2007

RoboTotal War

War Simulator Requirement Analysis Specification Report

This document is prepared for Software Engineering Course project; War Simulator. In this document you will find the first step of our Software project: The Requirement Analysis Specification.

Erol TOKALAÇOĞLU
Osman A. BIÇAKCI
4/18/2007

Table of Contents

1. Introduction

RoboTotal War is a war simulator that enables to the users in first step establishing war simulations as defining robot teams and the environment that the robots fighting. For the future aspect we try to add the environment step that users also write their own robots that are extends our ancestor robot that has some specifications for adapting the environment that we defined before.

1.1.Purpose:

This document provides detail surrounding the functionality of the RoboTotal War simulator software. Furthermore, the functional and non-functional requirements are discussed in depth. The document illustrates the functionality of the software through the use case diagrams. System constraints and required software and hardware are presented emphasizing the contribution to the system as a whole. Finally, a time plan for design and implementation is provided.

1.2.Scope:

RoboTotal War is mainly working by computer itself but adjusting the teams, maps, conditions gives user to effect game. In the later steps, adding own robots and the learning robots gives user to fight his own robot with others. The subsystem is the environment that enables the user to write the java code with extending our robot class or for security reasons just adding the library that we defined for this project. This project gives chance to users that they can write their own robots with AI for better wars, as like robocode but the difference between our platform and robocode is we define the rules for robots and we give some specifications and and the limits are restricted.

1.3.Definitions, Acronyms and Abbreviations:

RoboTotal War simulator we define first an environment that enables user to fight the robots and we give up the name EnWar for the environment. We have types of robots, and every robot has a defined specific features and for future aspect user can define more within the limits and environment rules. We will use an IDE (INTEGRATED DEVELOPMENT ENVIRONMENT) for designing GUI(GRAPHICAL USER INTERFACE). Mainly we think to write our code in JAVA Language for using the features of OOP(OBJECT ORIENTED PROGRAMMING) that enables us to extend new robots from the ancestors and for user part for future user can also define his own robots as like we do now. Also Java gives us the chance to run our environment in all OS(OPERATING SYSTEM) platforms as like LINUX, WINDOWS, SOLARIS, MacOSX etc. For future user can easily understand our platform with using API(APPLICATION PROGRAM INTERFACE) that enables user to see which classes we have and which relations that they have.

1.4.Overview:

Section 2 of this document provides an overall description of the Software system including the main functionality and ideas.
Section 3 illustrates the main functionality of the system using several simple activity diagrams and use case diagrams.

Section 4 explains the functional requirements of the system in detail grouped in order of importance by each of the components mentioned in Section 3.
Section 5 presents the non- functional requirements of the system. Section 6 User Interface that we develop. Section 7 contains the constraints we would be having in the making of our system Section 8 lists the hardware and the software to be used. Section 9 finally has the references we have used in our research followed by the glossary written.

2. Overall Description:

2.1. Overview of the System:

Our environment works on JAVA. The system includes two steps: first is user interface to adjust the specifications for the game and the second is starting the game. The game means autonym working robots that are hard coded by us and for future aspects we try to give chance to end user for designing own robots but our project in first step gives user to adjust conditions, robots, maps etc. before starting game.

2.2. Thesis Statement:

RoboTotal War simulation software's main aim is to bring end user to define own robots within the limits of our environment. That means end user will use own hardcoded robots, as like using the AI algorithms but actually in the growing procedure of the system shows that the first steps of the environment required a unhandled robot environment. End user uses only the robots that we define for simulation. But Adjusting is also a big constraint for the end user to change the way of the war.

2.3.Product Functions:

The product functionality can be divided according to the components that make up the system which include the user and the robot component.

User Functionality: User can compose the system and adjust the features of the game simulation. Can either handed the general options and game options, will start, save, open the simulator.

Robot Functionality: System gives robots to fight automatically but the before starting the game they are hardcoded. The user handle them by GUI and this changes are mainly seem in simulation.

3. Use Cases and Activity Diagrams:

3.1. End User Use Case:

End user can adjust the environment for the next game. The features of the main functionality can be adjusted by the Simulator options menu and the game options are seperated mainly 4.

1. Game general options
2. Match Type
3. Map Options
4. Robots and creating Teams

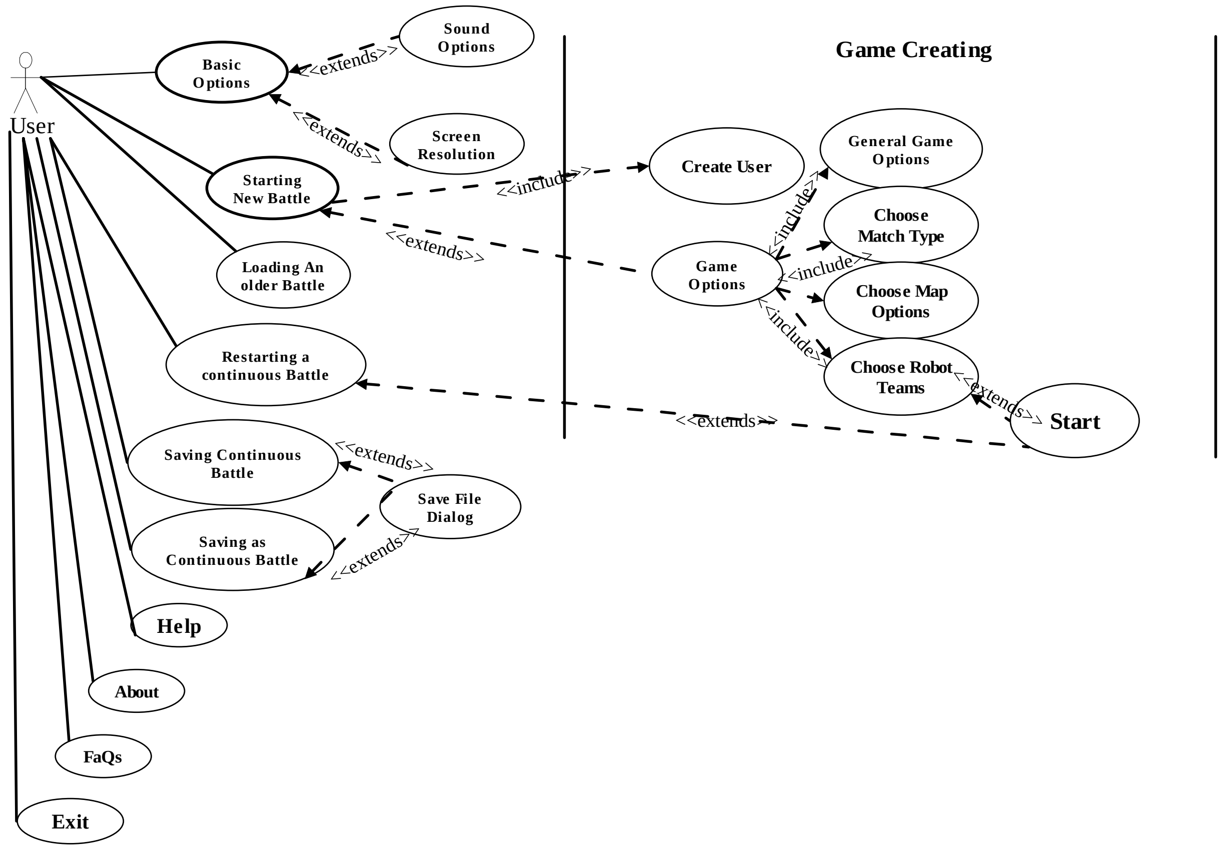


Figure 3-1: Use-Case of User

3.2. Robot Use Case:

Robots have lots of functionality. They are hardcoded system droids that they work by themselves. User only effect the conditions, team members, the guns, maps etc.

The other parts are worked by the robots and their functions are defined as like below use case.

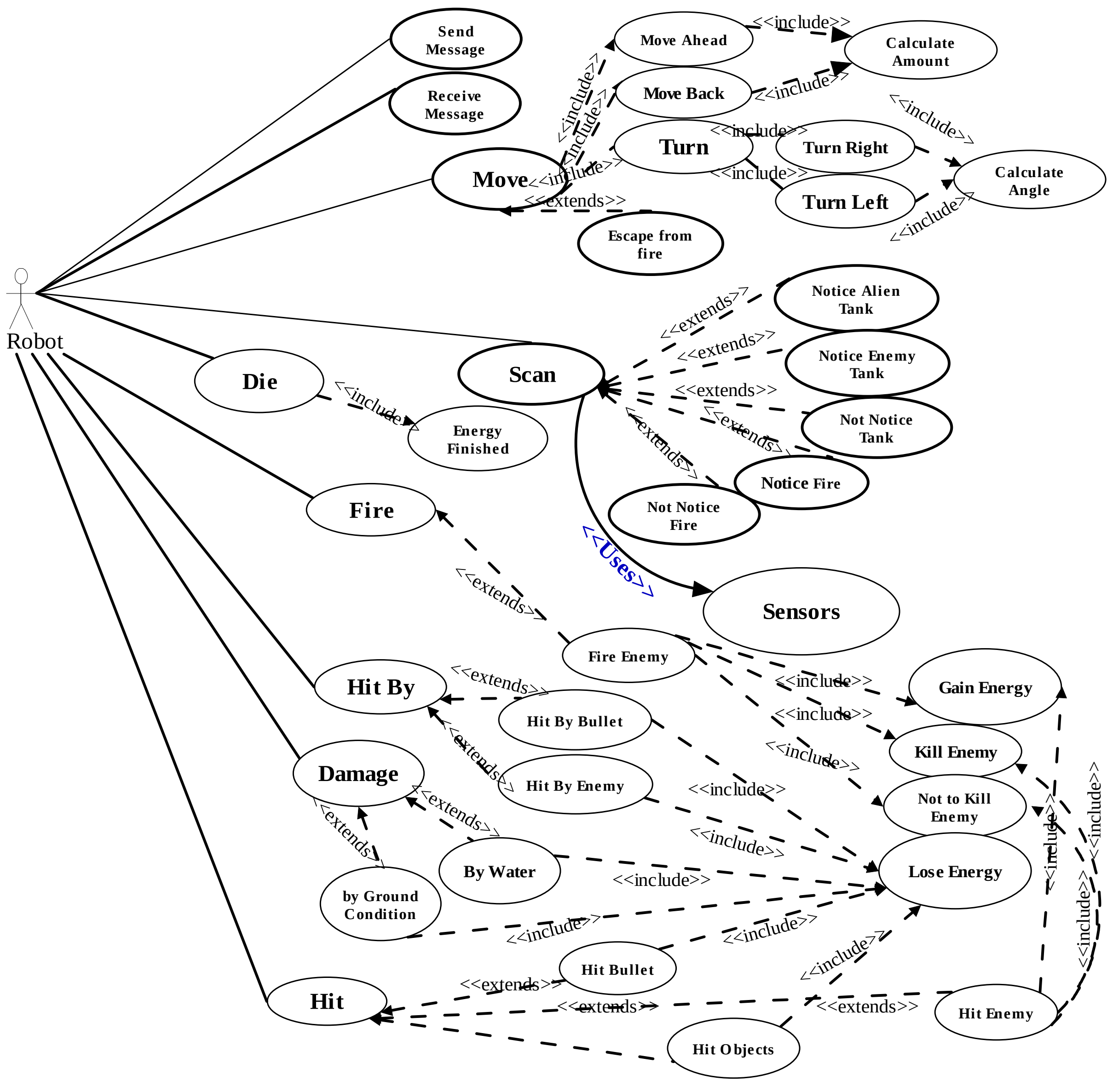


Figure 3-2: Use-Case of Robot

3.3.User Activity Diagram:

User can control the main environment. So user has the ability to create game, open game, restart game, save and save as game, exit game and adjust the main functions for the environment.

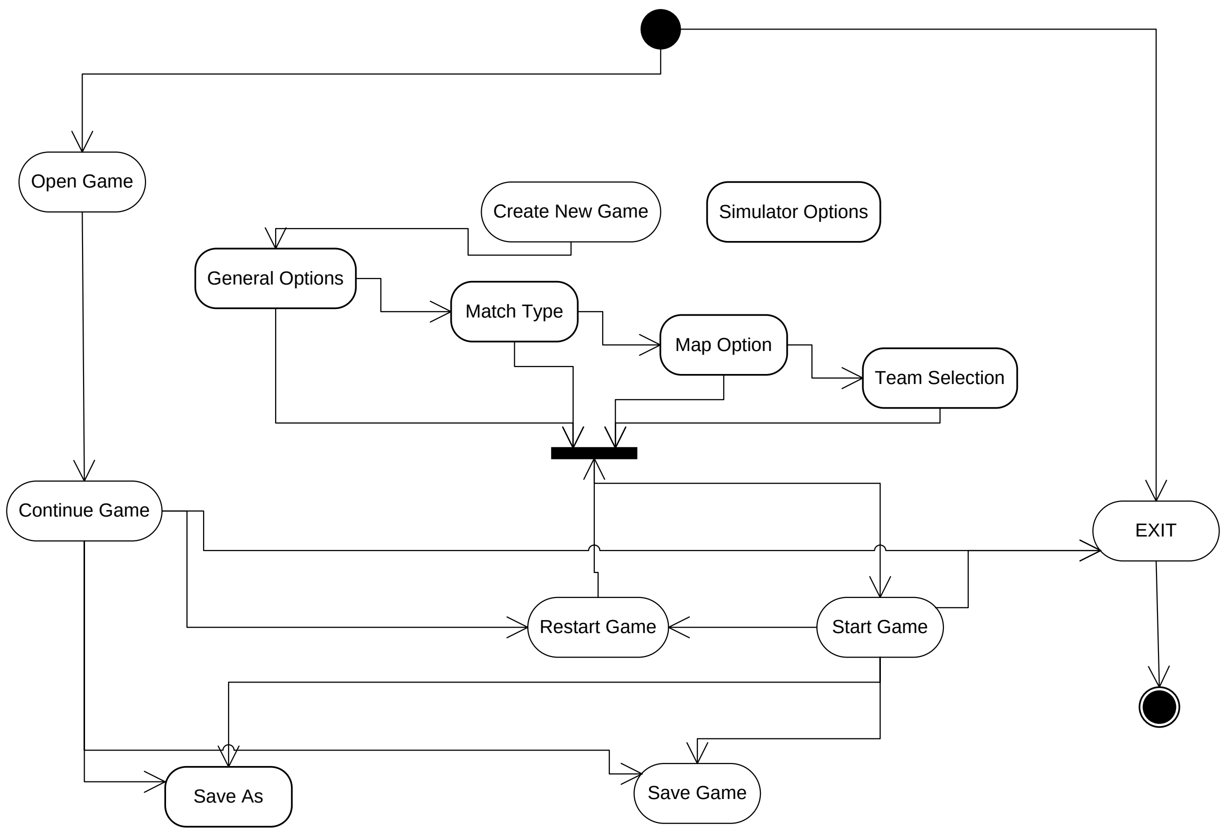


Figure 3-3: User State/Activity Diagram

3.4. Robot Activity Diagram:

This is the lifecycle of the robot. Robot has the ability to work by itself but the communication skills are also added by the environment. After user adjusts robot fight in the conditions of the environment. The rules are defined by Robot class and this is our ancestor robot it has the defined and non changeable functions.

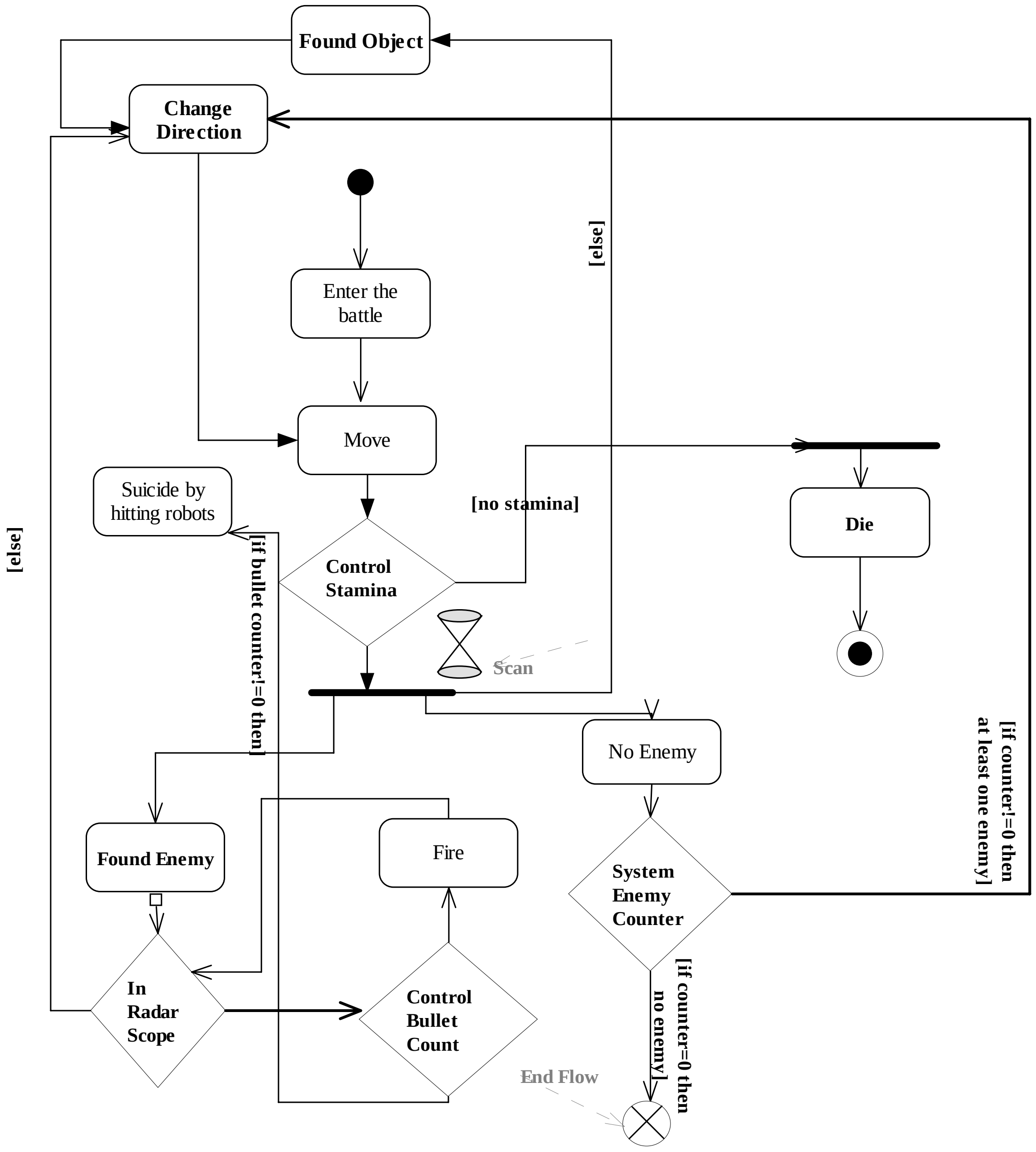
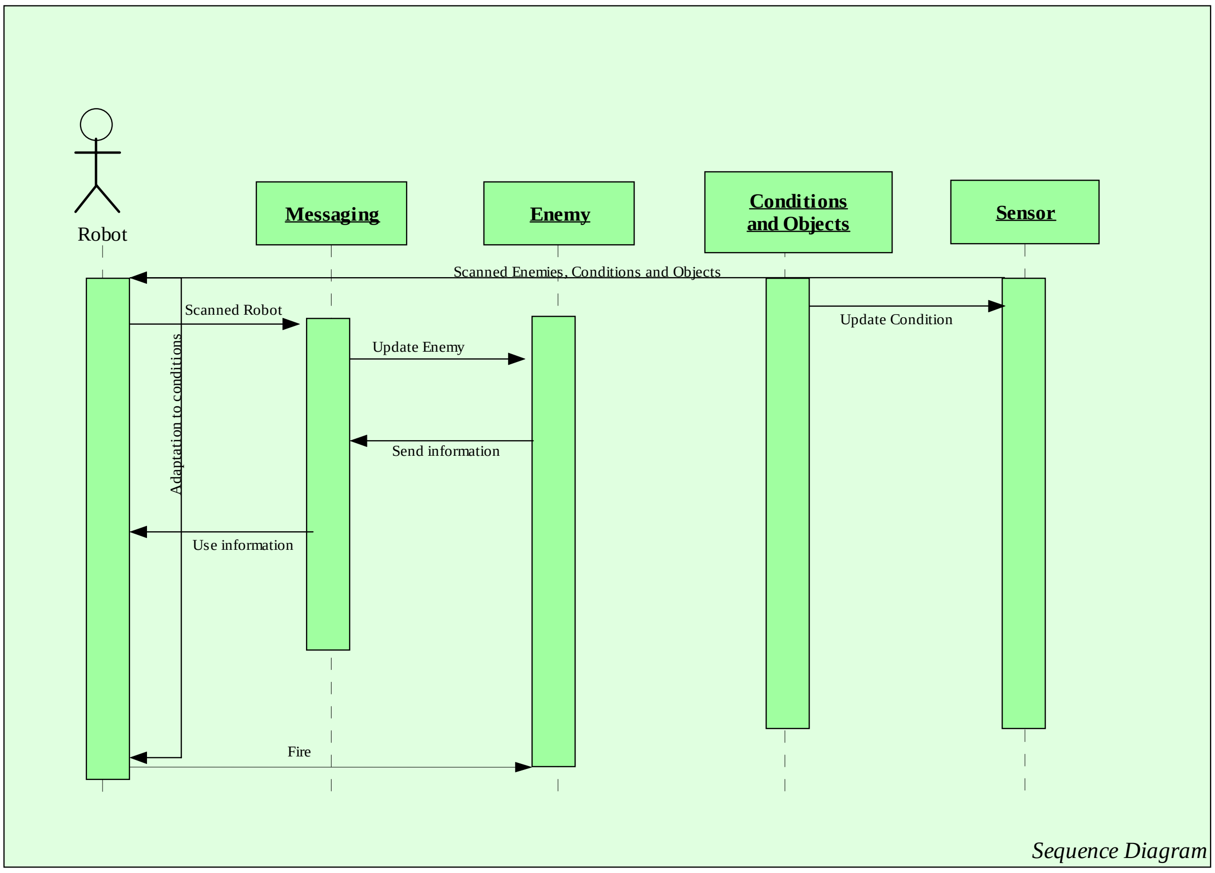


Figure 3-4: Robot State Diagram

3.5.Robot Sequence Diagram:

This is the timeline and objective sequences throughout the robot.

4.



5. Figure 6-1: Sequence Diagram

6. Functional Requirements:

6.1.1. Environment:

6.1.1.1. Board:

6.1.1.1.1. Borders:

There are no exact limits that they are seen but actually there are limits for extreme conditions to stop escaping or abnormal actions.

The limits are seems as like rivers, mountains or other enviromental limits that the robots can not break the limits in real too.

6.1.1.1.2. Shape:

The board is in a undefined shape as a rectangle, square or oval but the shape is in a map format as a region, country or something like that. The size of the shape also varies to condition of the war if a robot team starts to escape than the active shape engine stops to create map parts to limits the board.

6.1.1.1.3. Surface:

The surface is in 2D so there are no 3D objects in the surface but there are mountains, roads, rivers then the type of the surface will be vary as like: soft sand, stone, mud, dry and hard sand, water, swampy, sloping areas.

6.1.1.1.4. Topology:

The shape of the map and the things on the map are whole form topology. The surface area is 2D but actually the effects over the enviromental things are the same as real so there is a topological surface.

6.1.1.2. Conditions:

6.1.1.2.1. Light Condition:

Light conditions are in general two types as day and night but also we will add the different light conditions to effect the robots' vision features oriented with weather conditions; moistured weather, cloudy weather, semi cloudy etc. Means there are four types of light levels.

6.1.1.2.2. Weather Condition:

Weather conditions are in general sunny, cloudy, semi cloudy, rainy, snowy, foggy, tornado, sandstorm.

The weather condition effects the robots as like enviromental conditions, block or dicrease the attacks etc.

6.1.1.2.3. Ground Condition:

Oriented with weather conditions the types of ground will be varied, if the weather is rainy then the soft sand is going to be swampy or if the weather is snowy then the mountains are going to be block the roads to go somewhere.

6.1.2. Capabilities:

6.1.2.1. Fire:

6.1.2.1.1. Types:

In a instance there are 3 types of fires, but actually there are some bonus fires that the robots should get them from temporary packages.

The instance types of fires are;

- **Bullet (large area, small effect, low cost)**
- **Laser(large area, big effect, highest cost)**
- **Plasma (medium area, big effect, high cost)**

6.1.2.1.2. Damage Types:

The main idea should be defect the enemy robots. The damages should be in the types of;

- **Only one robot can be effected**
  - **Should lose power**
  - **Should be block,stop for time instance**
  - **Should Lose some properties such as shields, some fire types etc.**

6.1.2.1.3. Range:

There will be possible ranges for every fire type, for instance a bullet will effect a wide area range but maybe it should have a low effect or a laser has a small area range but its effect should be more than a bullet.

6.1.2.1.4. Cost of Fire:

There will be types of fires then the effects of these fires should not be the same also now we should know that the cost of fires must be different.

6.1.2.1.5. Speed of Bullet:

There must be a speed of a bullet, to catch the robots because every robot has a speed of motion and there will be fires to effect them or for escaping skills there must be a speed it also depends on the types of fire.

6.1.2.2. Motion:

6.1.2.2.1. Angular:

Robots have the capability to move with a angle in 2D plane, it helps them to find any object or sensing all map view.

6.1.2.2.2. Linear:

Linear motion is the capability to move in a linear way.

6.1.2.2.3. Speed of Motion:

The motion has a speed, it also depends on the power of the robot. Speed is very important for important time slices for instance enemy may fire and robot must escape very fast or vice versa. Or enemy attack to collegues than robot have to go to help collegues than time is important and speed is the answer.

6.1.2.2.4. Distance:

Robots are fighting in a 2 dimensional plane so they have to move from a point to another point that has a vectoral amount means a distance. We will calculate the distance as using vectors.

6.1.2.2.5. Cost:

For every motion, robots lose power, this means robots pay for the motion as like fire and they have power limits so they must spend the motion power in optimum way also the costs depend on type of robot, type of the way to the target(mountains, rivers has may spend more etc), the total distance etc.

6.1.2.3. Endurance:

6.1.2.3.1. Health (Power Level):

We have two ideas for power level:

- Team Power :
  - The total power is the same for every team in start condition
- Individual Robot power:
  - For every robot the power level may be vary so for a defence robot the power limit may be low for an attack robot it will be more also we have an idea to transfer power to higher to lower for adapting to tactics.

6.1.2.3.2. Shield:

Every robot has some basic shield in the start time of the game such as rainy shield, in some different levels of power shield etc.

Also with getting packages robots can get more and different shields (Armor, health, noise, water resistance, ground adaptation shields etc.)

Shields are moving also within a team members also for every death robot lose its shields and this shields can be taken by murder or anyother robot.

6.1.2.4. Sensing:

6.1.2.4.1. Vision:

With vision sensing robots can differ the enemy, collegues or objects. Robots has a limit of vision sense for instance just a limited angle of visibility they will have, or in bad weather condition or no light conditions they can not see. So there must be some other sensing technics.

6.1.2.4.2. Motion:

Robots shall sense motion for a range we learned that some conditions the vision sensing technic can not work so the there are some other technics one of them is motion sensing.

Motion sensing has the range, in a limited range robot has a radar for sense the things that they are in motion, then if the vision sensing technic works for differ the object.

6.1.2.4.3. Fire:

Fire sensing is another sensing technic that it works for a wide area range to get sense for war condition, for instance some collegues are fighting and their communication skills are doesn't work than the only way to find them is fire sensing.

6.1.2.5. Messaging:

6.1.2.5.1. Location:

Robots give each other the location infos about enemies and their own maybe infos about the topology too.

6.1.2.5.2. Sensor Info:

Robots have lots of sensors to get info about fire, weather condition,ground condition.

6.1.2.5.3. Range of Messaging:

Communication property is using in a wide area but it also should have a range, for some condition SOS call shall not have a range.

7. Non-Functional Requirements:

Heterogeneity: Types of Robots determined by capabilities. In general there are 6 types of robots where each type is dominant of one capability. The total power capacity of the teams are the same, but it will be distort to all robots not in the same level.

Objects: Objects which fixed at the first of the game. Rivers, mounts, deserts and other geographical objects can be defined as fixed object. These objects can be attached to the defined topologies.

Size of Board: The size of the boards will be varied as oriented with the number of robots in a team for instance;

- Number of robots is 50 then the size of the board should be medium.
- Number of robots is 20 then the size of the board should be small.

Total Fire Ammunition: In the start condition every team should have a total Fire Ammunition limit then for every fires the Ammunition should be decrease and the level of decrease is the cost of fire for instance a laser has big effect and wide are range than it should have a big cost as 20 times cost of a bullet etc.

8. User Interface Requirements:

Our environment uses JAVA so we use our own User Interface this includes a main menu and some sub forms to adjust and in general view of the system has a map view that has the resolution and can be adjusted by the user.

9. Dependencies and Constraints:

- Our platform contains an area that we defined the map where our hardcoded robots can battle against each other.
- Each Robot has similar and different equipments also they are changeable by the user before starting game. User can limit the gun, health and decide to play in this conditions. The equipments are mainly, radar, gun, moving ability, energy tank etc.
- Robots have energy limits and this will be defined by user before game and when the energy becomes zero the robot dies.
- Robots have ability to move. Moving is mainly 4 functions moving back, forward, turning left and right.
- Robots have ability to fire the enemy if it is in the range of the gun.
- The enemy can be scanned by the robot within the radar range.
- User only affect the game before starting game as like adjusting the conditions, robot selections, map options.
- Player define the robot number within the map limits

- Player define the total health, total gun power and max for each robot.
- Player will change the system sound, resolution, graphics.
- Player can restart, load, save, save as the game.

10. Hardware and Software to be used

10.1. Hardware:

- PC
- Sound system for sound effects

10.2. Software:

- Java Runtime Environment (at least JRE 1.4.0)
- OS (Platform Independent so LINUX, SOLARIS, WINDOWS etc.)

11. References:

Books:

[1] Software Engineering, Ian Summerville, 8th edition

Web Pages:

[1] http://www.java.sun.com

[2] http://robocode.sourceforge.net

[3] http:// robowiki.net/cgi-bin/robowiki