

Autonomous Pool Playing Robot

Requirements Specification

Eric Le Fort
leforte@mcmaster.ca
1308609

February 28, 2017

Contents

1	Introduction	3
1.1	Overview	3
1.2	Purpose	3
1.3	Naming Conventions & Definitions	3
1.3.1	Definitions	4
1.3.2	Acronyms & Abbreviations	4
2	Mechanical Components	5
3	Electrical System	5
4	Software System	5
4.1	Unit Tests	5
4.1.1	PC Controller Program	5
4.1.2	PC VR Program	14
4.1.3	μ C Program	14
4.1.4	Android Program	14
4.2	System Tests	14
5	Summary of Results	14

List of Tables

1	Revision History	2
2	Definitions	4
3	Acronyms and Abbreviations	4
4	Ball Constructor Good Inputs	5
5	Ball Constructor Large X	6
6	Ball Constructor Large Y	6
7	Ball Constructor Small X	6
8	Ball Constructor Small Y	7
9	Ball Constructor Small Value	7
10	Ball Constructor Large Value	7
11	Updating Table State	8
12	Selecting an Optimal Shot	8
13	Shot Constructor Good Inputs	8
14	Shot Constructor Large X	9
15	Shot Constructor Small X	9
16	Shot Constructor Large Y	10
17	Shot Constructor Small Y	10
18	Shot Constructor Large Angle	10
19	Shot Constructor Small Angle	11
20	Shot Constructor Large Power	11
21	Shot Constructor Small Power	11
22	TableState Constructor Good Inputs	12
23	TableState Constructor Too Many Elements	12
24	TableState Constructor Not Enough Elements	12

25	TableState Constructor Elements Too Small	13
26	TableState Constructor Elements Too Large	13
27	TableState Deep Copy	14

Date	Revision #	Comments	Authors
27/02/2017	0	- Initial document creation	Eric Le Fort

Table 1: Revision History

1 Introduction

This document will provide a specification of a test plan for an automated pool playing robot and report on the results of that plan.

1.1 Overview

This document breaks down the required testing for each domain of the system. It begins with the hardware aspect, then moves to the electrical side and then finishes with software. Each section will provide a traceability matrix to map the requirements to tests that check their completion and then go into further detail to describe each test case. Lastly, a summary of the results of testing will be provided to conclude the document.

1.2 Purpose

The aim of this document is to illuminate any design flaws, software bugs, or other issues in the system. Once these issues are discovered, the engineering team will be able to work on eliminating them or minimizing their frequency and consequences.

1.3 Naming Conventions & Definitions

This section outlines the various definitions, acronyms and abbreviations that will be used throughout this document in order to familiarize the reader prior to reading.

1.3.1 Definitions

Table 2 lists the definitions used in this document. The definitions given below are specific to this document and may not be identical to definitions of these terms in common use. The purpose of this section is to assist the user in understanding the requirements for the system.

Table 2: Definitions

Term	Meaning
X-axis	Distance along the length of the pool table
Y-axis	Distance across the width of the pool table
Z-axis	Height above the pool table
End-effector	The end of the arm that will strike the cue ball
θ	Rotational angle of end-effector
Cue	End-effector
Personal Computer	A laptop that will be used to run the more involved computational tasks such as visual recognition and the shot selection algorithm
Camera	Some form of image capture device (e.g. a digital camera, smartphone with a camera, etc.)
Table State	The current positions of all the balls on the table
Entity	Classes that have a state, behaviour and identity (e.g. Book, Car, Person, etc.)
Boundary	Classes that interact with users or external systems
Double	Double-precision floating point numbers

1.3.2 Acronyms & Abbreviations

Table 3 lists the acronyms and abbreviations used in this document.

Table 3: Acronyms and Abbreviations

Acronym/Abbreviation	Meaning
VR	Visual Recognition
PC	Personal Computer
μC	Micro-Controller
CRC	Class Responsibility Collaboration
TBT	To Be Tested

Test ID: n	Module: Ball	Status: TBT
Ball Constructor Good Inputs		
Pass/Fail Conditions: This test is passed if all the fields inside of Ball are correctly initialized.		
Pre-Conditions: None		
Input: 1, 0.7, 0		
Expected Results: A new ball with x-coordinate 1, y-coordinate 0.7, and the value 0.		Actual Results:
Post-Conditions: A new Ball object should be available.		

Table 4: Ball Constructor Good Inputs

2 Mechanical Components

3 Electrical System

4 Software System

The software system is comprised of four main components: a control system running on an Arduino microcontroller, an automated image capture application running on an Android smartphone, as well a visual recognition program and smart shot selection program running on a PC. On top of the typical suite of unit tests to verify correctness of methods, rigorous system testing will also be crucial to adequately test this system.

The following traceability matrix will demonstrate that the tests to be performed prove that specified requirements have been met.

4.1 Unit Tests

This section will provide a plethora of test cases which should prove correctness of the program. Each individual class will be tested in order to make finding specific test cases easier.

4.1.1 PC Controller Program

Ball Tests

InferenceEngine Tests

Test ID: n	Module: Ball	Status: TBT
Ball Constructor Large X		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None		
Input: 1.87658, 0.7, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 5: Ball Constructor Large X

Test ID: n	Module: Ball	Status: TBT
Ball Constructor Large Y		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None		
Input: 1, 0.94958, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 6: Ball Constructor Large Y

Test ID: n	Module: Ball	Status: TBT
Ball Constructor Small X		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None		
Input: -1.001, 0.7, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 7: Ball Constructor Small X

Test ID: n	Module: Ball	Status: TBT
Ball Constructor Small Y		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None		
Input: 1, -1.001, 0		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 8: Ball Constructor Small Y

Test ID: n	Module: Ball	Status: TBT
Ball Constructor Small Value		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown		
Pre-Conditions: None		
Input: 1, 0.7, -1		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 9: Ball Constructor Small Value

Test ID: n	Module: Ball	Status: TBT
Ball Constructor Large Value		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1, 0.7, 16		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: There should not have been a Ball created.		

Table 10: Ball Constructor Large Value

Test ID: n	Module: InferenceEngine	Status: TBT
Updating Table State		
Pass/Fail Conditions: This test is passed if all post-conditions are met.		
Pre-Conditions: None		
Input: A 16-by-2 array of doubles that are valid positions, BallType.STRIPEs		
Expected Results: None	Actual Results: None	
Post-Conditions: <div><div>1. Stored BallType is BallType.STRIPEs.</div><div>2. The stored positions array is the same as the one passed in.</div><div>3. The stored best shot is null.</div><div>4. The stored table state reflects the positions passed in.</div></div>		

Table 11: Updating Table State

Test ID: n	Module: InferenceEngine	Status: TBT
Selecting an Optimal Shot		
Pass/Fail Conditions: This test is passed if a reasonable Shot is returned.		
Pre-Conditions: The current table state is not null and the current ball type is not null or BallType.CUE.		
Input: None		
Expected Results: A reasonable Shot (no bank shots, shooting the right ball, valid x-/y-coordinates).	Actual Results:	
Post-Conditions: The best shot for the current table state is stored.		

Table 12: Selecting an Optimal Shot

Test ID: n	Module: PCCommunicator	Status: TBT
Read Valid Table State from File		
Pass/Fail Conditions: This test is passed if the output matches the data in the text file.		
Pre-Conditions: None.		
Input: A text file with 16 ball positions		
Expected Results: The 16 ball positions stored in the text file.	Actual Results:	
Post-Conditions: None.		

Table 13: Read Valid Table State from File

Test ID: n	Module: PCCommunicator	Status: TBT
Read Table State from Non-Existent File		
Pass/Fail Conditions: This test is passed if a FileNotFoundException is thrown.		
Pre-Conditions: None.		
Input: None.		
Expected Results: A FileNotFoundException is thrown.	Actual Results:	
Post-Conditions: None.		

Table 14: Read Table State from Non-Existent File

PCCommunicator Tests

Shot Tests

Test ID: n	Module: PCCommunicator	Status: TBT
Read Table State from File with Invalid Data		
Pass/Fail Conditions: This test is passed if an InputMismatchException is thrown.		
Pre-Conditions: None.		
Input: A file containing the text “Bad data”.		
Expected Results: An InputMismatchException is thrown.	Actual Results:	
Post-Conditions: None.		

Table 15: Read Table State from File with Invalid Data

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Good Inputs		
Pass/Fail Conditions: This test is passed if the Shot is successfully created and stores the correct information.		
Pre-Conditions: None		
Input: 1, 0.5, 3.5, 1		
Expected Results: A new Shot with an x-coordinate of 1, a y-coordinate of 0.5, an angle of 3.5, and a power of 1.	Actual Results:	
Post-Conditions: Shot has been created.		

Table 16: Shot Constructor Good Inputs

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Large X		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1.87658, 0.5, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 17: Shot Constructor Large X

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Small X		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: -0.001, 0.5, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 18: Shot Constructor Small X

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Large Y		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1, 0.94958, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 19: Shot Constructor Large Y

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Small Y		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1, -0.001, 3.5, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 20: Shot Constructor Small Y

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Large Angle		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1, 0.5, 6.284, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 21: Shot Constructor Large Angle

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Small Y		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1, 0.5, -0.01, 1		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 22: Shot Constructor Small Angle

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Large Power		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1, 0.5, 3.5, 1.001		
Expected Results: An IllegalArgumentException is thrown.	Actual Results:	
Post-Conditions: Shot has not been created.		

Table 23: Shot Constructor Large Power

Test ID: n	Module: Shot	Status: TBT
Shot Constructor Small Power		
Pass/Fail Conditions: This test is passed if an IllegalArgumentException is thrown.		
Pre-Conditions: None		
Input: 1, 0.5, 3.5, 0		
Expected Results: An IllegalArgumentException is thrown.		Actual Results:
Post-Conditions: Shot has not been created.		

Table 24: Shot Constructor Small Power

Test ID: n	Module: TableState	Status: TBT
TableState Constructor Good Inputs		
Pass/Fail Conditions: This test is passed if the TableState is successfully created and stores the correct information.		
Pre-Conditions: None		
Input: A 16-by-2 array of doubles that hold the position of the balls		
Expected Results: A new TableState with 16 balls in positions corresponding to those passed in.		Actual Results:
Post-Conditions: TableState has been created.		

Table 25: TableState Constructor Good Inputs

SimulationInstance Tests

TableState Tests

Test ID: n	Module: TableState	Status: TBT
TableState Constructor Too Many Elements		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None		
Input: A 17-by-2 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 26: TableState Constructor Too Many Elements

Test ID: n	Module: TableState	Status: TBT
TableState Constructor Not Enough Elements		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None		
Input: A 15-by-2 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 27: TableState Constructor Not Enough Elements

Test ID: n	Module: TableState	Status: TBT
TableState Constructor Elements Too Small		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None		
Input: A 16-by-1 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 28: TableState Constructor Elements Too Small

Test ID: n	Module: TableState	Status: TBT
TableState Constructor Elements Too Large		
Pass/Fail Conditions: This test is passed if the TableState is not created.		
Pre-Conditions: None		
Input: A 16-by-3 array of doubles		
Expected Results: An IllegalArgumentException has been thrown.	Actual Results:	
Post-Conditions: TableState has not been created.		

Table 29: TableState Constructor Elements Too Large

Test ID: n	Module: TableState	Status: TBT
TableState Deep Copy		
Pass/Fail Conditions: This test is passed if the array of Balls returned have the same values but are not the same Objects.		
Pre-Conditions: A TableState exists in memory.		
Input: None.		
Expected Results: An array of Balls that have the same positions as those in the TableState.	Actual Results:	
Post-Conditions: None.		

Table 30: TableState Deep Copy

4.1.2 PC VR Program

4.1.3 μ C Program

4.1.4 Android Program

4.2 System Tests

5 Summary of Results