ETSA02-SRS-BLB

SOFTWARE REQUIREMENTS SPECIFICATION

for

Basic Leader Bot

Version 0.1 approved

Prepared by Markus Borg
Dept. of Computer Science, Lund University

December 7, 2017

Contents

Revision History

Name	Date	Reason For Changes	Version
Markus Borg	2017-12-07	Initial draft.	0.1

1 Introduction

1.1 Purpose

Basic Leader Bot describes a robot that can be a part of team matches in RoboCode.

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>

1.2 Document Conventions

In the remainder of this document, we refer to Basic Leader Bot as "the robot".

This document is organized according to the ISO/IEC/IEEE template X...

This document uses the term "quality requirements" to specify operational requirements on system behavior. Other synonymous terms include "non-functional requirements" and "extra-functional requirements".

Section 4, System Features, constitutes the backbone of this document. The section is organized around high-level features that are briefly described and complemented with their respective stimuli, i.e., the event that trigges the execution of the feature. Each feature is further refined into detailed requirements of one or more of the following types: 1) use cases, 2) functional requirements or 3) quality requirements.

Use cases are used when the robot shall conduct a sequence of steps, and there are explicit exception cases that needs to be specified.

All quality requirements are explicitly categorized according to ISO/IEC 25010, specified with capital letters in brackets. Moreover, each detailed requirement has a unique identifier according to the below formats:

- UC-<Feature Number>-<Running number> for use cases
- FREQ-<Feature Number>-<Running Number> for functional requirements
- QREQ-<Feature Number>-<Running Number> for quality requirements

Using the identifiers, each detailed requirement can be traced both to the implementation in Java source code and the test specification

In the document, text in bold font is used to indicate the priority of a feature. Unless otherwise stated, the priority of a feature is inherited by the detailed requirements.

Terms expressed in italic font carry a particular meaning. All terms in italic font are defined in the Glossary in Appendix A.

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

1.3 Intended Audience and Reading Suggestions

The document constitutes an example of a software requirements specification for RoboCode robots. The primary readership is ETSA02 teams from two perspectives: 1) developing their own robots to offer on the RobotMarket, and 2) designing a robot team for the ETSA02 RoboRumble. In particular:

- ETSA02 Requirements engineers (learn what a SRS could look like)
- ETSA02 Domain experts ()

Secondary stakeholders include:

- ETSA02 Test managers (studying what testable requirements can look like)
- ETSA02 Project Supervisors

Tertiary stakeholders include:

- ETSA02 Sales engineers
- ETSA02 Project managers
- ETSA02 Development leads
- The general population of RoboCode users
- University teachers (to find inspiration for software engineering teaching)

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

1.4 Project Scope

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>

Table 1.1: Intended readers of ETSA02-SRS-BLB.

Stakeholder type	**Goal when reading document**	**Recommended sections**
ETSA02 Requirements engineers	Learn from an archetypical Robot SRS	All document
ETSA02 Domain experts	Considering robot purchase for own team	
ETSA02 Test managers	Learn from examples what testable requirements could look like	Section 4
ETSA02 Project supervisors	Support project grading by studying an archetypical Robot SRS	All document
ETSA02 Sales engineers	Adapt marketing strategy based on basic robots available on the market	
ETSA02 Project managers	Obtaining a general understanding of an available basicrobot	
ETSA02 Development leads	Preparing development activities based on an archetypical robot SRS	
Robocode users		
University teachers		

1.5 References

ISO/IEC/IEEE Template
ISO/IEC 25010
ETSA02 RoboTalk Protocol

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

2 Overall Description

2.1 Product Perspective

Basic Leader Bot constitutes a leader robot that can be used in robot teams in the ETSA02 RoboRumble. The robot lacks any offensive capabilities, instead it is entirely designed to broadcast information about enemy targets using radar actions. The robot is not completely stationary, however, as it moves a short distance forward and returns after completing its radar action.

The robot is available as open source as "MyFirstLeader" in the RoboCode sampleteam folder, from which this Software Requirements Specification has been reverse engineered. As all RoboCode robots, the Basic Leader Bot consists of a body, larvfötter, a turret, and a radar dish (see Figure X). Furthermore, as all RoboCode Leader robots, the Basic Leader Bot has 20% more energy than standard robots.

Figure X: The Basic Leader Bot, with team mates, and its main radar action.

The Basic Leader Bot is in no way a spectacular robot, but its low price and simplicity can make it a useful member of robot teams. The robot broadcasts the positions of enemy targets almost continuously, thus it can be successfully combined with effectively uses such information.

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>

2.2 Product Functions

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate,

such as a top level data flow diagram or object class diagram, is often effective.>

2.3 User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

2.4 Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>

2.5 Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

2.6 User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

2.7 Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are

already documented elsewhere (for example, in the vision and scope document or the project plan).>

3 External Interface Requirements

3.1 User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

3.2 Hardware Interfaces

N/A – The Basic Leader Bot is a pure software system entirely restricted to RoboRumble simulators.

3.3 Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

3.4 Communications Interfaces

Basic Leader Bot adheres to the ETSA02 RoboTalk protocol [REF] for message broadcasting. The communication is restricted to providing locations of enemy targets according to the following format:

PRESENT THE MESSAGE.

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

4 System Features

The Basic Leader Bot has four primary features: 1) Target Identification, 2) Broadcasting Information, 3) Standard Movement, and 4) Evasive Movement.

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

4.1 Feature 1 – Target identification

4.1.1 Description and Priority

The robot is exclusively designed to be a surveillance robot, thus target identification is a high priority feature. The target identification consists of a single extensive radar action.

4.1.2 Stimulus/Response Sequences

N/A – the robot executes target identification as part of its standard sequence.

4.1.3 Functional Requirements

FREQ-1-1 A radar action shall constitute a 10,000 degrees clockwise radar movement.

FREQ-1-2 The radar action shall distinguish between friendly targets and enemy targets.

FREQ-1-3 If an enemy target is located, the robot shall calculate the position of the enemy based on the robot's own bearing and the distance to the enemy target.

FREQ-1-4 When the position of the enemy target has been calculated, the position should be broadcast to all robot teammates as specified by Feature 2 – "Broadcasting Information".

4.1.4 Quality Requirements

QREQ-1-1 The robot shall detect enemy targets with an accuracy of TBD.

QREQ-1-2 Round-of errors...

4.2 Feature 2 – Broadcasting information

Whenever Feature 1 – Target identification locates an enemy target and calculates its position, the robot then provides the corresponding information to its teammates. As Basic Leader Bot is a surveillance robot, consequently Broadcasting information is a high priority feature.

4.2.1 Stimulus/Response Sequences

The robot performs this feature when the position of an enemy target has been calculated in Feature 1 – Target identification.

4.2.2 Functional Requirements

FREQ-2-1 All teammates shall receive the position of the enemy target.

FREQ-2-2 If broadcasting of a message fails, an error message shall be printed to standard output.

4.2.3 Quality Requirements

QREQ-2-1 The broadcast message shall adhere to the ETSA02 RoboTalk protocol. [ADHERENCE]

QREQ-2-2 The error message shall contain a stack trace. [MAINTAINABILITY]

4.3 Feature 3 – Standard movement

Standard movement is the default movement pattern the robot executes. It is a low priority feature, as it is independent of the high priority robot features and represents a very rudimentary behavior.

Figure X: Standard flow. Exception flow when collision moving forward/backwards.

4.3.1 Stimulus/Response Sequences

The robot conducts standard movement when one complete radar action has completed.

4.3.2 UC-3-1 – Restricted Line Movement

Step 1 The robot moves 100 units forwards.

Step 2 The robot moves 100 units backwards.

Exception cases that take precedence at any time:

Case A If the robot is hit by a bullet while executing restricted line movement, the robot initiates Feature 4 – Evasive movement.

Case B If the robot collides with another object (robot or wall) while moving forwards or backwards, the movement in that direction stops, see Figure X.

4.3.3 Functional Requirements

FREQ-3-1 The robot shall not be stationary on the battleground between radar actions.

FREQ-3-2 Unless the robot collides with another object, it shall return to its original position after completing a sequence of restricted line movement as specified in UC-3-1.

4.3.4 Quality Requirements

TBD

4.4 Feature 4 – Evasive Action

When the Basic Leader Bot is hit, more hits are likely to follow. The evasive action turns the tracks of the robot perpendicular to the bullet path, see Figure X. During the next standard movement, the robot will move along a new line. Evasive action is a medium priority feature.

4.4.1 Stimulus/Response Sequences

The robot undertakes evasive action when it has been hit by a bullet, no matter whether the incoming fire is from an enemy or a teammate.

4.4.2 Functional Requirements

FREQ-4-1 When hit by a bullet, the robot shall turn its tracks counter-clockwise perpendicular to the path of the incoming bullet.

4.4.3 Quality Requirements

TBD

5 Other Nonfunctional Requirements

This section describes quality requirements that apply to the system level rather than on an individual feature level.

5.1 Performance Requirements

The Basic Leader Bot must fulfil general performance requirements for RoboRumbles. This includes:

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

5.2 Security Requirements

RoboCode robots are executed in a tailored sandbox, and thus have very limited access to the underlying operating system and its file system.

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

5.3 Memory Footprint

Basic Leader Bot is designed to be simple and affordable robot. The jar-file of the entire robot deliverable should not exceed 15 kB.

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability,

reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

5.4 Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

6 Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

6.1 Appendix A: Glossary

Radar action – A robot action that explicitly scans the battleground for enemy targets. Standard sequence – Friendly target – Enemy target – RoboRumble –

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

6.2 Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

6.3 Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>