Autonomous Pool Playing Robot

Requirements Specification

Eric Le Fort
leforte@mcmaster.ca
1308609

Guy Meyer
meyerg@mcmaster.ca
1320231

March 1, 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 **4**

3 Electrical System **6**

4 Software System **6**
4.1 Unit Tests . . . 9
4.1.1 PC Controller Program . . . 9
4.1.2 PC VR Program . . . 19
4.1.3 μC Program . . . 19
4.1.4 Android Program . . . 19
4.2 System Tests . . . 19

5 Summary of Results **19**

List of Tables

1 Revision History . . . 2
2 Definitions . . . 4
3 Acronyms and Abbreviations . . . 4
4 Sychronious Motion in X Rail . . . 5
5 Motion in Y Rail . . . 5
6 End-Effector Orientation . . . 5
7 Shooting Command . . . 6
8 Shooting Mechanism Orientation . . . 6
9 User Input to Arduino . . . 7
10 Emergency Stop . . . 7
11 Move Away Command . . . 7
12 Perimeter Coverage . . . 8
13 Ball Avoidance . . . 8
14 Random Shot Request . . . 8
15 Ball Constructor Good Inputs . . . 9
16 Ball Constructor Large X . . . 9
17 Ball Constructor Large Y . . . 10
18 Ball Constructor Small X . . . 10
19 Ball Constructor Small Y . . . 10
20 Ball Constructor Small Value . . . 11
21 Ball Constructor Large Value . . . 11
22 Updating Table State . . . 12
23 Selecting an Optimal Shot . . . 12
24 Read Valid Table State from File . . . 12

25 Read Table State from Non-Existent File . . . 13
26 Read Table State from File with Invalid Data . . . 13
27 Shot Constructor Good Inputs . . . 13
28 Shot Constructor Large X . . . 14
29 Shot Constructor Small X . . . 14
30 Shot Constructor Large Y . . . 14
31 Shot Constructor Small Y . . . 15
32 Shot Constructor Large Angle . . . 15
33 Shot Constructor Small Angle . . . 16
34 Shot Constructor Large Power . . . 16
35 Shot Constructor Small Power . . . 16
36 TableState Constructor Good Inputs . . . 17
37 TableState Constructor Too Many Elements . . . 17
38 TableState Constructor Not Enough Elements . . . 17
39 TableState Constructor Elements Too Small . . . 18
40 TableState Constructor Elements Too Large . . . 18
41 TableState Deep Copy . . . 18

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

2 Mechanical Components

Test ID: 12.1	**Sychronious Motion in X Rail**	Status: TBT
Description: Verify that X-Rails can synchronously move to the same location at the same speed without getting stuck while loaded		
Pass/Fail Condition: If rail moves adequately and quickly as expected		
Pre-Conditions: None Input: Location along x-direction (i.e. 2000 steps)		
Expected Results: Smooth and consistent motion along axis until position is met. Followed by an immediate stop	Actual Results:	
Post-Conditions: Rails are stationary with no slip.		

Table 4: Sychronious Motion in X Rail

Test ID: 12.2	**Motion in Y Rail**	Status: TBT
Description:Verify that Y-Rail can move to a location without getting stuck while loaded		
Pass/Fail Condition: If rail moves adequately and quickly as expected		
Pre-Conditions: None Input: Location along y-direction		
Expected Results: Smooth and consistent motion along axis until position is met. Followed by an immediate stop	Actual Results:	
Post-Conditions: Rail is stationary with no slip.		

Table 5: Motion in Y Rail

Test ID: 12.3	**End-Effector Orientation**	Status: TBT
Description: Verify that EE-Base Motor can orient to a specific angle without getting stuck while loaded		
Pass/Fail Condition: If motor turns adequately and quickly as expected to correct angle		
Pre-Conditions: None Input: Angle of orientation with respect to the x-axis		
Expected Results: Smooth and consistent motion until orientation is met. Followed by an immediate stop	Actual Results:	
Post-Conditions: Motor is stationary.		

Table 6: End-Effector Orientation

Test ID: 13.1	**Shooting Command**	Status: TBT
Description: Arduino received command to take shot		
Pass/Fail Condition: Arduino initiates shot taking process		
Pre-Conditions: System powered on and ready to change states Input: Command to Arduino to take shot		
Expected Results: User commands system to shoot. Arduino initiates shot taking process.	Actual Results:	
Post-Conditions: Shot is taken and balls are settled		

Table 7: Shooting Command

Test ID: 13.2	**Shooting Mechanism Orientation**	Status: TBT
Description: EE is positioned correctly and waiting command to power piston		
Pass/Fail Condition: Piston is settled at correct oreintation, awaiting command to actuate piston		
Pre-Conditions: Motors orient piston to proper orentation Input: Position and orientation components sent to Arduino		
Expected Results: System moves to desired location and waits for piston signal	Actual Results:	
Post-Conditions: Piston can be safely actauted and strike cue ball		

Table 8: Shooting Mechanism Orientation

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

Test ID: 14.1	**User Input to Arduino**	Status: TBT
Description: User applies input, then the Arduino indicated a received message		
Pass/Fail Condition:Arduino output to console correct desired status		
Pre-Conditions: None Input: User pressed status button		
Expected Results: Console output	Actual Results:	
Post-Conditions: None		

Table 9: User Input to Arduino

Test ID: 14.2	**Emergency Stop**	Status: TBT
Description: Stop command during runtime		
Pass/Fail Condition: Upon user request the system stops immediately		
Pre-Conditions: System is preforming an action Input: Stop command is sent		
Expected Results: System halts immediately along with visual indication	Actual Results:	
Post-Conditions: System is motion unless and awaiting next command		

Table 10: Emergency Stop

Test ID: 15.1	**Move Away Command**	Status: TBT
Description: Physical system is interfering with players ability to approach the table. The user commands the system to move over		
Pass/Fail Condition: Upon user command the system moves to predetermined location		
Pre-Conditions: None Input: Command to Arduino to move in desired location on either side of table		
Expected Results: System indicates acceptance of command and preforms action. Upon completion the system stops.	Actual Results:	
Post-Conditions: System is motionless and awaits next command.		

Table 11: Move Away Command

Test ID: 16.1	**Perimeter Coverage**	Status: TBT
Description: EE will be moved around the table to ensure that it is able to reach all locations and orientations		
Pass/Fail Condition: EE is capable of completing a full trip around the perimeter without stops		
Pre-Conditions: None Input: Motion command from Arduino		
Expected Results: EE will travel around perimeter of table. Inspection that its location is sufficient for shot-taking is required.	Actual Results:	
Post-Conditions: System awaits next command.		

Table 12: Perimeter Coverage

Test ID: 16.2	**Ball Avoidance**	Status: TBT
Description: As the EE is moving around the table it much avoid the balls to not interfere with gameplay		
Pass/Fail Condition: Able to move randomly around table without moving rolling or stationary balls		
Pre-Conditions: Ball in motion OR stationary Input: Random motion along table		
Expected Results: EE travels directly over balls and does not make contact	Actual Results:	
Post-Conditions: None		

Table 13: Ball Avoidance

Test ID: 17.1	**Random Shot Request**	Status: TBT
Description: User can initiate shot taking process at any time		
Pass/Fail Condition: During arbitrary state the user requests to attempt shot and the system initiates shot taking process		
Pre-Conditions: Arbitrary state Input: Take a Shot command		
Expected Results: System switches state and initiates shot taking sequence	Actual Results:	
Post-Conditions: None		

Table 14: Random Shot Request

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 15: Ball Constructor Good Inputs

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 16: Ball Constructor Large X

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

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 17: 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 18: 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 19: 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 20: 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 21: Ball Constructor Large Value

InferenceEngine Tests

PCCommunicator Tests

Shot Tests

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: 1. Stored BallType is BallType.STRIPES. 2. The stored positions array is the same as the one passed in. 3. The stored best shot is null. 4. The stored table state reflects the positions passed in.		

Table 22: 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 23: 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 24: 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 25: Read Table State from Non-Existent File

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 26: 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 27: 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 28: 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 29: 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 30: 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 31: 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 32: Shot Constructor Large Angle

SimulationInstance Tests
TableState Tests

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 33: 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 34: 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 35: 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 36: TableState Constructor Good Inputs

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 37: 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 38: 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 39: 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 40: 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 41: TableState Deep Copy

4.1.2 PC VR Program

4.1.3 μC Program

4.1.4 Android Program

4.2 System Tests

5 Summary of Results