Software Design Document (SDD)

Group 2 *Coding Pharaohs*

Abdulaziz ALHULAYFI a1642362
Yu HONG a1616861
Jianqiu LI a1635717
Matthew NESTOR a1132338
Yifei PEI a1611648
Bowen TAO a1622211

November 4, 2013
Final Document

Contents

Change History

Date	Version	Reason for Change
10th Sept	0.1	Initial draft
15th Sept	0.2	Expanded sections 2,3,4
19th Sept	0.3	Expanded sections 7,8 and minor changes

1 Introduction

1.1 Purpose

SDD, which is abbreviation for Software Design Document, describes the system design in the software project. Specifically, it is needed to transfer a specification into an executable system. This document is the SDD for developing a robot to mark road closures in a city, which is destroyed by a natural disaster. This SDD will document the GUI, AI, Robot Move, Map Representation, and Communication that will be utilized in implementing the system for the robot.

1.2 Scope

The aim of this project is to develop a prototype of a robot which may be used to produce a map of the ruined city and mark the road closures in order to prevent people from moving into these areas.

The map includes virtual representations of closures, disaster area, obstacles, the intersection, roads, and unexplored areas. And the physical map includes using cardboard or some rigid material to represent obstacles and put them randomly, and using black lines as roads. The most important is that the robot should detect the disaster area and mark the road closures with a whiteboard marker. And in this procedure, the robot also has to avoid the obstacles and go forward.

The virtual map will be dynamically displayed on the system Graphical User Interface (GUI), and it is created and exported in the form of an Extensible Markup Language (XML) document.

Within the map, the robot will move from the start point and explore the whole map so that it can identify where the obstacles are and mark where the road closures are.

For the purposes of this project the city is no larger than A1 paper size.

This SDD should be related to the other project documents, namely the Software Project Management Plan (SPMP), the Software Requirement Specification (SRS).

1.3 Overview

This SDD consists of the following:

- Introduction introduces the aims and function of the project;
- System Overview briefly gives a broad outline of the architecture of the project;
- System Architecture and Components Design gives a detailed description of the project architecture, including its composite components;
- Data Design describes the a variety of data structures used in the project;
- Design Details includes the Class, State and Interaction Diagrams in this project;
- Human Interface Design describes the perspective of the user to utilize the robot.
- Resource Estimates summaries computer resource estimates required for operating the robot.

1.4 Document Conventions

All diagrams unless otherwise noted follow standard UML conventions.

2 System Overview

2.1 Background

The Road Closure Marking Robot is a prototype of an on-road, land-vehicle style robot, whose function is to investigate disaster-stricken urban zones and mark road closures to prevent access to hazardous areas. As this model is a prototype, the robot is miniaturised, and rather than following city roads, it follows black lines on an A1 size sheet of paper.

For a full description of the requirements, see the accompanying SRS.

2.2 Technologies Used

The LEGO Mindstorms NXT robot set was used to create the prototype, as its size and sensor functionality make it well suited as a prototype. Items used from the NXT set include the intelligent brick, three servo motors, an ultrasonic sensor, a light sensor, two touch sensors, two large wheels, one small wheel, and various connecting parts. The Intelligent brick is the computer and controller for the robot's motors and sensors. The brick has been flashed with leJOS firmware. LeJOS is a Java extension for the programming of the LEGO Minstorms NXT. Thus, all software for the project shall be written in Java and leJOS.

The host system and intelligent brick communicate via bluetooth.

2.3 Software Design Overview

The system consists of two sides: the robot side and the host side. The robot side is run on the NXT intelligent brick, and the host side is run on a remote terminal. The two sides communicate via bluetooth. The robot has two modes: automatic and manual. In automatic mode, the robot uses artificial intelligence to navigate to disaster zones and mark road closures on its own. In manual mode, a user controls the robot via a graphical display on the host side. The robot has the ability to create, save, and load maps of city areas. The maps contain information about such things as the city roads, disaster zones, the robot's location, and explored and unexplored areas (for more detail see section 4). The maps are represented graphically to the user on the host side, but are also available to the AI, and are used to efficiently guide the robot to roads in need of closure.

3 System Architecture and Components Design

3.1 Architectural Description

The system employs a pipe-and-filter architectural pattern, and consists of a Robot Side and a Host Side (see figure 1).

The robot side contains the following subsystems:

- **Robot Motor and Sensor Control:** Responsible for parsing movement commands from the Robot Side Communications Unit and turning them into leJOS motor commands. Responsible for receiving sensor data from the robot and sending it to the

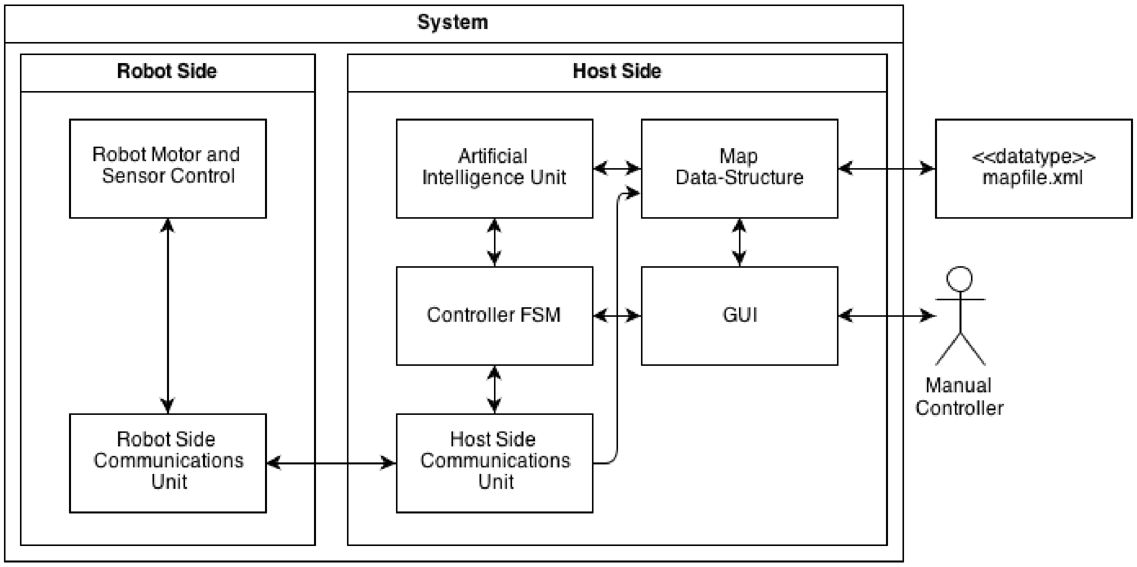

System
Robot Side
Host Side
Robot Motor and Sensor Control
Robot Side Communications Unit
Artificial Intelligence Unit
Map Data-Structure
Controller FSM
GUI
Host Side Communications Unit
<<datatype>> mapfile.xml
Manual Controller

Figure 1: A graphical display of the system architecture

Robot Side Communications Unit. Responsible for initiating emergency stop under certain circumstances (e.g. bluetooth connection failure).

- **Robot Side Communications Unit:** Responsible for receiving sensor data from Robot Motor and Sensor Control, and passing that data via bluetooth to the Host Side Communications Unit. Responsible for receiving movement commands from the Host Side Communications Unit via bluetooth, and passing them to Robot Motor and Sensor Control. Responsible for sending a "stop" message to Robot Motor and Sensor Control in the event of a bluetooth connection failure.

The Host Side uses the Manager Model, a centralised control style architecture. It contains the following subsystems:

- **Host Side Communications Unit:** Responsible for receiving sensor data from the Robot Side Communications Unit via bluetooth, and passing it to the Controller FSM. Responsible for receiving commands from the Controller FSM and sending them to the Robot Side Communications Unit via bluetooth. Responsible for sending sensor data to Map Data-Structure for the purposes of updating the representation of the robot's location.

- **Controller FSM:** A finite state machine with two states: automatic and manual. It is responsible for intercepting and responding to events from the GUI, the Robot (via the communications units), and the Artificial Intelligence Unit.

- **Artificial Intelligence Unit:** A finite state machine with five states: initial, search, explore, mark, and return. It is responsible for the behaviour of the robot when the Controller FSM is in automatic mode.

- **Map Data-Structure:** Responsible for creating a map data-structure to be used by (1) the Artificial Intelligence Unit for the purposes of AI navigation, and (2) the GUI for the purposes of generating a graphical display of the map to the user. The data-structure can be loaded from, and saved to, an xml map file (see section 4 for details). Receives and interprets sensor data from the Host Side Communications Unit in order to update its representation of the robot's location.

- **GUI:** The graphical user interface for controlling the robot and representing to the user the current state of the robot and the city (for details see section 5). Also responsible for sending commands to the Controller FSM and for interpreting map and robot data from Map Data-Structure.

3.2 Component Decomposition Description

The following is an object oriented component decomposition description of the above subsystems:

3.2.1 Robot Motor and Sensor Control

The Robot Motor and Sensor Control subsystem consists of two components:

- SensorInfo
- RobotBehaviourControl

3.2.2 Robot Side Communications Unit

The Robot Side Communications Unit consists of two components:

- RobotCommunication
- BTReceiver

3.2.3 Host Side Communications Unit

The Host Side Communications Unit consists of two components.

- PCReceiver
- PCComms

3.2.4 Controller FSM

Controller FSM contains only one component:

- ControllerFSM

3.2.5 Artificial Intelligence Unit

The Artificial Intelligence Unit consists of ten components:

- Graph
- Node
- Edge
- Vertex
- CollisionResult
- TestOfCollisionTest
- AStarComparator
- MyInt
- ArtificialIntelligenceFSM
- Action

3.2.6 Map Data-Structure

The Map Data-Structure has nine component:

- Map
- RobotMap
- Obstacle
- DisasterZone
- UnexploredZone
- Road
- Closure
- Intersection
- RobotLocation

3.2.7 GUI

The GUI consists of two components:

- GuiOfSEP
- MapDraw
- GUICommands
- BatteryInfoTask
- SensorRequestTask
- FileListener
- BatteryDisplay
- DistanceDisplay
- XMLFileFilter

3.3 Detailed Components Design Description

3.3.1 SensorInfo

- Component Identifier: SensorInfo
- Purpose: To fulfil requirements R02, SRS section 3.1.2, R03, section 3.1.3, R04, section 3.1.4, R05, section 3.1.5, R06, section 3.1.6, and R07, section 3.1.7
- Function: Responsible for constructing and the NXT sensor classes and providing convenience methodsmfor them.
- Subordinates: None
- Dependencies: RobotBehaviourControl
- Interfaces: None
- Data: Light sensor data, touch sensor data, ultrasonic sensor data, tachometer data

3.3.2 RobotBehaviourControl

- Component Identifier: RobotBehaviourControl
- Purpose: To fulfil requirements R01, SRS section 3.1.1, R02, SRS section 3.1.2, R03, section 3.1.3, R04, section 3.1.4, R05, section 3.1.5, R06, section 3.1.6, and R07, section 3.1.7
- Function: Creates an instance of SensorInfo. Receives commands from the Robot Side Communications Unit and turns them into leJOS motor commands. Retrieves sensor data from SensorInfo and sends it to the Host Side Communications Unit.
- Subordinates: SensorInfo
- Dependencies: RobotCommunication
- Interfaces: None
- Data: Motor command data

3.3.3 RobotCommunication

- Component Identifier: RobotCommunication
- Purpose: To fulfil requirement R08, SRS section 3.1.8, and H05, SRS section 3.3.5
- Function: Creates an instance of RobotBehaviourControl. Contains methods for opening and closing the connection between robot and host systems, creating datastreams between them, receiving commands from the Host Side Communications Unit, issuing commands to RobotBehaviourControl, and sending data to the Host Side Communications Unit. Responsible for sending a "stop" message to RobotBehaviourControl in the event of a bluetooth connection failure.
- Subordinates: RobotBehaviourControl
- Dependencies: PCComms
- Interfaces: None
- Data: motor command data, sensor data

3.3.4 BTReceiver

- Component Identifier: BTReceiver
- Purpose: To fulfil requirement R08, SRS section 3.1.8, and H05, SRS section 3.3.5
- Function: The main program which creates an instance of RobotCommunication and calls its connection setup
- Subordinates: RobotCommunication
- Dependencies: None
- Interfaces: None
- Data: None

3.3.5 PCReceiver

- Component Identifier: PCReceiver
- Purpose: To fulfil requirement R08, SRS section 3.1.8, and H05, SRS section 3.3.5
- Function: Interprets raw data received from RobotCommunication into light sensor, battery, tachometer, distance, and touch data. Contains convenience methods for retrieving interpreted data for each of the above sensors.
- Subordinates: None
- Dependencies: PCComms
- Interfaces: None
- Data: Sensor data

3.3.6 PCComms

- Component Identifier: PCComms
- Purpose: To fulfil requirement R08, SRS section 3.1.8, and H05, SRS section 3.3.5
- Function: Creates an instance of PCReceiver. Contains methods for opening and closing the connection between the robot and host systems. Passes raw data received from RobotCommunication to PCReceiver.
- Subordinates: PCReceiver
- Dependencies: RobotCommunication
- Interfaces: None
- Data: Motor command data, sensor data

3.3.7 ControllerFSM

- Component Identifier: ControllerFSM
- Purpose: To fulfil system features, SRS section 4.1 and 4.2, and to fulfil system requirements H02, SRS section 3.3.2, and R09, section 3.1.9
- Function: A finite state machine with two states: automatic and manual. It is responsible for intercepting and responding to events from the GUI, the Robot (via the communications units), and the Artificial Intelligence Unit.
- Subordinates: AiFsm, GuiOfSEP
- Dependencies: PCComms, AiFsm, GuiOfSEP
- Interfaces: None
- Data: Motor command data, sensor data, state data

3.3.8 Node

- Component Identifier: Node
- Purpose: To fulfil requirements R09, SRS section 3.1.9, R01, section 3.1.1, and M01, section 3.2.1
- Function: The node component of the SearchTree data-structure.
- Subordinates: None
- Dependencies: SearchTree
- Interfaces: None
- Data: None

3.3.9 Edge

- Component Identifier: Edge
- Purpose: To fulfil requirements R09, SRS section 3.1.9, R01, section 3.1.1, and M01, section 3.2.1
- Function: The edge component of the SearchTree data-structure.
- Subordinates: None
- Dependencies: SearchTree
- Interfaces: None
- Data: None

3.3.10 SearchTree

- Component Identifier: SearchTree
- Purpose: To fulfil requirements R09, SRS section 3.1.9, R01, section 3.1.1, and M01, section 3.2.1
- Function: A data-structure with associated methods for performing searches over sets of nodes connected by edges. For a given initial node and a goal node, SearchTree can return the shortest sequence of hops to get from one to the other.
- Subordinates: Node, Edge
- Dependencies: Pathfinder
- Interfaces: None
- Data: Edge and Node data

3.3.11 Pathfinder

- Component Identifier: Pathfinder
- Purpose: To fulfil requirements R09, SRS section 3.1.9, R01, section 3.1.1, and M01, section 3.2.1
- Function: Creates a SearchTree with a node for every intersection in the map and an edge for every road connecting them. Contains methods for returning the next road to be taken by the robot.
- Subordinates: SearchTree
- Dependencies: AiFsm
- Interfaces: None
- Data: SearchTree data

3.3.12 Explorer

- Component Identifier: Explorer
- Purpose: To fulfil requirements R09, SRS section 3.1.9, R01, section 3.1.1, and M01, section 3.2.1
- Function: Responsible for controlling the robot's movement when there are no areas of the map explored or when there are some known roads which are not mapped.
- Subordinates: None
- Dependencies: AiFsm
- Interfaces: None
- Data: Map data, sensor data, motor command data

3.3.13 AiFsm

- Component Identifier: AiFsm
- Purpose: To fulfil requirements R09, SRS section 3.1.9, R01, section 3.1.1, and M01, section 3.2.1
- Function: The main controller of the AI. A finite state machine, it is responsible for determining whether the robot should find a path to a location, explore, mark a road closure, or return to base. It retrieves map data from the Map Data-Structure, as well as sending robot commands to ControllerFSM to be passed to PCComms.
- Subordinates: Pathfinder, Explorer
- Dependencies: Map, ControllerFSM
- Interfaces: None
- Data: State data

3.3.14 Map

- Component Identifier: Map
- Purpose: To fulfil requirements M01, SRS section 3.2.1, M02, section 3.2.2, M03, section 3.2.3, and M04, section 3.2.4
- Function: Can create a map data-structures on the basis of map xml files, and can save these data-structures back to XML files. Interfaces with the GUI and AiFsm, providing convenience methods for retrieving data about the city and robot location.
- Subordinates: None
- Dependencies: PCComms
- Interfaces: None
- Data: Map data, sensor data

3.3.15 GuiOfSEP

- Component Identifier: GuiOfSEP
- Purpose: To fulfil requirements H01, SRS section 3.3.1, H02, section 3.3.2, H03, section 3.3.3, H04, section 3.3.4, M02, section 3.2.1, M03, section 3.2.3, and R08, section 3.1.8
- Function: This provides the main graphical user interface functionality for the system. It responds to events for manually controlling the robot, loading and saving the map, providing the user with information about the robot and the city, etc. See section six for more details.
- Subordinates: MapDraw
- Dependencies: ControllerFSM, Map, MapDraw
- Interfaces: None
- Data: Map data, robot data

3.3.16 MapDraw

- Component Identifier: MapDraw
- Purpose: To fulfil requirements H01, SRS section 3.3.1, H03, section 3.3.3, M02, section 3.2.1, M03, section 3.2.3
- Function: This is responsible for drawing the map on the basis of data retrieved from the Map component.
- Subordinates: None
- Dependencies: Map
- Interfaces: None
- Data: Map data

3.4 Architectural Alternatives

The main choice in the architecture design process was whether to locate the AI on the robot side or the host side. There is an intuitive appeal to having the AI on the robot side: this is true to the idea that the robot is in control when in automatic mode. There are also certain conditions under which the robot must respond to events purely from the robot side, such as in the event of a connection failure, so it stands to reason that perhaps many events must be handled in this way. Furthermore, although the difference might be negligible, AI functionality would also be slightly faster if located on the robot side, since commands would not have to travel via bluetooth.

Nevertheless, the decision was made to locate the AI on the host side. The principle reason is for ease of interfacing with the Controller FSM. The Controller FSM needs to switch between automatic and manual modes quickly and effortlessly. Having the AI on the robot side would require messages being sent across bluetooth to indicate state changes to the robot, and this is clumsy. Similarly, the robot side would have to implement its own finite state machine to remember whether to accept commands via bluetooth or to generate its own, and this duplication of state representation is unnecessary. By having both automatic and manual modes run from the host side, the Robot Side Communications Unit does not need to worry whether received motor commands are sourced from manual or automatic modes.

3.5 Design Rationale

Most of the design choices were a consequence of necessity: subsystems were created as required to fulfil functional roles. The main design rationale regards the organisation of the Controller FSM and Artificial Intelligence Unit subsystems. The requirement for automatic and manual modes made it clear early on that a finite state machine architecture would be particularly suitable. While this was an obvious step for the Controller FSM, it was also helpful to think of the AI as an FSM. The FSM architectures are also advantageous in view of the safety critical aspect of this project, since FSMs are straightforward to program and debug.

4 Data Design

4.1 Database Description

Data Design remains not so much consideration in this project, as it is not heavily data driven, and some of considerations has been set by the client early. We identify the relevant subsections of data design below:

- Map External Representation
- Map Internal Representation
- Robot Internal Representation

4.2 Data Structures

- Map External Representation
  Map is to be saved in and read from XML format following the DTD presented by the client.In the XML file, attributes distinguished with each other, for instance, closures, obstacles, intersections, roads, disaster areas and unexplored zone will be added. And also, the boundary of map and robot status as a part of attributes will be griven in the XML file.

- Map Internal Representation
  The function is implemented by the interface name Map in this project.It is a interface defined all of methods to read the map file from XML format via loadMap(String s) method, absorbs the whole attributes on the map through different kinds of methods such as getObstacles(). More specifically,we created Closure, Obstacles, Intersections, Road, DisasterZone, and UnexploredZone objects, and under Map interface we created those methods whic could use these objects to get attributes information. So when loadMap method load a XML file to decode all the attributes, all the information regarding attributes could put into corresponding object and then be gotten correctly by those get methods.

- Robot Internal Representation
  The function is implemented by the other interface called RobotLocation. It is a interface defined all the methods describing the robot in the virtual map. More specfically, under its interface there are methods which can get the precise robot information from XML file and then set into virtual map so that we can see the robot sign in GUI and also there are methods which can set robot orientation and rotate the robot and finally return the current orientation of robot.

5 Design Details

5.1 Class Diagrams

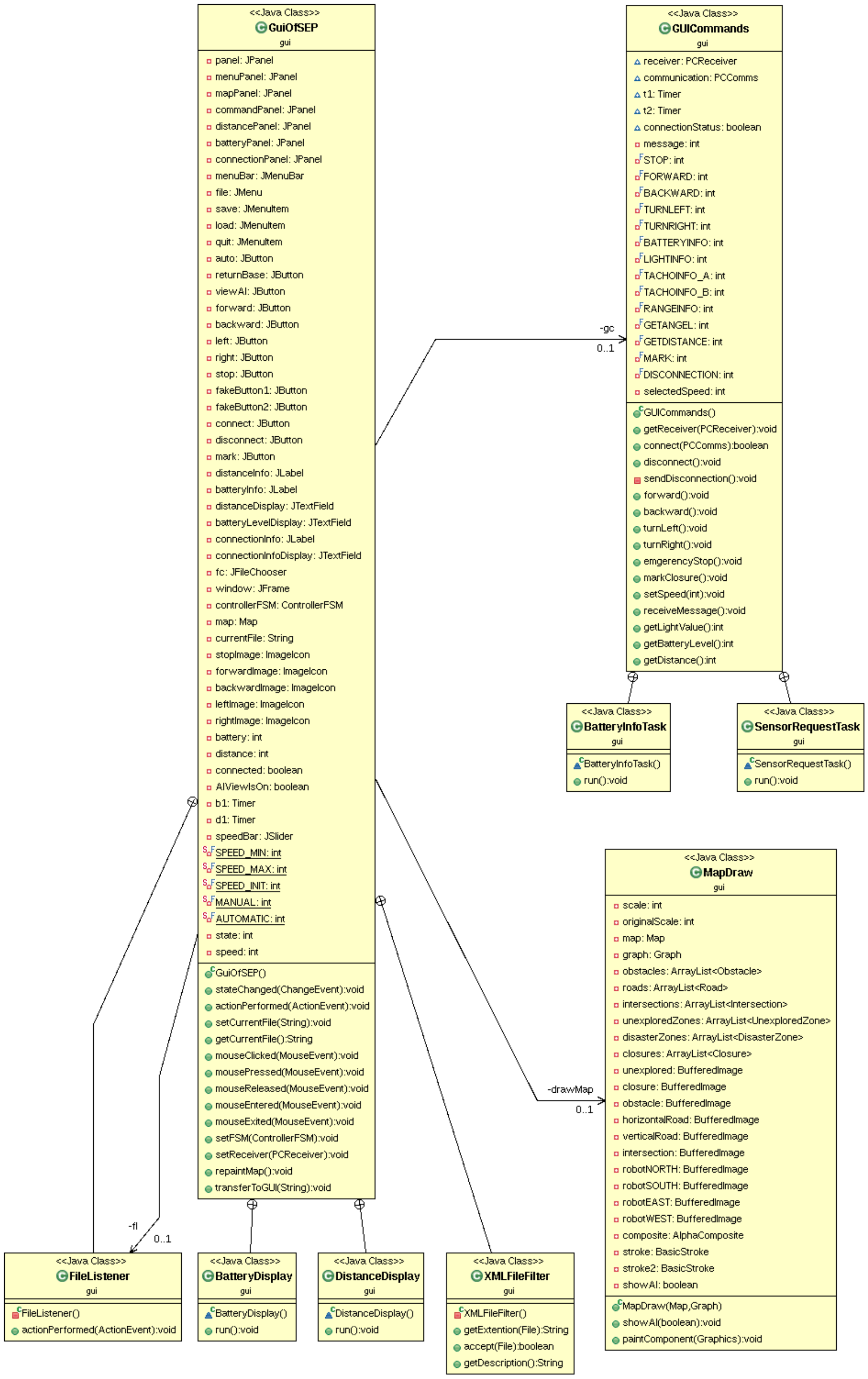


Figure 2: The GUI class diagram

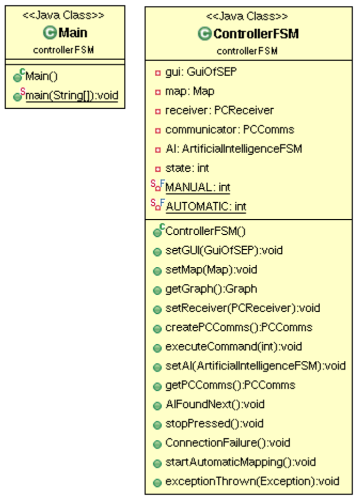


Figure 3: The ControllerFSM diagram

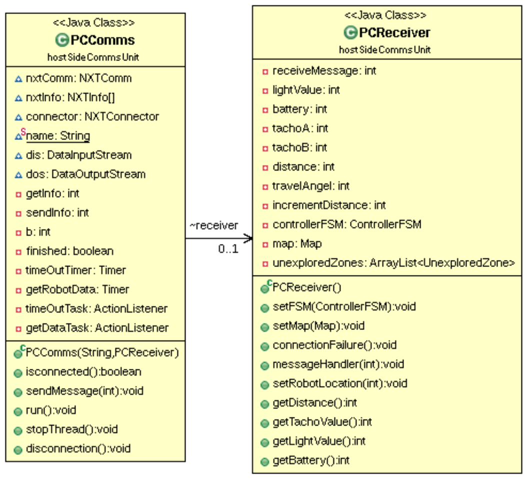


Figure 4: The host side diagram

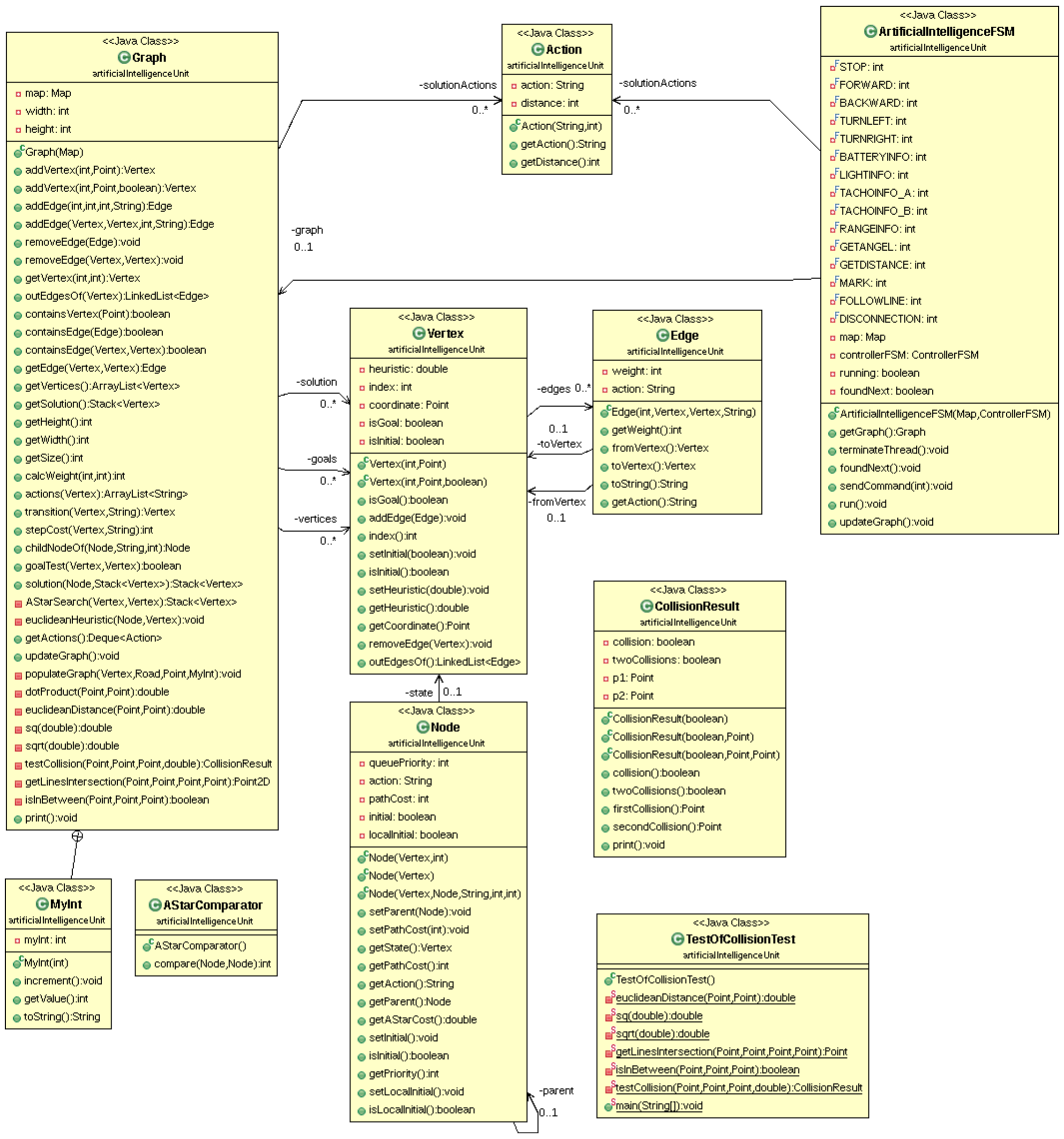


Figure 5: The AIUnit diagram

6 Human Interface Design

6.1 Overview of the User Interface

According to SRS, a GUI is essential on the PC side for the user to implement the following activities:

- The operator can establish communication with the robot by using GUI to manipulate the Bluetooth device.
- The operator is able to manually control and monitor the robot's movement by operating a set of buttons and seeing a map panel.
- The operator has the authority to command the robot to perform the AI mode by clicking buttons on the GUI, and then notice the robot status by receiving a bunch of

messages showed on the GUI.

- The operator is able to stop the movement of the robot by using the emergency stop button in the GUI whenever an emergency occurs.
- The operator can control the robot in some perticular extents to ensure the optimised task completion of the robot through the GUI, e.g. monitoring the battery life or adjusting the speed of robot.

The design of the GUI is in accordance with SRS purposing a simulation of the real world to provide visibility of system status to user. The GUI should be consistent and standardised to ensure user control freedom, error provention, safety precaution, and risk handling. To achieve the ease of using for users, the GUI should be flexible, efficient of use, aesthetically friendly, minimalist design, and smooth for control flow.

The development of the GUI follows the process below:

1. Relevant data gathering
   Infer information for GUI from requirements; analyse user habits, contorl flow, and environments; derive from initial strategy and data presentation.
2. Prototypes
   Create prototype based on the results of last phase.
3. Revisions
   Show the prototype to stakeholders (the group and client) for feedbacks and recommendations. If passed move to next stage otherwise design a new prototype or modify the existed prototype.
4. Documentation
   Create User Manual for the GUI.
5. Final review
   Final demonstration for assessment.

6.2 Detailed Design of the User Interface

According to the "User Interface" section of SRS, the GUI should consist of four parts:

1. Command buttons allow the operator to control the movement or change the status of the robot, including:
   (a) forward. Press once and the robot will continue moving forward.
   (b) backward. The same as forward.
   (c) left. Rotate the robot ninty degrees to left.
   (d) right. Rotate the robot ninty degrees to right.
   (e) connect. Establish the connection from PC to robot.
   (f) disconnect. Disconnect the bluetooth connection between PC and robot.
   (g) mark road closure. Command the robot to manually mark road closure.
   (h) start automatic mapping. Enable the AI mode of the robot to automatically explore the map.
   (i) return to base. Make the robot return to the starting position.
   (j) stop. The emergency stop for the robot.
2. Robot information area contains the information in relation to the robot's status, including:

(a) robot name.

(b) battery level. Display the current battery level of the robot.

(c) connection status. Display the status of Bluetooth connection using "on" and "off".

(d) message. A textfield shows the messages sent by the robot including values of sensors, status of the robot, and warnings.

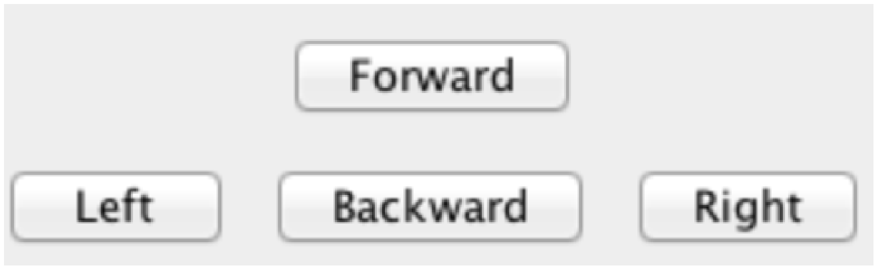
3. Map area
   A panel in the GUI to diaplay the current map. All objects defined in the DTD will be showed on the map, including roads, road intersections, obstacles, disaster area, closure, and unexplored area. The current location of the robot and traversed path by the robot would also be showed in the map.

4. List menu.
   A menu that allows the operator to save and reload the map in the form of XML file in the format specified by the DTD by clicking items in the list menu.

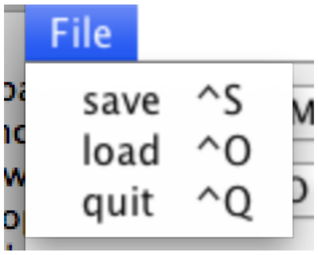
The display space for mapping is the major core component of the GUI, and it shall be maximised. All other functions except for the list menu are located at left-hand-side of the display window with a constant width so the display of map can be enlarged for larger screen sized window. The menu list is put on the top left of the window as a button on the top tool bar. The whole menu will emerge after a click on the menu button. Inside the menu list the user will see menu items to deal with the XML file.

The functions of GUI, which were designed upon the requirements elicitation, are outlined as following:

1. R01 and H02: Manual Control of the robot:

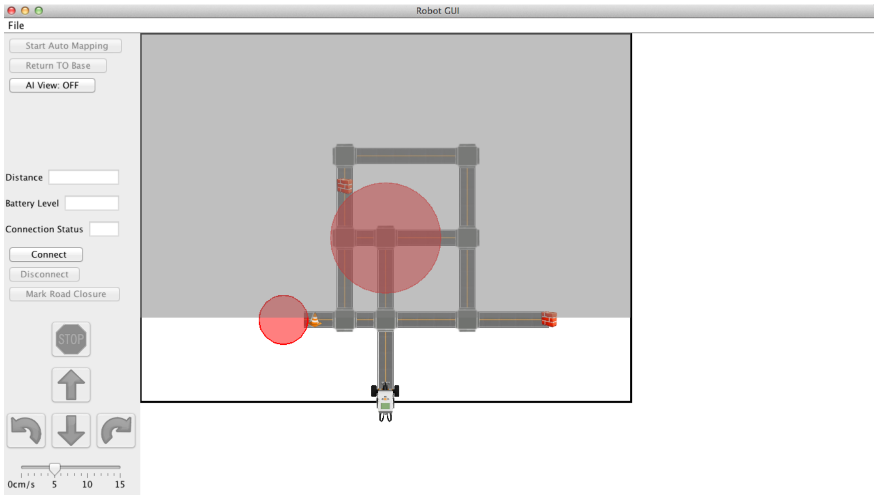


2. R02-03: Map saving and loading:



3. R07: Road closure marking:

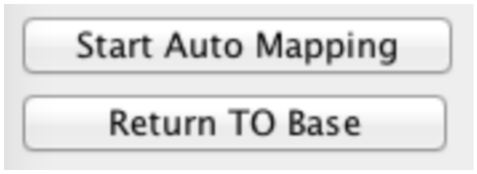
4. H01: The whole GUI:



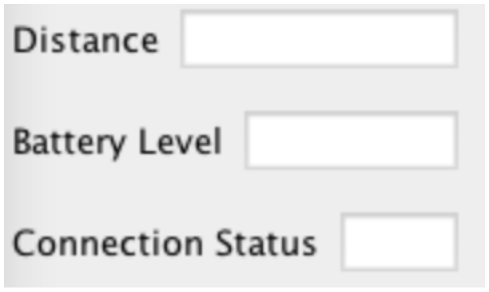
5. H04: Emergency stop:

6. Other attributes stated in other descriptions of SRS:

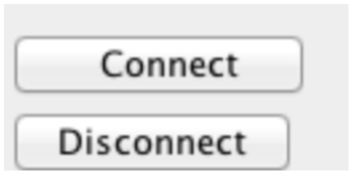
   AI mode:



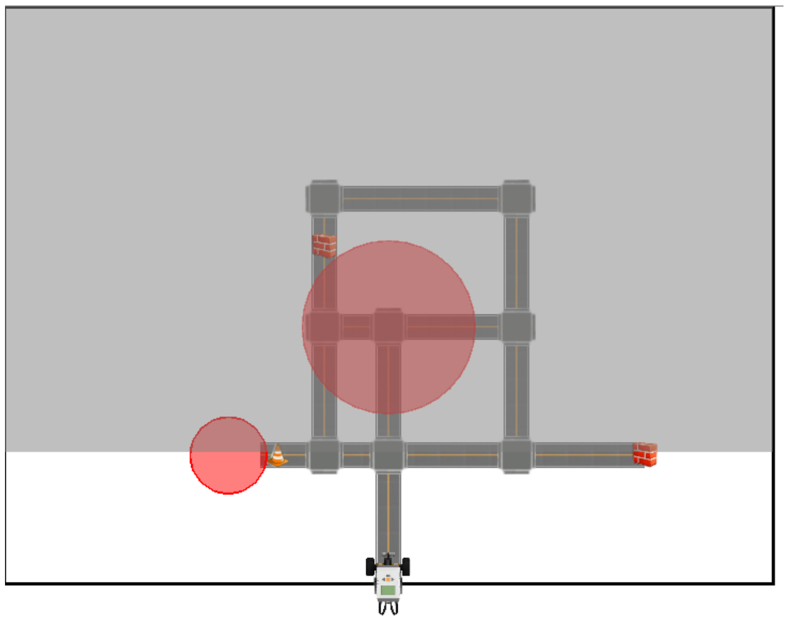
   Status:


Distance
Battery Level
Connection Status

Connection:


Connect
Disconnect

7. The display of the map:



The functions that have already been defined in SRS but not been implemented yet in the GUI, are outlined as following:

1. M01: Display the taversed path by the robot in map panel.
2. H03: Display the current position of robot in map panel.
3. H06: A textfield on the GUI to show messages such as alert message.
4. SA01: A speed bar on the GUI to adjust the speed of the robot. The speed should be within a safe speed.
5. Other function that was not described in SRS: Zoom feature for the map.

6.3 Graph Structure Used in AI

The data structure I wrote above will be utilized in AI to make robot explore in the map.Specfically, The Graph class will get data from the map data structure and then make a graph data structure. For instance, vertices are made from the objects such as intersections, closures and so on. And edges are made from the connections between them. Goals are made from vertices then the robot has to travel to, for example, obstacles.

7 Resource Estimates

The minimum hardware requirements are: Intel Pentium Dual-Core CPU E5200 with clock rate of 2.50G HZ; 3.7 GB Memory and 128GB Hard-disk Space.

8 Definitions, Acronymns and Abbreviations

API Application Programming Interface

Coding Pharaohs The software developing team undertaking the project

DTD Document Type Definition

GUI Graphical User Interface

JVM Java Virtual Machine

PIN Personal Identification Number

RCMR Road Closure Marking Robot

SDD Software Design Document

SEP Software Engineering and Project

SPMP Software Project Management Plan

SRS Software Requirements Specification

SVN Subversion

TBD To Be Determined

XML Extensible Markup Language used to store map information