Attributes	**Attribute descriptions**
-	-
Functions	**Function descriptions**
`private function translateSystemPlatform(String s)`	Translate events from System Platform to the right format.
`private function translateEyeSim(String s)`	Translate event from EyeSim to the right format.
Interfaces	
`SystemPlatformIO`	
`EyeSimIO`	

TABLE 73 - VWOCS INTERPRETER CLASS DESCRIPTION

10.4.2.1.2 SYSTEMPLATFORMLISTENER

This class checks an excel spreadsheet regularly for changes in the System Platform environment. When it finds a change in the spreadsheet it will notify all registered objects that a change has occurred. The objects that register to this class have to contain the function `onSystemPlatformEvent(String object, int state)`.

Attributes	**Attribute descriptions**
`private File excelPLC`	Reference to excel spreadsheet.
`private Object clients[]`	Objects registered to the listener.
Functions	**Function descriptions**
`private function run()`	Loop that runs in a separate thread checking for changes in the spreadsheet regularly and notifies clients of changes.
`private function getEvents()`	Checks for changes in the spreadsheet.
Interfaces	
`Reporter`	

TABLE 74 - SYSTEM PLATFORM CLASS DESCRIPTION

10.4.2.1.3 EYESIMLISTENER

This class regularly checks for changes in the EyeSim environment by using a communication library in EyeSim. When it finds a change it will notify all registered objects that a change has occurred. The objects that register to this class have to contain the function `onEyeSimEvent(String object, int state)`.

Attributes	**Attribute descriptions**
`private Object clients[]`	Objects registered to the listener.
Functions	**Function descriptions**
`private function run()`	Loop that runs in a separate Thread checking for changes in the EyeSim environment through a communication library in EyeSim.
`private function getEvents()`	Check for new events in EyeSim.
Interfaces	
`Reporter`	

TABLE 75 - EYESIM LISTENER CLASS DESCRIPTION

10.4.2.2 INTERFACE DESCRIPTIONS

10.4.2.2.1 SYSTEMPLATFORMIO

This interface contains virtual functions for communicating with System Platform.

Virtual functions	Virtual function descriptions
`private virtual function sendSystemPlatformCommand(String s)`	Send command to System Platform.
`public virtual function onSystemPlatformEvent(String object, int state)`	This function runs when an event has happened in System Platform (The excel spreadsheet has changed).

TABLE 76 - SYSTEM PLATFORM IO INTERFACE DESCRIPTION

10.4.2.2.2 EYESIMIO

This interface contains virtual functions for communicating with EyeSim.

Virtual functions	Virtual function descriptions
`private virtual function sendEyeSimCommand(String s)`	Send command to EyeSim.
`public virtual function onEyeSimEvent(String object, int state)`	This function runs when an event has happened in EyeSim.

TABLE 77 - EYESIM IO INTERFACE DESCRIPTION

10.4.2.2.3 REPORTER

This interface contains virtual functions for implementing an observer pattern.

Virtual functions	Virtual function descriptions
`public virtual function register(Object client)`	Registers clients to the listener.
`public virtual function unregister(Object client)`	Unregisters clients to the listener.
`Private virtual function notifyAll()`	Notifies all the clients.

TABLE 78 - REPORTER INTERFACE DESCRIPTION

10.4.3 SEQUENCE DIAGRAMS

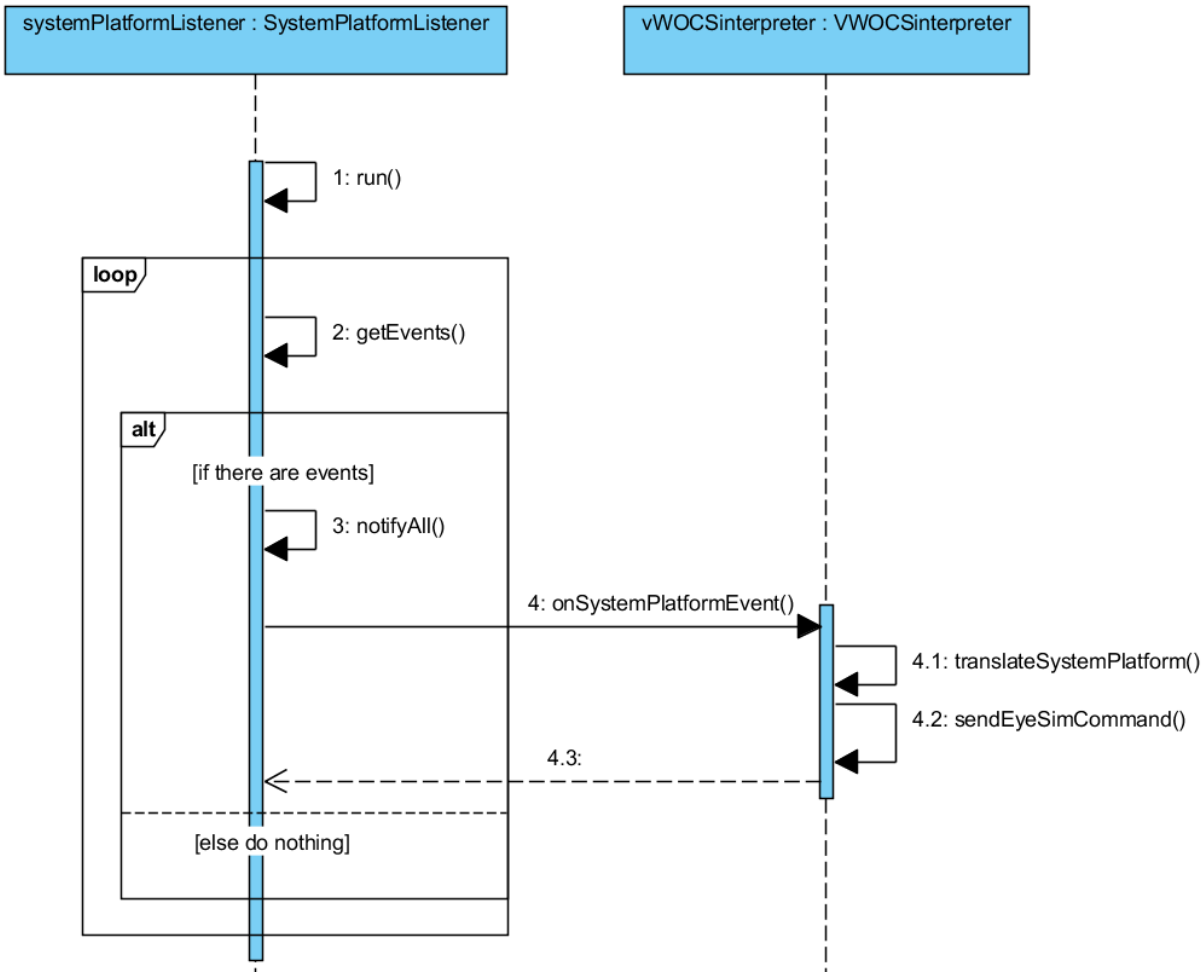

FIGURE 12 - SEQUENCE DIAGRAM SYSTEM PLATFORM

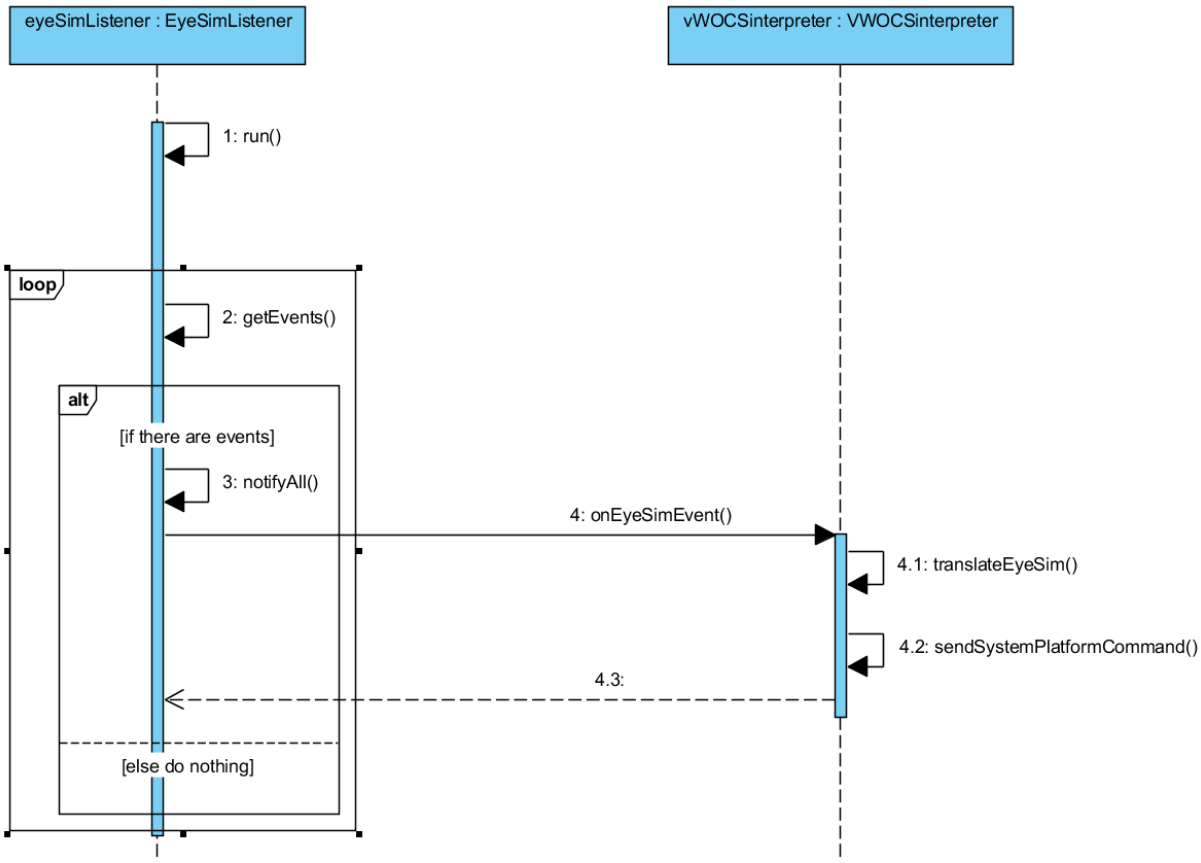

FIGURE 13 - SEQUENCE DIAGRAM EYESIM

10.4.4 ACTIVITY DIAGRAMS

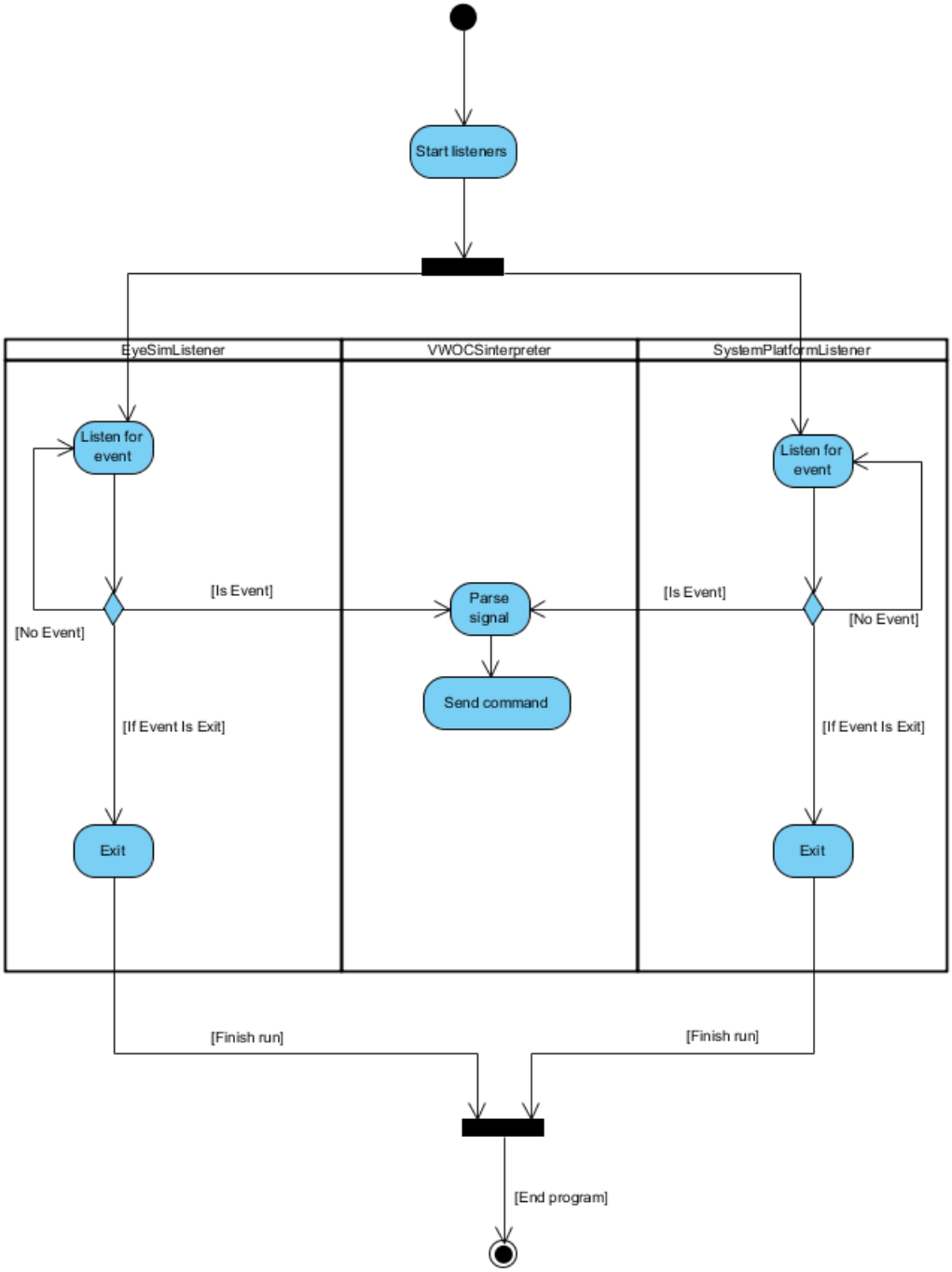

FIGURE 14 - ACTIVITY DIAGRAM

10.4.5 STATE DIAGRAMS

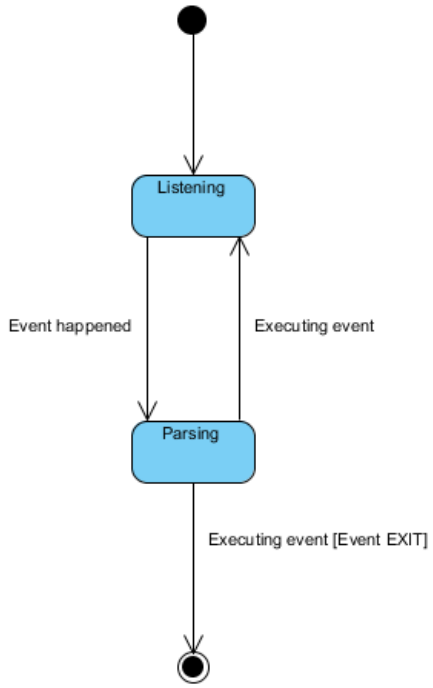

FIGURE 15 - STATE MACHINE DIAGRAM

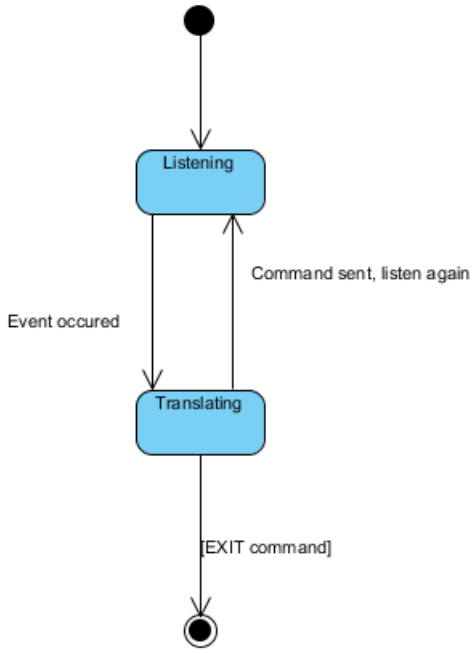

FIGURE 16 - STATE MACHINE DIAGRAM 2

10.5 MISSIONS

10.5.1 MISSION LIST

- Evacuation training
- HPU – Start pump
- Change clogged filter
- WOCS start-up
- Raise/lower stack

10.5.2 MISSION DESCRIPTIONS

Evacuation training	
Find the right evacuation zone in a given time limit. The right zone will depend on where you are on the platform.	
Criteria for success	**Criteria for failure**
1. React to the alarm. 2. Find out where the nearest evacuation zone is. 3. Head towards the zone.	• Time limit runs out. • Fall off platform. • Run into the wrong evacuation zone.
Requirement category	**A**

TABLE 79 - MISSION: EVACUATION TRAINING

HPU – start pump	
Start the hydraulic power unit pump.	
Criteria for success	**Criteria for failure**
1. Approach the button. 2. Push the button.	• Run out of the mission area.
Requirement category	**A**

TABLE 80 - MISSION: HPU START PUMP

Change clogged filter	
Change the clogged filter. You will be measured in time used to complete the mission.	
Criteria for success	**Criteria for failure**
1. Secure the casing. 2. Open casing. 3. Take out clogged filter. 4. Insert new filter. 5. Start system again.	6. Run out of the mission area.
Requirement category	**A**

TABLE 81 - MISSION: CHANGE CLOGGED FILTER

WOCS start-up	
Start up a WOCS system.	
Criteria for success	**Criteria for failure**
1. Purge the WOCS container by applying overpressure. 2. Set and regulate the supply pressure of the tanks during the start-up.	• Pressure exceeds 700 bars or goes below 100 bars. • Run out of the mission area.
Requirement category	**B**

TABLE 82 - MISSION: WOCS START-UP

Raise/lower stack	
Raising or lowering the stack on the platform.	
Criteria for success	**Criteria for failure**
1. Push button to raise/lower stack.	• Run out of the mission area.
Requirement category	**C**

TABLE 83 - MISSION: RAISE/LOWER STACK

10.6 OBJECTS

10.6.1 OBJECTS LIST

- Container
- Valve
- Pipe
- Computer display
- Keyboard
- Mouse
- Ceiling lamp
- Manometer
- Hydraulic pressure unit (HPU)
- Filter
- Filter casing

10.6.2 OBJECT DESCRIPTIONS

The appearance of objects will be decided when constructed in the 3d modeling tool.

10.7 3D MODELS AND ANIMATION

In a project like this, 3D models are a natural thing to either make or acquire. We chose to make most of our 3D models linked to the container. We therefore also found the need for some animations linked to the models we created. Keeping this in mind, only a few of the models we created will be active or available for use and there are these models we want to animate. These models are buttons, doors, handles and a filter.

Buttons
Depending on the button, something will either start or shut down.
Animation
When button is used, it will move inwards light up and also move outwards till original position again. Depending on the function of the button, different lights will be used.

TABLE 84 - ANIMATION BUTTONS

Doors
The doors will have a collision box, and therefore needs to be opened.
Animation
When the door is used, the door knob will move down, the door will open and the door knob will be released. Same thing when closing the door.

TABLE 85 – ANIMATION DOORS

Filter
The filter has to be replaced some time during the simulation
Animation
When the filter needs replacing, the animation will show the user that the filter us releasing from its position, to be replaced by a new filter which will be put back where the original filter was

TABLE 86 – ANIMATION FILTER

Handles
Depending on which handle you turn, an action will take place
Animation
When the user wants to turn a handle and perform an action, he will see that the handle is turning.

TABLE 87 - ANIMATION HANDLES

11 Implementation document

11.1 PURPOSE OF THIS DOCUMENT

This document is written to give an overview over how the implementation has been done, in comparison to the design document. This is sort of an update of the design document, where we describe the stable architecture of the system. One can use this document to specialize oneself in our system.

11.2 AUTHOR

Dejan Vukobratovic

Leif H. Larsen

11.3 IN CHARGE OF ACTIVITY

Leif H. Larsen

11.4 INTRODUCTION TO IMPLEMENTATION

11.4.1 GENERAL

The basis of the implementation is what we established in the design document. As it will be clear from this document, we have made some changes from the original design, but these are necessary to get the stable architecture. The intended structure is however still the same.

We did not receive EYESIM within the given deadline, so therefore we chose to build the rest of the system using the engine from our prototype, Source.

11.4.2 UML/API

Throughout this document we will refer to UML diagrams, which describe the system better. These diagrams are as one will see different than our earliest diagrams, but these are the finished diagrams. We will also refer to the API created, instead of describing everything in detail.

11.4.3 DEVELOPMENT SOFTWARE

For the prototype the following development software have been used:

- Hammer editor – Default map editor for source projects
- Visual Studio 2010 – C++/C# editor
- Netbeans – Java editor
- 3D studio max – 3D model editor

11.5 SYSTEM AS A WHOLE

The system can be deployed on either one computer or two computers. If it's deployed on one computer you have a virtual machine, which is the host of System Platform, with its simulator and receiving client. On the physical machine you run the VWOCS server, as well as the virtual environment, Source.

The following figure describes this deployment.

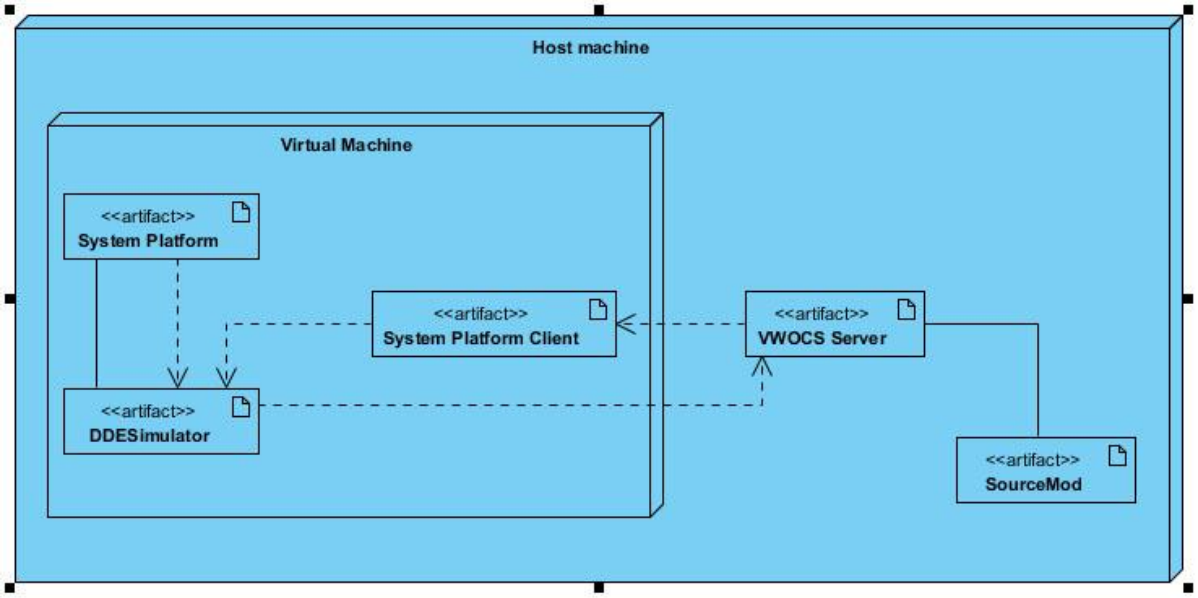

FIGURE 17 - DEPLOYMENT DIAGRAM ONE MACHINE

Seeing as both the virtual machine and the virtual environment use a lot of the CPU and GPU, it can be an advantage if we split this up to two computers. By doing this, we are able to separate the control system from the graphics, and thus being able to connect other clients as well. This may be for instance an instructor wanting to change scenarios etc.

The way this is done is by starting our Java server on one machine, and starting up the virtual machine on the same machine. On another machine we run the virtual environment, which connects to our server.

The following figure describes this deployment.

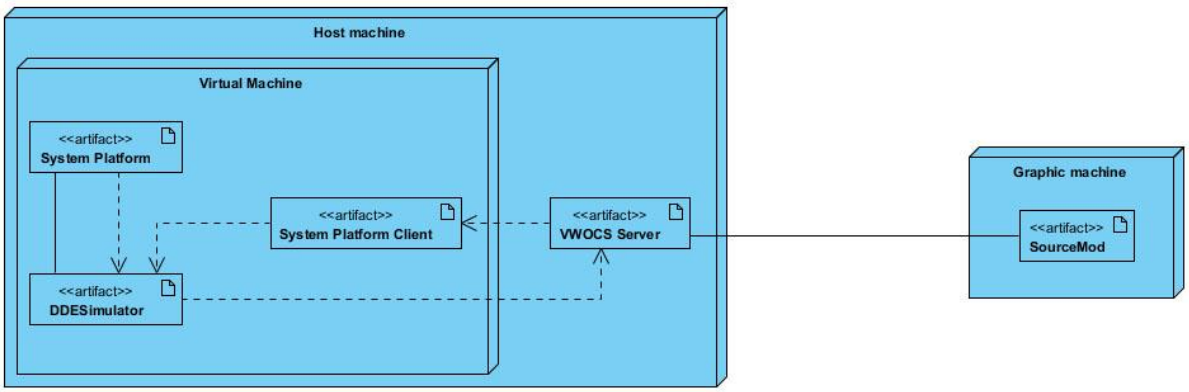

FIGURE 18 - DEPLOYMENT DIAGRAM TWO MACHINES

The VWOCS client has been integrated into the Source. This is a requirement from Source, because it requires all additional modifications (all *.cpp* and *.h* files) to add the base-file called "*cbase.h*". This does not pose any trouble at all, since all we do is add include "*cbase.h*" clause at top of each of our files. The rest of our code can then be interacting with Source through its entity-based system which is explained in the technology document "Source SDK".

When something happens in Source (e.g. a valve is opened), the VWOCS client will send info to VWOCS server which is required to execute the same action on System Platform. The server will pick up the object identity and the action to be performed, and forward it to the other VWOCS client on System Platform. The client updates the DDE Simulator and the change is picked up by System Platform and updated inside the Control-system. The same procedure applies the other way around as well.

We decided to use Berkeley sockets (*ws2_32*) library for Windows to communicate via network. This is a standard networking library for Windows, and works well for our purpose.

11.5.1 TECHNOLOGY

To be able to communicate with server and clients, we choose to use sockets. Sockets make it easy to create network-enabled programs, without having to construct network connections from scratch. They also work across different platforms such as Java, C++ and other languages.

11.5.2 PROTOCOL

As for our protocol, we send everything as strings to the java server. However, because of the way sockets are built with C++, we need to parse the strings, which are to be sent to the recipient in Source, as it can only receive chars. Further on we need to cast the input string to a char array, so it can be received properly.

Command	Expected value	Description
RunShutDown_NWL	0/ 1	ESD is triggered in Source

TABLE 88 - EXAMPLE OF PROTOCOL

Our protocol is built on the principles seen above. You have a command, which is sent as a string, with the given value after a comma. So if we trigger an ESD from Source, we send "RunShutDown_NWL, 1" which will be parsed and start the shutdown sequence in System Platform. All the lines under "Command" in our Transmission Protocol document (where you can find a complete list of protocols) have names, based on the element in System Platform which is either the sender or receiver.

In Source, we are receiving several sensor values, which are displayed on certain valves. Since these values are to be connected to different valves, we are in the need to parse all the input, and make sure that proper variables are set. Since all the incoming values changes from time to time, we need to parse the input and split the string. The delimiter here being ",". When the parsing is done, we have the given value in a global variable, which is accessible from the code which creates the GUI for information for each valve.

11.5.3 PORTABILITY

Since we had to choose our backup graphic engine, Source, we needed to write our code, so it may be easily converted to fit with EYESIMs architecture. Without having the possibility to test this, this was a case of optimizing the code, and preparing it as good as possible for any other engines.

11.6 SERVER

Since we needed two clients, there was a need for a server which could handle multiple connections. The easiest way to achieve this is to create a server, which starts a new thread for each connection. Since threads in C++ are somewhat difficult to create, we chose to code the server in java. *Please refer to the technology document "Java VS C++" for more information regarding this choice*

For the complete class diagram of the server, please refer to Server class diagram.jpg

For the complete sequence diagram of the server, please refer to Server sequence diagram.jpg

When the server starts up it binds to port 8080, and starts a socket. Further on it is listening for client connections, and when a client connects it creates a new object for the client. Also the server adds this client to a client array, so that it can send messages to every client. Then it opens the clients I/O stream, and starts the thread.

The thread that is started will immediately go to an infinite while loop, which reads lines, sent from clients. If a line is received, it will process this by calling a function in the server thread, which sends the message to other clients. To make sure the right message is sent, we have some predefined char arrays that actually are being sent. The line the server received is being parsed, to decide which of the predefined arrays to send.

To get further information regarding the server, please refer to Server API.

11.7 CLIENTS

11.7.1 SOURCE CLIENT

Source SDK has been released in several versions until now. We use the latest available version (Base 2007) because the most recent version (Base 2009) is not 100% ready for distribution via Steam.

All objects in Source engine are called entities. These entities can have several attributes and behaviors set up pre-compile time. When something happens to an entity (e.g. button pushed), there are functions in the source code for the engine which will be executed. There is a number of functions available (onAction, onPickup, onTouch, onChange etc.). Inside the function description is the place we put our code which extends the Source engine and makes it communicate with System Platform. That way, when something happens with our object, the output we added will fire and execute at the same time.

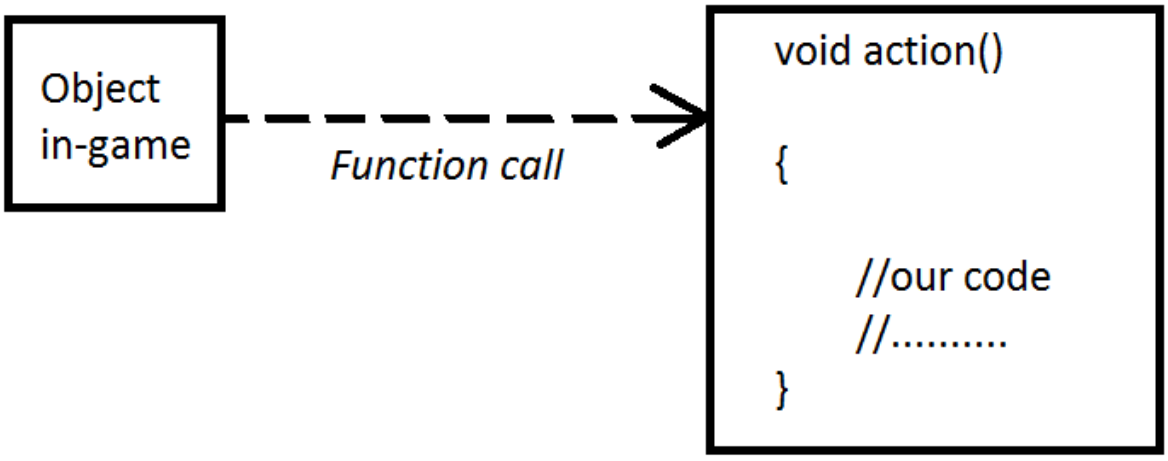

FIGURE 19 - FUNCTION CALL SOURCE

The function call will send a message to the Java server via our client code, which we have embedded into the Source code.

For a complete class diagram, please refer to Source Class diagram.jpg

For a complete sequence diagram, please refer to Source sequence diagram.jpg

As we can see from the class diagrams, we use sockets for this client. Before the source project starts, we need to make sure the server is running. When the source project starts

up, our code connects to the server. Further on a thread starts, which starts a while loop for sending. Whenever an action happens to the button a method is called, to send the appropriate string to the server.

For get further information regarding the Source client, please refer to the Source API [7]

11.7.2 SYSTEM PLATFORM CLIENT

For System Platform, the communication is quite different than the one we looked at previously for Source.

As one can tell from the deployment diagram, above, there are two pieces of client code which is needed for the System Platform bit to work properly.

System Platform uses an own simulator program for running inputs and outputs. As we can see from the diagram this is our only connection to the galaxy. This simulator will therefore have to connect to the client program, in order for us to send commands to System Platform. So when we send information from Source, the server will then pass the information received through network to the simulator where System Platform picks it up when scanning for changes. We define the input scan ourselves – e.g. once every 500 milliseconds. This will relieve the CPU time on the Virtual Machine running System Platform. System Platform operates on a very complex level of detail, especially when defining the structure of the “plant” definition. Every industrial plant is to be divided in sections, also called areas. This makes it easy for later use where operators need to see which section an alarm has occurred in.

For the simulator to work, it needs to be set up to communicate with System Platform. This is done by defining what usually a PLC I/O module in the physical world is. Inputs and outputs are always going through a PLC which collects the signals from sensors and alike in the plant. The simulator will allow System Platform to collect these signals without having to use a real physical PLC with connections.

All of the above goes for one-way communication with System Platform as the receiver. Again we refer to the above deployment diagram, where one can see that System Platform sends information directly from its simulator to our server.
Based on what we explained so far when the simulator receives a value through the client, the value will be forwarded to System Platform by the simulator.

For a complete class diagram of the System Platform client, please refer to SP Class diagram.jpg

For a complete sequence diagram, please refer to SP sequence diagram.jpg

Code wise we use sockets here as well, but unlike the Source client this client is written in c#. This is done because of the fact that the System Platform simulator is written in c#. We can yet again conclude that cooperation between C#, Java and C++ is no problem, and only gives us advantages.

To get more information regarding the code, please refer to the SP Client API

12 Iteration plan – Iteration #8

12.1 PURPOSE OF THIS DOCUMENT

The purpose of this document is to describe the tasks to be done throughout the iteration. It should contain estimate hour consumption for each activity, who is in charge of the activity, and who is supposed to work on that activity. It should also describe what the object of the iteration is.

12.2 AUTHOR

Leif H. Larsen

12.3 IN CHARGE OF ACTIVITY

Leif H. Larsen

12.4 OBJECTIVES OF THIS ITERATION

This iteration is the last iteration of the two transition iterations, as well as the last iteration of the project. We will finish all of our documents, and prepare everything for delivery, and we will start to prepare for the last presentation.

12.4.1 PRIMARY OBJECTIVES

- Complete documentation
- Create project poster

12.4.2 SECONDARY OBJECTIVES

- Start preparing for our final presentation

12.4.3 MILESTONES

- Project delivery

12.5 ITERATION PLAN

The timeframe for this iteration is 8 work days, starting at May 18th, thus ending at May 27th. Seeing as we only have documentation left, we allow ourselves to cut our working days from 7 hours a day, to 6 hours a day. This gives us a total of 192 hours for this iteration.

12.5.1 TIME SCHEDULE

The time schedule is an attachment to this document, please refer to "Estimated hours – Iteration #8".

13 Iteration report – Iteration #8

13.1 PURPOSE OF THIS DOCUMENT

To give an overview of how we worked in iteration #8 and how things went according to the plan, in terms of hours and achievements of goals.

13.2 AUTHOR

Leif H. Larsen

13.3 IN CHARGE OF ACTIVITY

Leif H. Larsen

13.4 GOALS

13.4.1 LIST OF GOALS FOR THE ITERATION

13.4.1.1 PRIMARY

- Complete documentation
- Create project poster

13.4.1.2 SECONDARY

- Start preparing for our final presentation

13.4.1.3 MILESTONES

- Project delivery

13.5 TIME CONSUMPTION

13.5.1 ESTIMATED HOURS

Please refer to the attachment "Estimated hours – Iteration #8" for the hours we estimated to use.

13.5.2 USED HOURS

Please refer to the attachment "Used hours – Iteration #8" for the hours we have used through the iteration, compared to what we estimated.

13.6 CONCLUSION

13.6.1 GOALS

13.6.1.1 PRIMARY

We did complete all of our primary goals, which we are happy with.

13.6.1.2 SECONDARY

We have started planning our final presentation, so we have completed this goal as well.

13.6.2 TIME CONSUMPTION

This iteration we had 192 hours available, which was 8 days' worth of work, with 6 hours a day. As one can see from the "Used hours – Iteration #8" document, one can see that we were spot on with the estimation.

14 Project report

14.1 PURPOSE OF THIS DOCUMENT

The purpose of the project report is to give a good overview and summary of the scope of the project. It is also supposed to give a conclusion containing ours view on how things went according to the plan.

14.2 AUTHOR

Jan Hansen & Leif H. Larsen

14.3 IN CHARGE OF ACTIVITY

Leif H. Larsen

14.4 WHY ARE WE DEVELOPING THIS SYSTEM?

We are developing this system for Nebb Engineering that wanted to look at the possibilities of connecting the industrial Control system “System Platform” to a 3D game engine for operator training purposes.

Nebb wanted us to simulate a Workover Control System in a virtual world so that you could train wocs operators in a safe environment before sending them to work on a real system offshore.

The benefits of this are reduced cost, and the possibility to train on scenarios that is difficult to carry out in the real world. An example would be evacuation of an oil-rig.

14.5 ACHIEVED GOALS

14.5.1 PROJECT RESULT

By looking at the project plan we can get an overview of the results.

14.5.1.1 PROCESS MODEL AND TOOL

The process model we used was Unified Process (UP) together with Scrum, and the tool we used was Unified Modeling Language (UML). This fitted great to this type of project and we always had a good grip on how things went according to the plan and the remaining time of the project. We have used scrum meetings every day to find out what everyone was going to work on that day and we feel that this helped us getting better progress in the longer run. Using UML for analysis and design of the system was a success and described the project the way we want.

14.5.1.2 DOCUMENTATION

In the projects scope we have written many documents. We have written a pre study report that reflects what we knew about the project in the beginning. This gave us an idea of what we were to do, which in turn gave us the possibilities to determine requirements. All the requirements were gathered in the requirement specification, which were written in parallel with the test specification. Further on we have written a system analysis to analyze what we were to do, which then led on to the design document. After the design document we moved on to the implementation, and the implementation document, and at last we have this project report, to conclude and sum up the project plan.

We have also written a lot of other documents, related to project management, such as weekly follow-up document, iteration plans and reports, meeting requests/commentaries. Also we have written several technology documents, which describe different technologies we have used, in terms of how to use it, and why we have chosen the different technologies.

14.5.1.3 RESPONSIBILITIES

As for responsibilities we have had different technical responsibilities, as well as non-technical responsibilities. This has worked quite well, and the distribution of responsibilities turned out to be very good. Each one of us took responsibility when we were assigned to do so, and took charge of what to do. The distribution of assignments within each phase also worked quite well, and everybody has been working with every area of the project. Each of us have also had responsibility of at least one technical thing, so that everybody have something they can say "I did this" about.

14.5.2 REAL COST

When it comes to the cost of the project we are talking about two different costs; first the actual cost in money, and second, the cost in man hours, and resources.

If we see our budget, we had a quite big budget, which gave us a total of NOK 83.256,-. This is mostly because we had some quite expensive 3D modeling tools. However, as it turned out we were able to get student licenses for these, and these were free. Our total accounting ended on NOK 22.107,-, where the computer hardware were the biggest post.

As for man hours we had a budget of 500 hours each group member. When we sum up all hours we have used, we see that this budget has been blown. We are all around 570 hours, which can be explained with the fact that we had to use another graphics engine than originally planned. Because of this we had to use more time on learning about this engine, and many hours have been used to configuring the source code to our needs.

14.5.3 EVALUATION OF THE PRODUCT

14.5.3.1 TECHNICAL ACHIEVEMENT

When it comes to our requirements, we managed to finish them all, despite the fact that we did not receive EyeSim. This is because we had done proper risk analysis beforehand, and we had a plan for what to do if this was the case.

14.5.3.2 PRODUCT EVALUATION

The product itself is a prototype, and it is obvious not ready to be sold at the current state. We are however very happy with how it has turned out, as is a good way of proving that you can connect a real life control system to a virtual training environment. This being said, there are several points that can be improved.

Please refer to the document Future recommendations to see a list of things that can be improved.

14.6 PROJECT EXECUTION

14.6.1 PROJECT MODEL

When it comes to the project execution we chose to follow the unified process model. This is a model we have used in several courses at HiBu, and it is appropriate for use in software projects. This model comes with four phases, which are inception, elaboration, construction and transition.

14.6.1.1 INCEPTION

Through this phase we did a pre-study, and agreed on different areas each of us were to be responsible of. We found, in cooperation with Nebb, requirements, and specified tests and test strategies. We did also schedule the rest of the project, writing a project plan and defining the project scope.

14.6.1.2 ELABORATION

During this phase we did a system analysis, and went on to design our system. We created some UML diagrams, and got a good idea of what to do. We did also get our initial, stable architecture in place through this phase.

14.6.1.3 CONSTRUCTION

During this phase we did most of the work, as far as implementation concerned. We had already a defined plan of what to do, and we had a stable architecture, so we worked on this to complete our requirements. We did also some testing of our product, as defined in the test specification.

14.6.1.4 TRANSITION

During this phase we did a lot of testing, and we were finishing our product. We did also a lot of documentation, as this was to be delivered at the end of this phase

14.6.2 HOURS USED

As we were to define a project plan, we were also supposed to estimate the hours we were to use throughout each iteration. At the start of the project this turned out to be quite difficult, seeing as none of us had any experience with this earlier. We missed by many hours for the few first iteration, but as the project progressed, we got more exact estimations.

14.6.3 QUALITY CONTROL

14.6.3.1 MEETINGS

We have been having weekly meetings with our internal supervisor, as well as monthly meetings with our external supervisor. We have also been in contact with our employees through mail, and for the last two months, we have been working at their location one day a week.

14.6.3.2 DOCUMENTS

For the documents we have made sure that every document has been read through by others than the author, to make sure that everything makes sense, and that there are no errors.

14.6.3.3 PRODUCT

As for the product we have been doing a lot of testing during our work.

14.6.3.4 PRESENTATIONS

We have had feedback forms for the audience during each of our presentations, which have given us valuable feedback on what to improve to the next presentations.

14.6.4 CHALLENGES

Through this project we have had some challenges.

14.6.4.1 EYESIM NOT RECEIVED

We did not receive EYESIM as planned. There are several reasons for this, but mainly because time issues. Seeing as this product is brand new, the developers thought we would need a crash course as an introduction, but they were not able to give this within the time limits we had. Also the communication was slower than we had wished.

14.6.4.2 TECHNICAL

We have also had several technical difficulties, as found during testing. This includes the following:

- VWOCS server only able to have one client connected
- Difficulties to configure an entity in Source to receive commands
- General difficulties with communication
- AI implementation in Source
- A lot of missing/not described documentation on Source SDK

14.7 CONCLUSIONS

We feel that the project has been a great experience, and a great introduction to how an engineer's life could be. We have had some difficulties especially that the graphics engine we were supposed to use never showed up but from this we have learned that good planning can save a project.

Because of the way we have used Source engine there was not that many tutorials, and there was not many people that were able to help, when we had trouble, on the official forums, so we had to figure a lot out ourselves. But this is just positive, as this helps us becoming more creative and proves that we can think outside the box when needed.

The four of us now feel ready to start working as newly educated engineers. We have seen that we are capable of producing a professional system, as well as administrate the process, so we can conclude that this project has been successfully completed.

15 Future recommendation

15.1 PURPOSE OF THIS DOCUMENT

This document is meant to give some recommendations for future development of the project.

15.2 AUTHOR

Leif H. Larsen

15.3 IN CHARGE OF ACTIVITY

Leif H. Larsen

15.4 IMPROVEMENTS POTENTIALS

As we have solved this project we have found several points which can be improved.

15.4.1 SIMULATOR

The simulator we have got from Nebb is only a beta and it has a few bugs in it. Sometimes it crashes without any reasons, and there are times it freezes during runtime. Also this simulator is running as a DDE server, which requires a lot of modification in the galaxy database, as far as inputs sources and so on. This could be avoided by using a native OPC simulator, which by our understanding is available. If this is to be used it needs to be modified a bit, to allow running as a client, sending and receiving commands.

15.4.2 GRAPHICS ENGINE

The Source engine is an old game engine, at has a lot of limitations. The graphics can be done more realistic, simply by changing the engine itself. Suggested engines are Unity 3D and Crytek.

15.4.3 GENERAL IMPROVEMENTS

For some general points that can be improved we see that we could have built the system differently. However, this was the best solution considering that we had to be able to change from Source to EYESIM. For further development of this project it is recommended that the server is included in either the simulator part or the graphics part. This could make the work flow easier, and it might take away any potential problems with delay in the transfer of commands.

Another thing that should be a feature is the possibility to supply both the graphics engine and simulator with one file, containing all the inputs sources, so everything got properly configured with this file.

16 Fun Facts

This document is a bit different from the rest. The purpose is to inform the reader with information in a somewhat summarized context that covers our whole project from our point of view.

To keep ourselves fresh, motivated and waking up every morning with a burning desire to get our hands dirty with some project work (well not literally, but you get the point), we had to actually travel to Sweden! We were quite burned out after the first presentation in January, so we went on a road trip to Svinesund, Sweden the day after. Here we bought a whole load of beverage as you can see from the picture beneath:

FIGURE 20 - SOCIALISING HAS ITS PRICE

Thanks to Torjus' extensive need for Caprisonne juice drinks and Leif's huge thirst for Coke, the trunk in the poor Peugeot was full after only 1 trip to the store! Much of the road trip time was spent on buying beverage and food. Some sweets were also bought of course. All in all, this trip was a marvelous idea which helped us improve our motivation further on when it was most needed.

We did not have any special socializing events apart from this road trip, but we had a pizza evening where we worked at school.

So, to present you with some random "out-of-nowhere" information, let's check the numbers! We have summarized the total count of words written in the project and the total number of code lines written…

Number of words written:	**83 153**
Number of code lines in Java:	301
Number of code lines in C#:	284
Number of code lines in C++:	3455
Total:	**4040**

As you can see we haven't exactly slacked throughout the last year! It was a time we spent well considering all the things we learned in such a short time. We probably couldn't pull this off alone, but we managed to complete the project as a group. We have had a great time and a good deal of memories from the project time.

17 References

17.1 INTERN DOCUMENTS

- Vision Document
- Definitions, acronyms and abbreviation
- List of documents
- Standarddokument (Document standard, written in Norwegian)
- Contact information
- Activities
- Overordnet (A Gantt chart)
- Budget
- Responsibilities
- Code standard
- Introduction to Requirements
- Introduction to UML
- Source API
- SP Client API
- Server API
- Test strategy
- Transmission protocol
- Estimated hours – Iteration 8
- Used hours – Iteration 8

17.2 WEBSITES

- http://www.nebb.no/
  (Last visited: 20.05.2011)
- http://iom.invensys.com/EN/Pages/SimSci-Esscor_EYESIMImmersiveVirtualRealityTrainingSystem.aspx
  (Last visited: 20.05.2011)
- http://en.wikipedia.org/wiki/Unified_Process
  (Last visited: 27.09.2010)
- http://en.wikipedia.org/wiki/Use_case
  (Last visited: 02.12.2010)
- http://en.wikipedia.org/wiki/Unified_Modeling_Language
  (Last visited: 02.12.2010)
- http://en.wikipedia.org/wiki/Software_Requirements_Specification
  (Last visited 02.12.2010)
- http://www.buzzle.com/editorials/4-10-2005-68350.asp
  (Last visited: 15.12.10)
- http://www.buzzle.com/editorials/4-10-2005-68349.asp
  (Last visited: 15.12.10)
- http://en.wikipedia.org/wiki/Systems_analysis
  (Last visited: 20.05.2011)

17.3 EXTERNAL BOOKS AND/OR MAGAZINES

- Project manual (HiBu), Torbjørn Strøm & Olaf Hallan Graven
- Essential software testing – A use care approach by Greg Fournier
- Software testing: An ISTQB – ISEB Foundation Guide by Brian Hambling
- Raven Test specification
- Rivet test specification