Draft Workplan

Wadie Assal 6398693
Jeroen Rooijmans 5887410
Jasper Timmer 5995140
Maarten de Waard 5894883
Maarten Inja (vz) 5872464

June 10, 2011

1 Specification

In this chapter you can find a list of the products that will be delivered during the project Software Engineering and Distributed Applications (BINSEGA6). You can also find a description of the product and witch specifications it will comply to.

1.1 Work plan

The work plan is the document you are reading now. The work plan will contain design decisions, the project planning and specifications of the different software components. Furthermore it will contain the information needed to start the project. It will also function as guidance during the project.

1.2 Presentation

We are going to present our finished software product on Thursday in week 26. This will be done with the help of a slide show. The following subjects are going to be discussed in this presentation:

- How did the development process go?
- What can the software product do?
- What cant the software product do?
- What can be improved in future?

1.3 Demonstration

The demonstration is going to be given right after the presentation mentioned above. The sole purpose of the demonstration is showing that our software works correctly as specified in the delivered documents. We will be proving this by running our software in different scenarios.

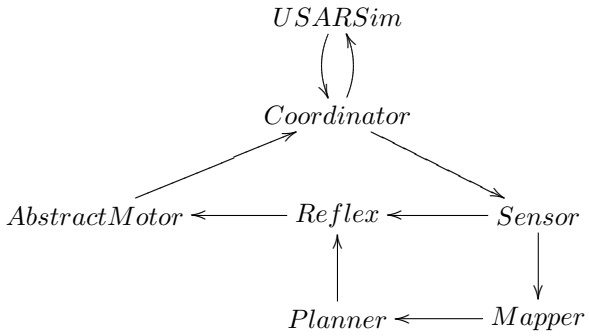

Figure 1: A representation of our functional design

1.4 Final report

The final report is going to be the last delivered product. After a review with the professor of the draft, we will use the given feedback to produce the final version of this product. You can find the following subjects in the final report:

- Project goals
- The development process
- Software architecture and explanation of design decisions
- Software manual

1.5 AP2DX Software

This is the actual software that has to be delivered. The goal of the software is to use a robot to map an area. During the mapping the robot needs to avoid obstacles and he has to constantly know where he is on the map. Further requirements are:

- The software needs to be distributed
- Limited failsafe possibilities
- It should run over a network or on one device

2 Functionial Design

2.1 Components

This section will discuss the components of the functional design seen in figure 1

2.1.1 Coordinator

This module will coordinate data traffic between the USARSim engine and our program. It distributes the data to several modules, as can be seen in the diagram.

2.1.2 Abstract Motor

With this module, we maneuver the P2DX robot in the virtual world. We can control two motors, one connected to the left wheel and one to the right wheel. When a path is planned or a reflex kicks in, it sends a command to the Abstract Motor module. This translate the given commando to a series of operations we then apply to the virtual motors.

2.1.3 Sensor

The sensor module collects sensory data from the robot. This data is checked and translated. If certain data crosses a critical level, a command is sent to Reflex to prevent the robot from crashing. All data is translated and sent to our Mapper.

2.1.4 Reflex

To prevent the robot from crashing, we implement a module that is always checking the sensory data. When some critical level is reached (e.g. a wall gets too close in front of our robot, or the motors try to turn, but the wheels do not) we take action, thus preventing our robot from breaking.

2.1.5 Mapper

To be able to plan routes and navigate through the environment, our robot should be able to create a map. This map is a representation of its environment, based on the sensory data the robot receives. This module will also be able to track the robots location in this map, by using data from, for example, the wheel sensors.

2.1.6 Planner

To be able to create a map, our robot needs to plan a path to (and through) unknown regions. Our planner sends the robot to new areas, while avoiding obstacles. The Planner also calculates our current location in the environment and checks if our location is on the path we want to follow. The Planner uses data from the Sensor and from the Mapper.

3 Milestones

This section will describe the milestones, and what deliverables we will provide.

Milestone 0

This will describe what we will have done by the 10th of June. These are the deliverables:

Working environment: We will set up a Git repository[1] and a testing environment (using jUnit).

[1] https://github.com/Y3PP3R/AP2DX

Base class: We will make an abstract base class, on which we can base all our Java classes. This will contain all the standard methods, e.g. TCP/IP protocols.

Work plan: We will make a work plan, consisting of our planning and description.

Milestone 1

This will describe which deliverables we will have done by the 17th of June.

Drive: We want the program to be able to direct the robot through the environment. We will not yet focus on the ability to follow lines or walls.

Avoid collision: The robot should be able to avoid collision with objects and walls. It will stop, turn a random corner, and drive on. This way it will cover most of the area without colliding.

Experiment and content of final report: We will have a draft of the final report, containing a description of the experiment and its contents.

Classes needed to be implemented for these goals:

- Coördinator
- Sensor
- Reflexes
- Motor

Milestone 2

This is what we will have finished before the 23rd of June.

Avoid obstacles: The robot will be able to avoid the obstacles that cross its path, in stead of stopping and turning a random corner.

Navigate: The robot will be able to navigate through the room.

What we will implement for this:

Mapper: A class that creates a map of the room out of the sensor data. In the time of milestone 2 it does not have to be able to create an entire map and be very accurate, but it will be able to make some implementation.

Improved Sensor: The sensor class will be improved to be able to make an accurate map.

Improved Reflexes: The reflex class needs to be able to use some sensor data to be able to avoid objects appropriately.

Demonstration

Before the demonstration we will be able to do the following things:

Planner: We will have a planner class that can specify directions based on the current map position and what part of the map we have not yet discovered.

Improved Mapper: The mapper will now be able to make an accurate map and find our location on it, while taking the errors in sensor data into account.

Tests: We will test everything thoroughly.

Report: We will work on a report, describing our progress, problems and (test)results.

Documentation: We will work on a proper documentation of our code.

These classes we will be implemented to achieve our goals:

- Mapper
- Planner
- All test classes

Name	Work
Create working environment	**3d**
Set up Git	1d
Set up testing software (jUnit	1d
Set up test server	1d
Base class	**1d 7h**
Create class	1d
Create Test class	7h 45...
Work Plan	**3d**
Draft work plan	
Make Work Plan	3d
Turn in Work Plan	
Drive and avoid collision	**14d 1h**
Make Coordinator Class	2d
Make Abstract Motor Class	3d
Make Sensor Class	2d 6h
Make Reflexes class	2d 6h
Test everything	1d 4h
Improve Sensor Class	1d
Improve Reflexes Class	1d
Milestone 1	
Navigation and Paths	**9d**
Start on Mapper	1d
Make Planner	4d
Make Mapper	4d
Milestone 2	
Perfection and tests	**8d**
Improve Planner	3d
Improve Mapper	3d
Test and fix everything	2d
Final report	**19d**
Make draft of final report	1d
Make and fill contents	8d
Specify experiment and contents	
Finish draft report	3d
Turn in a draft	
Finish report	5d
Turn in report	
Make presentation	2d
Do presentation and demo