**Software design**

**Team project – Deliverable 1**

**Team number**:29

**Team members:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Student Nr.** | **E-mail** |
| Casper van Ek | 2556917 | [2556917@student.vu.nl](mailto:2556917@student.vu.nl) |
| Thomas Webbers |  |  |
| Yuri Henrique Borges |  |  |
| Kristina Hristov |  |  |

This document has a maximum length of 10 pages.

**April 2013**

# Table of Contents

[Table of Contents 2](#_Toc505179048)

[1. Introduction 3](#_Toc505179049)

[2. Requirements Specification 4](#_Toc505179050)

[2.1 Requirements 4](#_Toc505179051)

[2.2 Use Cases 4](#_Toc505179052)

[3. Implementation remarks 5](#_Toc505179053)

[4. References 5](#_Toc505179054)

# 1. Introduction

**Author(s)**: Casper

The ROVU platform is a type of rover specifically designed for autonomous use. That is to say, to function and achieve its goals without a human operator. We have outlined a design for a modified version of the ROVU platform, with the goal of having it explore an unknown environment.

Our modified version sports more sensors than the regular version to make sure object detection is done correctly. This is especially important as each ROVR needs to be able to operate in concert with other ROVR’s in the same environment. And as such the extra sensors are necessary to prevent a ROVR from viewing another ROVR as an obstacle.

Each ROVR will maintain a grid representation of the environment. A ROVR will also keep track of which locations it detected obstacles in, and which locations it has already visited. It will prevent collisions by mapping a path through its grid representation of the environment, avoiding any obstacles detected.

<Extensions of the base mission?>

To determine its path to each location the ROVR needs to visit it will use a <?> algorithm. Which works by going through cycles starting at the start and end point of the path to be generated. Each cycle all nodes reachable from the already visited nodes will be registered, which means after 5 cycles you will have found all nodes which you can reach with 5 moves in the grid. This continues until the nodes reachable from the start and end overlap. At which point the shortest path to the goal has been found.

We will also add a central station overseeing the robot behavior. However it will not be responsible for the paths of the ROVR’s. It’s main purpose will be the exchange of data between the ROVR’s and as a retrieval point of data for the user. This is to make sure ROVR’s have a central location to exchange what parts of the grid have already been visited and where obstacles are located. But to still ensure the robots can function on a high level of concurrency. And finally to make sure the user has a single point to retrieve data from, instead of a variable number of robots.

<does not mention number of robots, or types of robots. But I see that as implied>

Write a short description of your version of theROVU system that you will design here. Clearly specify which are the key aspects of your system, such as:

* obstacle avoidance strategies,
* extensions to the base mission provided in the team project guide,
* presence of a central station orchestrating all the robots, if needed,
* decide whether your environment is fully known at the beginning of the mission (so the robots can actually plan their trips), or the environment is unknown (so the robot can encounter unpredicted obstacles), etc.,
* number of involved robots in the missions (maybe your system can be totally independent from the number of robots),
* Types of considered robots, for example you may have a special fast robot only for mapping all the obstacles in the environment, and then “dummy” robots without sensors that take photos only, or you may have always the same type of robots, etc.,

Be creative here!

Don’t forget to mention your references (e.g., to known obstacle avoidance algorithms like this: <http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>), if applicable.   
  
Recommended amount of pages: 2-3

# 2. Requirements Specification

**Author(s)**: <name of the team member(s) responsible for this section>

This chapter contains the specification and UML representations of all the requirements. The chapter is sectioned in thesections below.

Recommended amount of pages for the whole chapter (including also sections 2.1 and 2.2): 6-8

## 2.1 Requirements

**Functional requirements**

|  |  |  |
| --- | --- | --- |
| **#** | **Short Name** | **Description** |
| F1 | Obstacle avoidance | The rovers shall move freely in the environment and avoid obstacles autonomously. |
| F2 |  |  |
| … |  |  |

**Non-functional requirements**

|  |  |  |
| --- | --- | --- |
| **#** | **Short Name** | **Description & reasoning** |
| NF1 | Obstacle avoidance [Performance] | Each rover shall react to the presence of an obstacle within 50 milliseconds. |
| NF2 | Obstacle avoidance [Safety] | A rover shall always be at least 1 meter from obstacles, other robots, and human beings. |
| … |  |  |

Each non-functional requirement must be tagged with the corresponding category (see slides 24 and 25 of the second lecture for knowing them + refer to Chapter 4.1 of the Sommerville book in Canvas).

## 2.2 Use Cases

Describe ANDillustrate your system use case diagrams in this paragraph. Each use casediagram must be represented by:

* a UML use case diagram AND
* a table conforming to the Cockburn templatefor each use case.

<Figure representing the UML use case diagram>

|  |  |
| --- | --- |
| **Name** | Use case 1 |
| **Short description** |  |
| **Precondition** |  |
| **Postcondition** |  |
| **Error situations** |  |
| **System state in the event of an error** |  |
| **Actors** |  |
| **Trigger** |  |
| **Standard process** |  |
| **Alternative processes** |  |

# 3. Implementation remarks

**Author(s)**: <name of the team member(s) responsible for this section>

In this chapter you will elaborate on what you implemented in your base version of the robotic system.

Recommended amount of pages for this chapter: 1

# 4. References

References here.