

Software Test Document

Version 1.0

May 07, 2012

Robot Motion Planning Challenge

Abdullah Hasan TAHER

Anıl ULUTÜRK

Submitted in partial fulfillment

Of the requirements of

“CENG350 – Software Engineering” Course

Change History

Date	Revision	Comment	Writer
07.05.2012	1.0	First Draft	-Abdullah Hasan TAHER -Anıl ULUTÜRK

Preface

This document contains the test specifications, test types and test cases for Robot Motion Planning Challenge Project. The document is prepared according to the “IEEE Standard for Software and System Test Documentation – IEEE Std 829 – 2008”.

This Test Plan provides a complete description of all the test cases of the Robot Challenge Project.

The first section of this document includes document identifier, scope, references, context and notation for description.

The second section describes the software test cases in details. It includes the main types of testing that will be performed on the application system and their inputs, expected outcomes, objectives, the preconditions and the environmental needs for each test case.

The third section includes glossary and document change procedures.

Table of Contents

Preface	3
1. Introduction.....	5
1.1 Document identifier	5
1.2 Scope	5
1.2.1 Test Groups to be Considered.....	6
1.2.2 Test Groups to be Ignored	6
1.3 References	6
1.4 Context	7
1.5 Notation for description	7
2. Details.....	9
2.1 Coding Standards Tests	9
2.2 Functional Tests	9
2.2.1 Scenario Selection	10
2.2.2 Increase Translation Speed	11
2.2.3 Decrease Translation Speed.....	12
2.2.4 Increase Rotation Speed	13
2.2.5 Decrease Rotation Speed.....	14
2.2.6 Run.....	15
2.2.7 Pause	16
2.2.8 Replay	17
2.2.9 Put Obstacle	17
2.2.10 Quit.....	18
2.3 Platform Tests	19
2.4 Acceptance Tests.....	20
3. Global	21
3.1 Glossary	21
3.2 Document change procedures and history	22

1. Introduction

1.1 Document identifier

In order to enable reader to identify this document, following information are provided.

Date of Issue:	10.03.2012
Date of Submit:	07.05.2012
Status of Document:	Draft
Organization:	METU CENG

Authors	Signature
Abdullah Hasan TAHER	
Anıl ULUTÜRK	

1.2 Scope

Scope of this test document is drawn by scope of test cases defined in section 2. As the presented project is not multi-partitioned and not fine grained, some of the test subjects are ignored. Maintainability, Security, Accessibility and Legality issues are not included within test and set aside the scope of this document. Furthermore; tests of Platform and Acceptance are defined tentatively, just like many of the holistic tests and their results must be completed and evaluated after completion of tests regarding small portions of the project. Therefore, other than irrelevant test cases which are ignored, holistic test cases and considerations are open to change in further versions and until final draft of this document. Following two subsections define test groups that are considered and test groups that are ignored, consecutively.

1.2.1 Test Groups to be Considered

Listed group of tests are applied for this project and their properties/plans are defined in section 2 of this document.

- Coding standards
- Functional requirements
- Platform
- Acceptance

1.2.2 Test Groups to be Ignored

Listed group of tests are ignored because of their irrelevancy of the project's concerns.

- Security
- Compatibility
- Content
- Navigation
- Usability
- Accessibility
- Legality
- Maintainability

1.3 References

IEEE. *IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation Specifications*. IEEE Computer Society, 2008.

1.4 Context

The tests presented in this document are tested on a pc with average features listed below.

Hardware properties of the PC:

CPU	Intel Core2Duo T9550 2.66 GHz
GPU	ATI Mobility Radeon HD (1024 mb)
Ram	4 GB
Peripheral Devices	No
Internet Connection	No

Software properties of the PC:

Operating System	Refer to Section 6.3
Frameworks	JDK 6
Development Tools	Netbeans IDE 6.9

Beyond average PC environment defined above, no other specific context definitions are required for this project. Especially software environment in this section is not minimal, but contains satisfactory preliminary elements before execution of the program and all of the test cases described in section 2 refers to this software basis. Any other specific platform and environment on which this project successfully works/compiles shall be added to this section as well as their notes are added to the document history.

1.5 Notation for description

In chapter 2, all of the test cases are listed in a table with eight fundamental fields with some or all of them are in use. Aims of these fields are explained below:

- Test case ID: Identifies a test case with an integer value.
- Test Case Name: Name of the test case related to the requirement (Referring to SRS document) or design constraint (Referring to SDD document) defined previously.
- Test Case Type: Points out the category of requirement which can be one of these two:

- Functional
- Non-functional (Coding Standards, Platform, Acceptance)
- Test Case Objective: Simply defines the purpose of the test case i.e. correctness of which constraint it was aimed to demonstrate.
- Inputs: Generally defines preliminary user interface inputs which need to be applied in order to induce occurrence of this test case.
- Outputs: Expected correct output of this test case, when “Inputs” part is satisfied.
- Inter-case Dependencies: References pre-required test or contextually related test cases for providing a better picture of relations between testing process.
- Special procedural requirements: Defines pre or post conditions in order to execute test case(s) properly.

2. Details

2.1 Coding Standards Tests

Coding standards are important ways of hindering inadequacies within code, not the ones with functional or dynamical basis; but the ones with non-functional characteristics. Since static checking is the first step of testing a code, constraints below are defined and decided to be tested so that many of the emergent problems may occur during other tests are shed light on by the help of this phase.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective
CdngTest-01	Syntax Correctness	Coding Standards	Check the code for conformity to Java SE 6 syntax standards.
CdngTest-02	Argument Pass Controls	Coding Standards	Check the methods of classes for control against the case of invalid argument passing from the callers.
CdngTest-03	Deadlock Prevention	Coding Standards	Check the interclass dependencies against occurrence of deadlocks in concurrent execution.
CdngTest-04	Proper Coding Practice	Coding Standards	Check the indentation and proper commenting in all source files in order to ease test and possible future update phases.

2.2 Functional Tests

The purposes of these tests are to test the product whether it accomplishes the requirements that are implemented. The objective is to test the functionality of the product by not interfering the internal processes of the software. In this process certain inputs are applied to software and responses to these inputs as outputs are tested. In these tests both positive and negative scenario tests are considered and run on software. In positive test scenarios the required and planed during design state inputs are applied and outputs are evaluated. In

negative scenario tests, unplanned, unvalued, unexpected inputs are feed to software and outputs are evaluated.

The functional test cases are described below.

2.2.1 Scenario Selection

Special procedural requirements: The application is already opened by the user and a Scenario Selection window popped up. (pre-condition for all Selection Scenarios)

Intercase dependencies: All other functional test cases are depends on this test case.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
Selection-01	Select True File	Functional	Selecting a true scenario file which means that the system will get the path of this scenario file correctly.	Choose a Scenario File then click “Select Scenario File” button	The system will get the path of the chosen scenario file and opens the application’s main window.
Selection-02	No File Selected	Functional	Click cancel without selection any file to start the run mode with a scenario file specified earlier.	Click Cancel button without choosing any file	The system will get the path of the “scenario2.txt” file and opens the application’s main window.
Selection-03	Select Wrong File	Functional	Give a wrong file to the application system to test error handling.	Choose a different file (not a scenario file) then click “Select Scenario File” button	An exception will be thrown and the scenario selection window will disappear

2.2.2 Increase Translation Speed

Special procedural requirements: The application is already opened by the user and the system is in run mode. (pre-condition for all Increase Translation Speed Test Cases)

Intercase dependencies: All the test cases of the Increase Translation Speed depends on Scenario selection test case. Also it has a symmetric dependency with Decrease Translation Speed test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
IncTrans Speed-01	Increase Translation Speed True Case	Functional	Increase the translation speed of the robot	Press the up arrow key (↑) on the keyboard.	The transition speed of the robot will increase 5 units/s
IncTrans Speed-02	Increase Translation Speed out of run mode (Wrong Case)	Functional	Try to increase the translation speed out of the run mode (which should not be increased successfully)	While the system is not in the run mode (ex: Pause mode) press the up arrow key (↑) on the keyboard.	The system will do nothing
IncTrans Speed-03	Increase Translation Speed more than max (Wrong Case)	Functional	Try to increase the translation speed when the robot reaches maximum translation speed (which should not be increased successfully)	While the robot reaches its maximum translation speed, press the up arrow key (↑) on the keyboard.	The system will do nothing

2.2.3 Decrease Translation Speed

Special procedural requirements: The application is already opened by the user and the system is in run mode. (pre-condition for all Decrease Translation Speed Test Cases)

Intercase dependencies: All the test cases of the Decrease Translation Speed depends on Scenario selection test case. Also it has a symmetric dependency with Increase Translation Speed test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
DecTrans Speed-01	Decrease Translation Speed True Case	Functional	Decrease the translation speed of the robot	Press the down arrow key (↓) on the keyboard.	The transition speed of the robot will decrease 5 units/s
DecTrans Speed-02	Decrease Translation Speed out of run mode (Wrong Case)	Functional	Try to decrease the translation speed out of the run mode (which should not be decreased successfully)	While the system is not in the run mode (ex: Pause mode) press the down arrow key (↓) on the keyboard.	The system will do nothing
DecTrans Speed-03	Decrease Translation Speed more than min (Wrong Case)	Functional	Try to decrease the translation speed when the robot reaches minimum translation speed (which should not be decreased successfully)	While the robot reaches its minimum translation speed, press the down arrow key (↓) on the keyboard.	The system will do nothing

2.2.4 Increase Rotation Speed

Special procedural requirements: The application is already opened by the user and the system is in run mode. (pre-condition for all Increase Rotation Speed Test Cases)

Intercase dependencies: All the test cases of the Increase Rotation Speed depends on Scenario selection test cases. Also it has a symmetric dependency with Decrease Rotation Speed test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
IncRot Speed-01	Increase Rotation Speed True Case	Functional	Increase the rotation speed of the robot	Press the left arrow key (←) on the keyboard.	The rotation speed of the robot will increase 5 deg/s
IncRot Speed-02	Increase Rotation Speed out of run mode (Wrong Case)	Functional	Try to increase the Rotation speed out of the run mode (which should not be increased successfully)	While the system is not in the run mode (ex: Pause mode) press the left arrow key (←) on the keyboard.	The system will do nothing
IncRot Speed-03	Increase Rotation Speed more than max (Wrong Case)	Functional	Try to increase the Rotation speed when the robot reaches maximum Rotation speed (which should not be increased successfully)	While the robot reaches its maximum Rotation speed, press the left arrow key (←) on the keyboard.	The system will do nothing

2.2.5 Decrease Rotation Speed

Special procedural requirements: The application is already opened by the user and the system is in run mode. (pre-condition for all Decrease Rotation Speed Test Cases)

Intercase dependencies: All the test cases of the Decrease Rotation Speed depends on Scenario selection test cases. Also it has a symmetric dependency with Increase Rotation Speed test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
IncRot Speed-01	Decrease Rotation Speed True Case	Functional	Decrease the rotation speed of the robot	Press the right arrow key (→) on the keyboard.	The rotation speed of the robot will decrease 5 deg/s
IncRot Speed-02	Decrease Rotation Speed out of run mode (Wrong Case)	Functional	Try to decrease the Rotation speed out of the run mode (which should not be increased successfully)	While the system is not in the run mode (ex: Pause mode) press the right arrow key (→) on the keyboard.	The system will do nothing
IncRot Speed-03	Decrease Rotation Speed more than min (Wrong Case)	Functional	Try to decrease the Rotation speed when the robot reaches minimum Rotation speed (which should not be decreased successfully)	While the robot reaches its minimum Rotation speed, press the right arrow key (→) on the keyboard.	The system will do nothing

2.2.6 Run

Special procedural requirements: The application is already opened by the user. (precondition for all Run Test Cases)

Intercase dependencies: All the test cases of Run depends on Scenario selection test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
Run-01	Run True Case	Functional	Change the mode of the robot from Pause to Run	While the robot is in Pause mode press the '1' key on the keyboard.	The System will transmit to the run mode
Run-02	Run while in Replay (Wrong Case)	Functional	Try to change the mode of the robot from replay to run directly (without pausing the robot)	While the system is in Replay mode, press the '1' key on the keyboard.	The system will do nothing
Run-03	Run after target reached (Wrong Case)	Functional	Try to change the mode of the robot after it reaches its target.	After the robot reached the target, press the '1' key on the keyboard.	The system will do nothing
Run-04	Run after scenario selection (True Case)	Functional	Display information related to the robot and score of the game continuously	After the selection of scenario file the application started in run mode (No	The score will start to decrease (according to timePenalty, obstaclePenalty values in delta

				input needed for this test case)	time) and displayed on the left corner of the screen. Target position, robot's position, its velocity and heading will also displayed.
--	--	--	--	----------------------------------	--

2.2.7 Pause

Special procedural requirements: The application is already opened by the user. (precondition for all Pause Test Cases)

Intercase dependencies: All the test cases of Pause depends on Scenario selection test cases. In addition it depends on the Replay test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
Pause-01	Pause True Case	Functional	Change the mode of the robot from Run to Pause mode	While the robot is in Run mode press the '2' key on the keyboard.	The System will transmit to the Pause mode which pauses the robot.
Pause-02	Pause while in Replay (Wrong Case)	Functional	Try to change the mode of the robot from Replay to Pause (5 seconds replay does not finished yet)	While the system is in Replay mode, press the '2' key on the keyboard.	The system will keep running in its current mode (does not affected)

2.2.8 Replay

Special procedural requirements: The application is already opened by the user.(precondition for all Replay Test Cases)

Intercase dependencies: All the test cases of Replay depends on Scenario selection test cases. In addition it depends on the Pause test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
Replay-01	Replay True Case	Functional	Change the mode of the robot from Pause to Replay	While the robot is in Pause mode press the '3' key on the keyboard.	The System will transmit to the Replay mode which replays the motion of the robot for the last 5 seconds
Replay-02	Replay while in Run (Wrong Case)	Functional	Try to change the mode of the robot from Run to Replay directly (without pausing the robot)	While the system is in Run mode, press the '3' key on the keyboard.	The system will do nothing

2.2.9 Put Obstacle

Special procedural requirements: The application is already opened by the user and a scenario file is selected. (pre-condition for all Put Obstacle Test Cases)

Intercase dependencies: All the test cases of Put Obstacle depends on Scenario selection test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
PutObs-01	Put Obstacle True Case	Functional	Put Obstacles on the display panel	Press the left button on mouse and then release it.	The System places an obstacle on the screen at the position of the cursor

2.2.10 Quit

Special procedural requirements: The application is already opened by the user and Scenario Selection is made. (pre-condition for Quit Test Cases)

Intercase dependencies: All the test cases of Quit depends on Scenario selection test cases.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
Quit-01	Exit Application	Functional	Exit the application by pressing a key from keyboard	Press 'Q' key on the keyboard	The Operating System will close the application
Quit-02	Fail to Exit	Functional	Try to exit within scenario selection window (pre-condition is not satisfied)	While the scenario selection window is active press 'Q' key on the keyboard	The system will do nothing

The characteristics and configurations of the hardware and software required to execute these functional test cases and the test environmental needs for test setup, execution, and results recording are described previously in “1.4 Context”.

2.3 Platform Tests

Platform tests provide a better understanding of cross platform compatibility of the project being developed. In order to provide such a view, five different operating systems with different characteristics are going to be used in platform tests; with common ground of “Special Procedural Requirements” defined in the table below.

Test Case ID	Test Case Name	Test Case Type	Test Case Objective	Inputs	Expected Outcome(s)
PltfTest-01	Windows7 32-bit	Platform	Test the compilation and execution of the project files.	Source files (Refer to SDD document)	Project execution according to functional requirements
PltfTest-02	Windows7 64-bit	Platform	Test the compilation and execution of the project files.	Source files (Refer to SDD document)	Project execution according to functional requirements
PltfTest-03	Linux Mint 32-bit (Debian)	Platform	Test the compilation and execution of the project files.	Source files (Refer to SDD document)	Project execution according to functional requirements
PltfTest-04	Fedora 64-bit (Redhat)	Platform	Test the compilation and execution of the project files.	Source files (Refer to SDD document)	Project execution according to functional requirements
PltfTest-05	MacosX 10.7 Lion 64-bit	Platform	Test the compilation and execution of the project files.	Source files (Refer to SDD document)	Project execution according to functional requirements

Special procedural requirements	Intercase dependencies	Other
JDK 6 and NetBeans IDE with version higher than 6.8 is installed on operating systems.	None	None

2.4 Acceptance Tests

Acceptance test is considered as application of the previously defined tests in this document with a different team/tester in order to provide a last step before demo phase of the project. For the project documented to be tested in this document, the same approach will be used. Coding Standards tests (Refer to section 2.1), Functional tests (Refer to section 2.2), Platform tests (Refer to section 2.3) applied in the process of design and development will be applied the last time and if these aforementioned tests pass, Acceptance Tests will be considered as passed as well.

3. Global

3.1 Glossary

Term	Definition
STD	Software Test Document
Deadlock	A situation where two or more competing actions are each waiting for the other to finish, and thus neither ever does.
Inter-case Dependencies	Identifiers of test cases which must be executed prior to other test cases.
METU CENG	Middle East Technical University – Computer Engineering
Acceptance Testing	A test conducted to determine if the requirements of a specification or contract are met
Platform testing	Experience of testing applications that work in various different environments, operating system, and hardware set-ups.
Redhat	A popular Linux based operating system
Debian	Unique software distribution made up of a large number of software packages
Security	A judgment of how likely it is that the system can resist accidental or deliberate intrusions.
Compatibility	A characteristic of software components or systems which can operate satisfactorily together
Accessibility	Describe the degree to which a product, device, service, or environment is available to as many people as possible
Maintainability	Reflects the extent to which the system can be adapted to new requirements
Reliability	The probability that the system will correctly deliver services as expected by users.

3.2 Document change procedures and history

As testing phase succeeds design and development phases and in this project, incremental testing is adopted; most of the holistic tests such as component testing and integration testing are defined in a non-determinate way. As the project and their unit tests proceed, holistic tests will be edited or updated so that a better test plan which lets reader to perceive all of the project's elements in a complete way is to be achieved in the latest version of final draft. All of these updates and changes regarding all-covering test categories will be presented and noted in Change History section.