Software Engineering and Project 2013

(COMP SCI 3006/7015)

School of Computer Science,
The University of Adelaide

Software Design Document

First Draft

SEP Group 2

September 20, 2013

Declaration

We declare that all material in this assessment is our own work, except where there is clear acknowledgement and reference to the work of others. We have read the University Policy Statement on Plagiarism, Collusion and Related Forms of Cheating: `http://www.adelaide.edu.au/policies/?230`

We give permission for our assessment work, including our subversion repository, to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted and retained in a form suitable for electronic checking of plagiarism.

Name	Student ID	Group Member Signature
Abdulaziz ALHULAYFI	a1642362	
Yu HONG	a1616861	Yu Hong
Jianqiu LI	a1635717	
Matthew NESTOR	a1132338	
Yifei PEI	a1611648	
Bowen TAO	a1622211	

The University of Adelaide
COMP SCI 3006/7015 Software Engineering and Project

Software Design Document (SDD)

for

Road Closure Marking Robot (prototype)

Version 0.3

Prepared by Group 2

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 implementation procedures of a 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 the closures, disaster area, the obstacles, the intersection and roads, and unexplored areas which includes anything I mentioned above. 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 systems 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.

1.3 Related Documents

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

fication (SRS).

1.4 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.

1.5 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.

References?
I'd recommend looking into different design patterns and citing them in your document. There are some great books in the library about this.

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.

Need to document the major constraints as these will influence your design

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.

This does it give me any idea about your software design. What is the relationship between host/robot? Any high level models you can reference here? eg. Slave/Master

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).

explain the pattern or cite reference.

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 Robot Side Communi-

Why do you use this pattern? This is not explained (see design rationale). Need to talk about ~~sort~~ trade offs etc.

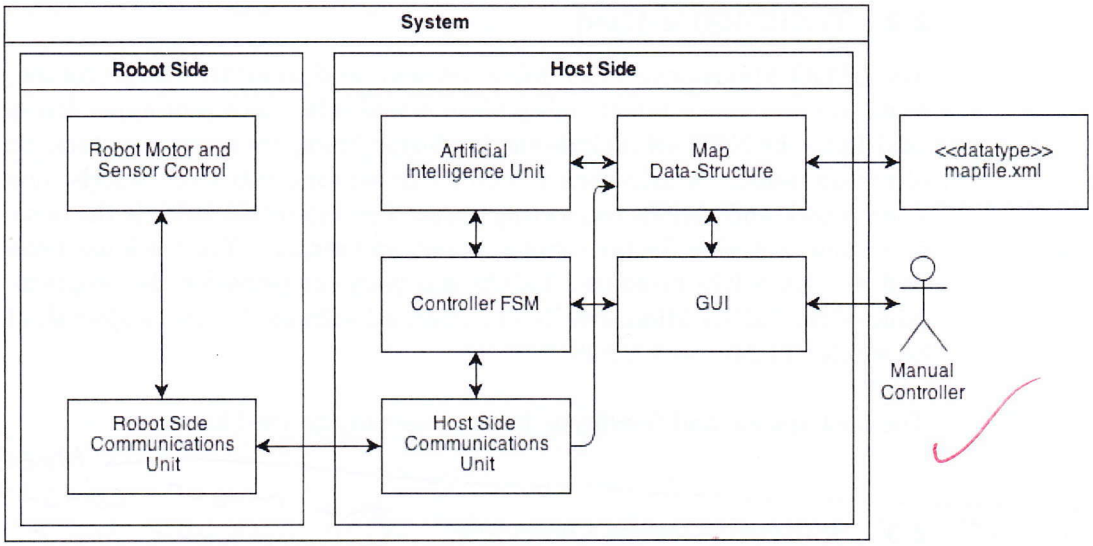


Figure 1: A graphical display of the system architecture

cations 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.

Good break down.

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

~~3.1 and 3.2 actually content is not structured quite right.~~
~~Either need~~ Need ~~to provide more detail of Set components~~
~~in 3.2. or move some of~~

In 3.1 you should provide a broad description of the System. Then in 3.2 list each provide a summary of the components. So some of the detail in 3.1 will need to move to 3.2. Then in 3.1 focus more on explaining architecture choice and what exactly pipe-filter and manager model are.

3.2.4 Controller FSM

Controller FSM contains only one component:

- ControllerFSM

3.2.5 Artificial Intelligence Unit

The Artificial Intelligence Unit consists of six components:

- Node
- Edge
- SearchTree
- Pathfinder
- Explorer
- AiFsm

3.2.6 Map Data-Structure

The Map Data-Structure has only one component:

- Map — More components here.

3.2.7 GUI

The GUI consists of two components:

- GuiOfSEP
- MapDraw

3.3 Detailed Components Design Description — So split into host and robot part.

3.3.1 SensorInfo

- Component Identifier: SensorInfo — It's useful to list requirement title and identifier.
- 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 (format ??)

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

- 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

~~Is this used to~~ should probably be a directed graph and not a tree

- 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

Map information shouldn't be coupled with Comms.

GUI and AI should interface with Map not the other way around

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.

Need to consider processing power and memory on the brick compared to PC. If AI on brick you need to store all info of city on there. It's better to have a dumb robot and leave the host in control. The host needs to store the map data to display it to the operator. If the robot is also storing the data then you have the data set in two locations and they'll need to remain consistent.

Good !! I should have read this before writing this

Also can always ensure you are in a valid state.

Should discuss alternatives to manager model and pipe-filter model too.

Need to talk more about trade offs.

4 Data Design

4.1 Database Description

Data Design does not occupy as much as other components in this project, as it has not many requirements elicitation, 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

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.

- Map Internal Representation
  The function is implemented by four main class in this project, the Draw class, the ReadMap class, the SaveMap class and the DataDetecting class.
  ReadMap, which is used to read the map file from XML format, absorbs the whole information on the map through loadBoundary method. More specifically, we created two-dimension array in terms of unexplored area, disaster, intersection, obstacle and closure and three-dimension array in terms of road. Then we put the information absorbing from XML file in terms of position and range (something like length, radius) into different arrays, waiting to be utilized in draw method.
  SaveMap, which is to save current map and the whole information on the current map. Through DataDetecting class, it would receive the latest information detected by the robot so that the data would updated on time and save into a new file. So if the information on the map has been modified, all of the modified things will be saved and also, the original is still there. For example, after exploring the unexplored area by the robot, the more roads, obstacles, intersections and so on will be displayed on the map, and all of them can be saved as a new file at any time, distinct from the original file.
  DataDetecting, which is to store the information detected by the robot. It also created a variety of two-dimension arrays in terms of road, obstacle, intersection, closure and disaster to save the value which is modified due to the detection behaviour of the robot, waiting SaveMap class to utilize.
  Draw, is to draw the original map in the GUI. The class includes paint

method to draw things. And different things will be drawn by different colors so that everyone can distinguish them easily. More specifically, it gets data from different arrays in the ReadMap class and then draw the map followed by those data. In addition, the red circle is disaster, green point is intersection, magenta line as road, orange point as obstacle, and gray point as closure.

Should talk about graph structure used in AI

5 Design Details

5.1 Class Diagrams

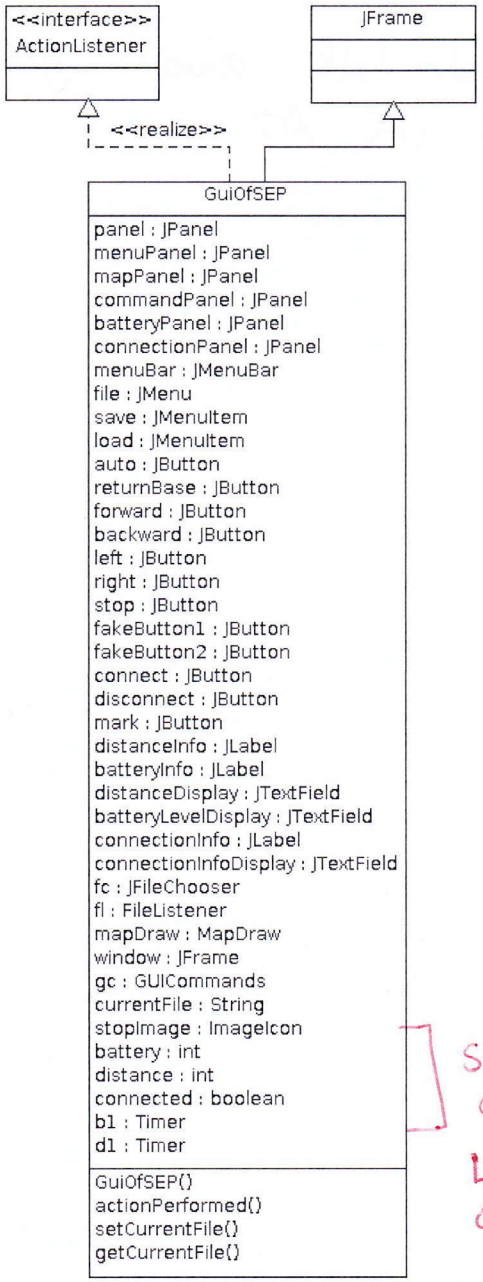
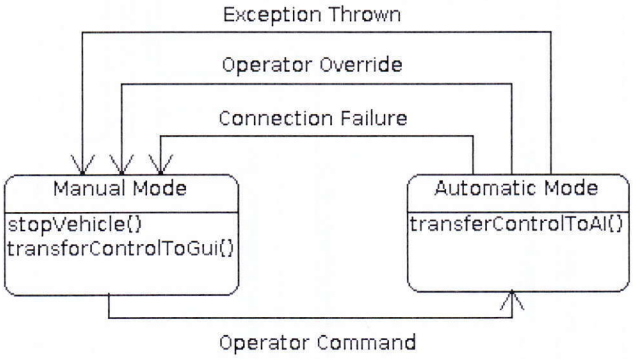


Figure 2: The GuiOfSEP class diagram

The rest of your class diagrams? ---

5.2 State Diagrams



starting point?

Figure 3: The ControllerFSM finite state machine diagram

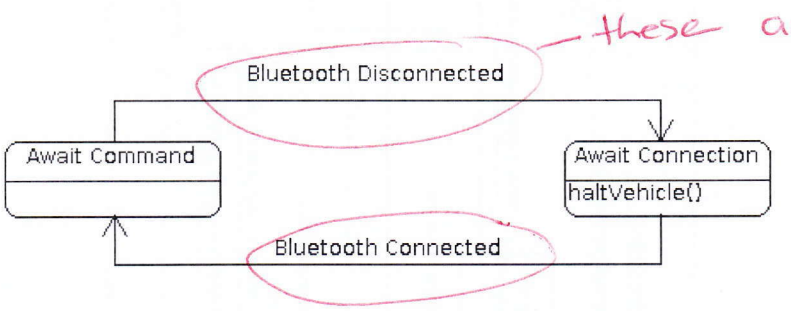


Figure 4: The robot side finite state machine diagram

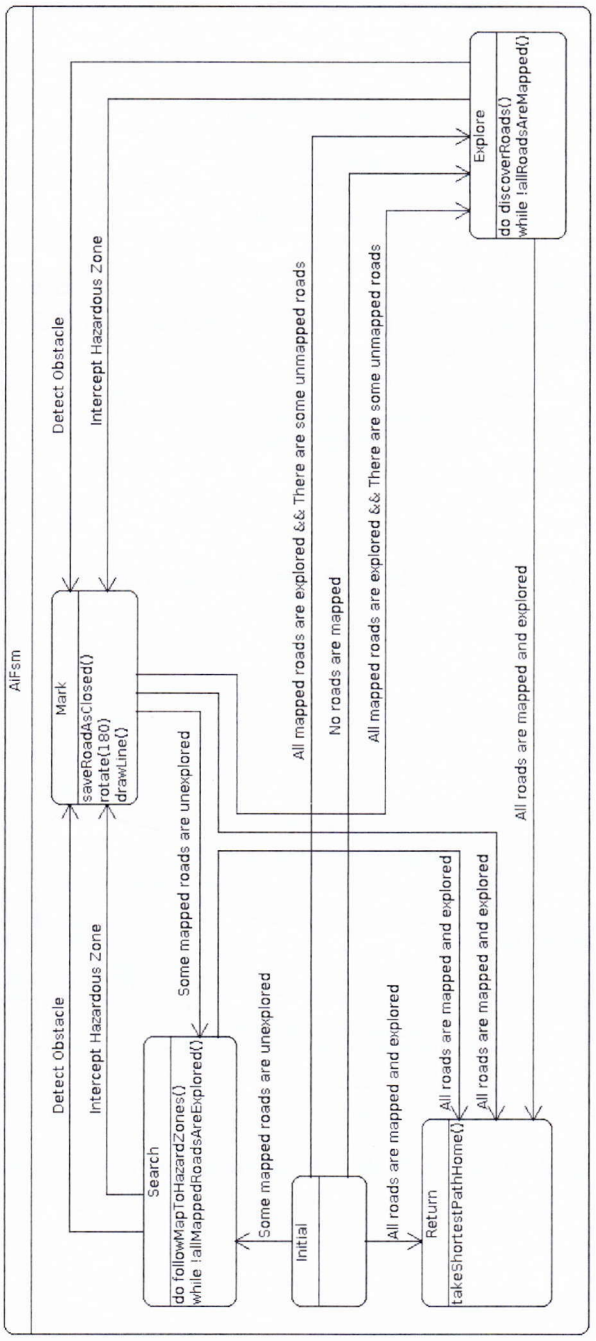


Figure 5: The AiFsm finite state machine diagram

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:

Sounds like something for SPMP

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:

Don't need to restate SRS.

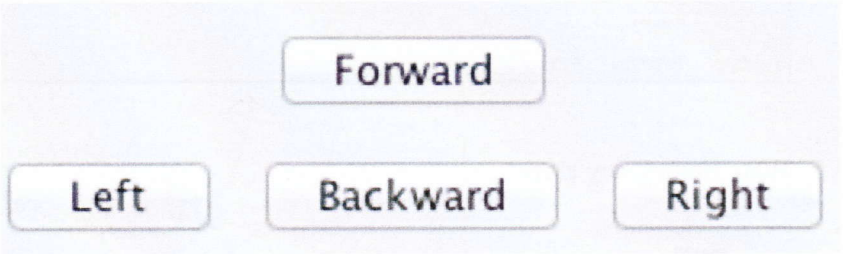
1. Command buttons allow the operator to control the movement or change the status of the robot, including:
   (a) forward. Press once and the robot will continue moving forward.
   (b) backward. The same as forward.
   (c) left. Rotate the robot ninty degrees to left.
   (d) right. Rotate the robot 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.

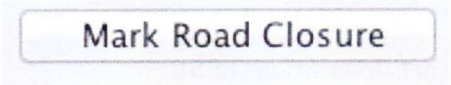
The display space for mapping is the major core component of the GUI, and it shall be maximised. All other functions except for the list menu are located at left-hand-side of the display window with a constant width so the display of map can be enlarged for larger screen sized window. The menu list is 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:

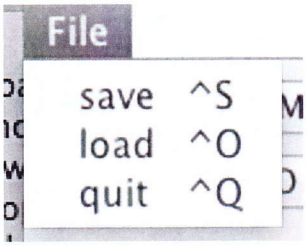
1. R01 and H02: Manual Control of the robot:



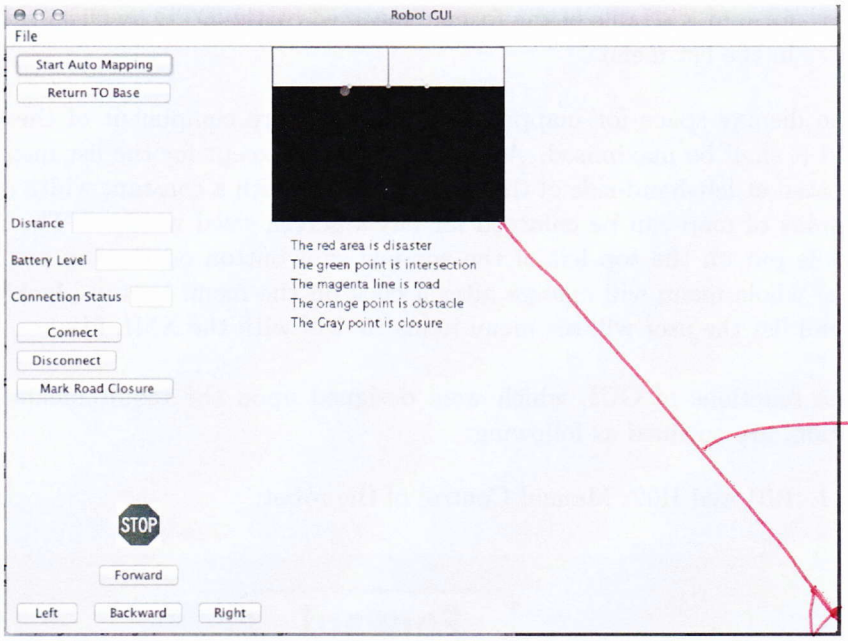
2. R02-03: Map saving and loading:



3. R07: Road closure marking:



4. H01: The whole GUI:



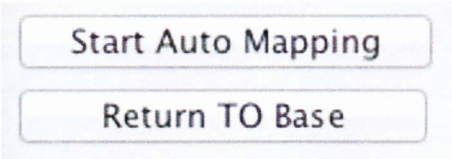
Map should fill the screen

Should show several screens of entire UI in different states and highlight ~~the~~ and describe how the state changes

5. H04: Emergency stop:

6. Other attributes stated in other descriptions of SRS:

   AI mode:

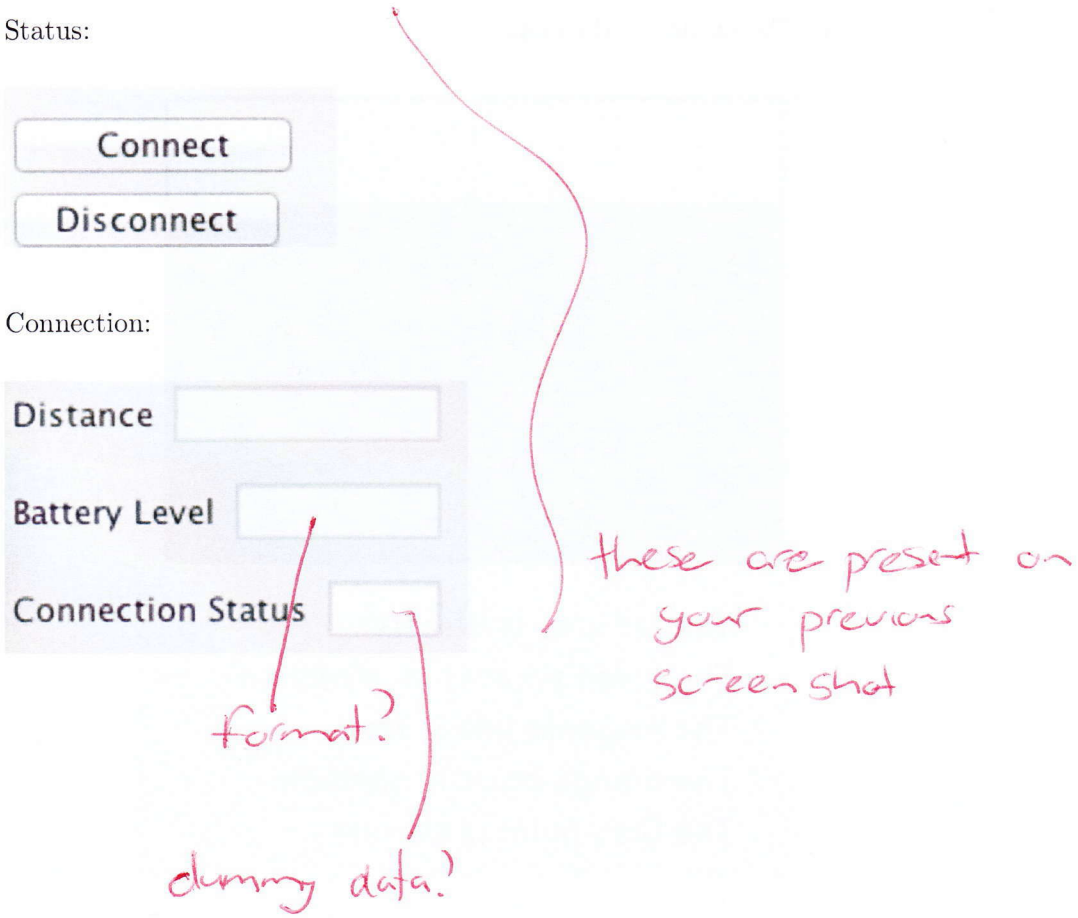


eg. This is the screen when you load up the program.

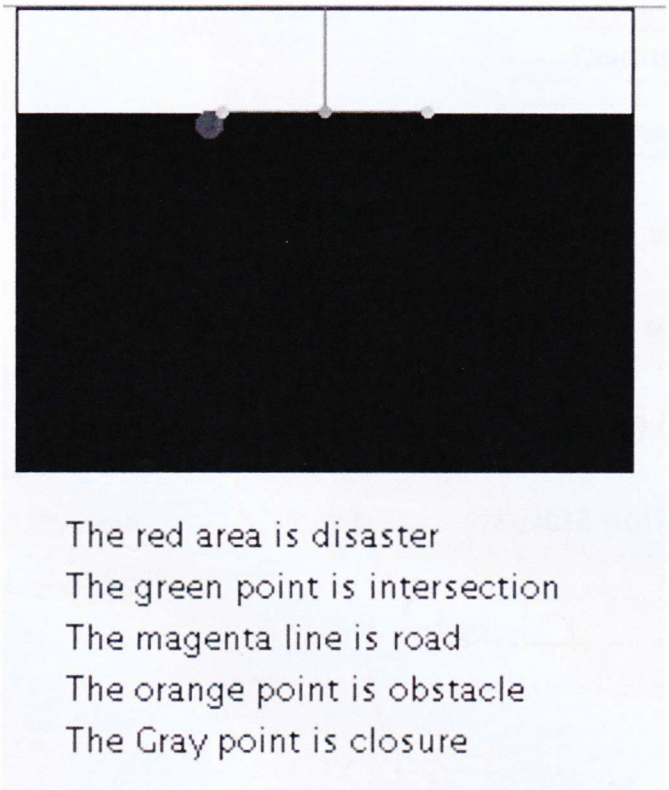
When loading a new file you see this - - - etc

Step through use cases

As you do this you will cover the requirements


Status:
Connect
Disconnect
Connection:
Distance
Battery Level
Connection Status
these are present on your previous screen shot
format?
dummy data?

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.

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.

To store your program? only list HDD space required for program

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