CengBall
System Test Document

25.05.2014

Team Project Contorium

Alper Demir
Doğa Uzuncukoğlu
Emre Can Küçükoğlu
Necati Çevik

Change History

Date	Revision	Comment
25.05.2014	1.0	Document created

TABLES

1. Introduction

This System Test Document (STD) ‘s aim is to describe the entire test effort for ‘CengBall’ project. The goal is to provide a framework that can be used by testers to plan and execute the necessary tests in a timely and cost-effective manner.
The project will have one level of testing which is System Testing. At the end of the test phases, it will be determined if the system satisfies the requirements that had been described at the beginning of the project development process.
IEEE Std 829-2008 will be the reference for this System Test Document.

1.1 Scope

This document’s purpose is to describe the entire test effort for CengBall. There will be one level of testing, so test documentation is limited to System Testing. Although the software-based systems consist of software, hardware and users, the test phase for CengBall will be focus on the software and users mainly.
In the Software Requirements Specification document, the functional requirements and the use cases of CengBall were specified. The test cases described in this document will be according to the those functional requirements and use cases. The results of operations will be controlled if they are reliable and true or not.
For each test cases, inspection, analysis, demonstration, verification and validation will be considered while testing phase.
IEEE Std 829-2008 will be taken as a basis and “Level Test Design Outline” will be the outline for this document.

1.2 Definitions, Terms and Abbreviations

Term - Acronym - Abbreviation	Definition
CengBall	The name of the project.
TC	Test Case
UC	Use Case

1.3 References

- CengBall, SRS
- CengBall, SDD
- IEEE Std 829-2008
- Home Finder, STD (as a guide for formatting)

2. Details for System Test

2.1 Test Items and Their Identifiers

The only test item is the CengBall system itself. Each functionality of the system which will be tested is defined in detail in section 2.4.1. There are no installation instructions since the project is a single runnable. While the system is operating system independent an installation of Java Development Kit required.

- Test - CengBall, STD (this document)
- Requirements - CengBall, SRS
- Design - CengBall, SDD
- Plan - CengBall, SDD (planning section)

2.2 Approach

2.2.1 Test Levels

Black Box testing methods will be used in System Test Level. Features and use cases which are defined in SRS Document will be tested by using appropriate test scenarios and outputs of the system will be used to analyze test results.
Performance of the system will be tested using automated test scripts which will be generated to stress the system.Under defined workload in SRS document, behavior of the system will be observed.
Reliability of the system will be tested with multiple test data with consistent and inconsistent values and outputs of the system will be used to analyze.
User Acceptance Test will be performed after all system functional and non functional test scenarios are performed. System requirements will be performed on the system to analyze if the system meets user's expectation and needs or not.

2.2.2 Analyzing Test Results

Test results of functional system properties will be evaluated using pass/ fail criteria of test cases. Test steps of test cases will be executed manually by members of the development team. Actual results of the test steps will be compared with expected results and invalid results will be evaluated as failed test scenarios.
Performance of the system will be tested with automated performance test tools which are developed to stress the system. The results will be observed and later it will be decided if the system performance needs are met or not.
Reliability of the system will be tested by using various test data which produce same results.

2.3 Item pass/fail Criteria

Since this document will focus only on system testing, only test item is the whole CengBall system. If all tests are successful the item will pass otherwise it will be considered a failed test item.

2.4 Test Deliverables

2.4.1 System Test Cases

2.4.1.1 Test Case : Import Agent

Table 1 - Test Case : Import Agent

Test Case ID	CengBall.TC.1
Test Case Name	IMPORT AGENT
Related Use Case	CengBall.UC.1 - Import Agent
Objective	This case verifies if users can import a written java code as an agent into system.
Inputs	A java file(.java)
Preconditions	There should be a java file to be imported (check special requirements)
Postconditions	-
Expected Test Result	The submitted java code should be compiled and 1. If there are compile errors, the system should say it so and cancel the importation.

	2. If the compilation completed without errors, said file should be imported as an ‘Agent’ which can later be selected as a team.
Pass/Fail Criteria	The imported agent should be observable in Kick Off menu as a selectable team.
Intercase dependencies	
Special Requirements	The java file should be coded in a pre-determined specific way. Users are provided with a template file and are expected to fill it accordingly.

Table 2 - Test Case : Import Agent - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Import Team‘	Import Team screen appears.
3	Click ‘Browse’	Browse screen appears.
4	Locate the file to be imported and click ‘Open’ button.	“File is imported” appears.
5	Click ‘Check’	Code should be compiled. If there are compilation errors, they will be shown otherwise “Successfully imported” message appears.
6	Click ‘Finish’	Return to main menu.
7	Click ‘Kick Off’ and observe the imported agent.	Kick off screen is displayed,imported agent should be selectable as a team here.

2.4.1.2 Test Case : Simulate a Match

Table 3 - Test Case : Simulate a Match

Test Case ID	CengBall.TC.2
Test Case Name	SIMULATE MATCH
Related Use Case	CengBall.UC.2 - Simulation
Objective	This case verifies if the system can simulate a game between two agents.
Inputs	Two agents selected by the user.
Preconditions	There should be at least one imported agent.
Postconditions	-
Expected Test Result	A match should be simulated and a save file should be created.
Pass/Fail Criteria	Simulation should finish and later should be able to be visualized.
Intercase dependencies	CengBall.TC.1 - Import Agent should work so there are agents to simulate a game.
Special Requirements	-

Table 4 - Test Case : Simulate a Match - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Kick Off‘	Kick Off screen appears
3	Select agents for Home and Away teams	
4	Click ‘Start Match’	Simulation screen appears.
5	Click ‘Play’	Simulation begins and goes on for a while.
6	After the simulation is completed Click ‘Watch’	Visualization of the simulation appears.
7	Click ‘Play’	Visualization continues till the end of the simulation.

2.4.1.3 Test Case : Watch a Match

Table 5 - Test Case : Watch a Match

Test Case ID	CengBall.TC.3
Test Case Name	WATCH MATCH
Related Use Case	CengBall.UC.3 - Watch a match
Objective	This case verifies if a previously simulated match can be re-visualized.
Inputs	A save file
Preconditions	There should be a save file meaning that the system must have been simulated a match already.
Postconditions	-
Expected Test Result	The previously simulated match will be visualized.
Pass/Fail Criteria	The visualization should be observed.
Intercase dependencies	CengBall.TC.2 - Simulate a Match
Special Requirements	-

Table 6 - Test Case : Watch a Match - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Watch’‘	Watch screen appears where the previously simulated matches are displayed.
3	Select one of available matches.	
4	Click ‘Watch’	Visualization starts.

2.4.1.4 Test Case : Change Match Length

Table 7 - Test Case : Change Match Length

Test Case ID	CengBall.TC.4
Test Case Name	CHANGE MATCH LENGTH
Related Use Case	CengBall.UC.4 - Change match length
Objective	This case verifies if length of the simulation changes accordingly to the settings.
Inputs	User selects the match length. (an integer)
Preconditions	-
Postconditions	-
Expected Test Result	The simulation will last as long as selected
Pass/Fail Criteria	The visualization should be observed and it should last as selected otherwise test fails.
Intercase dependencies	CengBall.TC.2 - Simulate a Match
Special Requirements	-

Table 8 - Test Case : Change Match Length - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Kick Off’	Kick Off menu appears
3	Select teams for both sides and change the match length	
4	Click ‘Start Match’	Simulation screen appears
5	Click ‘Play’ and wait till the simulation finishes and later click ‘Watch’	Visualization begins
6	Watch the match until it finishes and notice the time when the simulation is finished.	Time should be equal to the one selected.

2.4.1.5 Test Case : Visualization controls

Table 9 - Test Case : Visualization controls

Test Case ID	CengBall.TC.5
Test Case Name	VISUALIZATION CONTROLS
Related Use Case	CengBall.UC.5 - Visualization
Objective	This case verifies if the visualizer actions work properly
Inputs	User selects actions for visualizer to perform
Preconditions	There should be a match to be visualized.
Postconditions	-
Expected Test Result	The visualizer will perform tasks pause,resume,skip,next and prev as intended.
Pass/Fail Criteria	Each task should act as intended otherwise test fails.
Intercase dependencies	CengBall.TC.3 - Watch a match
Special Requirements	-

Table 10 - Test Case : Visualization controls - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Watch’	Watch menu appears
3	Select one of the available matches to visualize and click ‘Watch’	Visualization begins.
4	Click ‘Play’	Objects will start to move.
5	Click ‘Stop’	Visualization pauses.
6	Click ‘Next’	Visualization advances one frame.
7	Click ‘Prev’	Visualization regresses one frame.
8	Click ‘Resume’	Visualization continues from where it left.

9	Click ‘Skip’	Visualization finishes, the last frame appears.
10	Click ‘End Match’	Returns to main menu.

2.4.1.6 Test Case : Terminate Simulation

Table 11 - Test Case : Terminate Simulation

Test Case ID	CengBall.TC.6
Test Case Name	TERMINATE SIMULATION
Related Use Case	CengBall.UC.6 - Terminate Simulation
Objective	This case verifies if simulation terminates correctly.
Inputs	User can click terminate any time during the simulation process.
Preconditions	A match should be during simulation
Postconditions	-
Expected Test Result	The simulation will terminate correctly.
Pass/Fail Criteria	The simulation should stop and there should be no save file or visualization.
Intercase dependencies	CengBall.TC.2 - Simulate a Match
Special Requirements	-

Table 12 - Test Case : Terminate Simulation - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Kick Off’	Kick Off menu appears
3	Select teams for both sides and click ‘Start Match’	Simulation screen appears
4	Click ‘Play’	Simulation starts.
5	Click ‘Terminate’	Simulation stops.

2.4.1.7 Test Case : Change number of players

Table 13 - Test Case : Change number of players

Test Case ID	CengBall.TC.7
Test Case Name	CHANGE PLAYER COUNT
Related Use Case	CengBall.UC.7 - Change number of players
Objective	This case verifies if number of players changes according to selection.
Inputs	User changes the number of players in the kick off menu.
Preconditions	-
Postconditions	-
Expected Test Result	Number of players will be as selected.
Pass/Fail Criteria	If number of players will be as selected test pass otherwise fails.
Intercase dependencies	CengBall.TC.2 - Simulate a Match
Special Requirements	-

Table 14 - Test Case : Change number of players - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Kick Off’	Kick Off menu appears
3	Select number players as 5,6 or 7 and click ‘Start Match’	Simulation screen appears
4	Click ‘Play’	Simulation starts.
5	Observe the number of players.	Number of players will be as selected.

2.4.1.8 Test Case : Debug mode

Table 15 - Test Case : Debug mode

Test Case ID	CengBall.TC.8
Test Case Name	DEBUG MODE
Related Use Case	CengBall.UC.8 - Debug mode
Objective	This case verifies if a log file is created if the debug mode is on.
Inputs	User enables the debug mode.
Preconditions	-
Postconditions	-
Expected Test Result	A log file will be created.
Pass/Fail Criteria	A log file will be created if debug mode is on otherwise it fails
Intercase dependencies	CengBall.TC.2 - Simulate a Match
Special Requirements	-

Table 16 - Test Case : Debug mode - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Settings’	Settings screen appears
3	Enable debug mode by clicking ‘Debug mode’ checkbox	
4	Return to main menu and simulate a match.	Log file will be created.

2.4.1.9 Test Case : Set JDK path

Table 17 - Test Case : Set JDK path

Test Case ID	CengBall.TC.9
Test Case Name	SET JDK PATH
Related Use Case	CengBall.UC.9 - Set JDK path
Objective	This case verifies if setting jdk path in settings works properly
Inputs	User browses jdk path.
Preconditions	-
Postconditions	-
Expected Test Result	Agent will be imported without errors using the given JDK path.
Pass/Fail Criteria	If agent is imported properly test passes.
Intercase dependencies	CengBall.TC.1 - Import Agent
Special Requirements	-

Table 18 - Test Case : Set JDK path - Test Steps

No	**Step Description**	**Expected Results**
1	Open the executable of the system	The system starts, main menu is displayed.
2	Click ‘Settings’	Settings screen appears
3	Set the JDK path by clicking browse next to the ‘Java development Kit Path’.	
4	Return to main menu and import an agent using ‘Import Agent’ menu .	Agent will be imported.

3. Test Management

3.1 Planned Activities and Tasks, Test Progression

Each test will be implemented in a fail-safe manner. To keep the testing consistent some of the test cases will be implemented before the others. Since there can't be a simulation without an agent, test case which tests importation of the agents will be implemented before the test case which tests the simulation.
Testing is collaborative effort amongst all of the team members. Testing environments are described in section 3.2

3.2 Environment / Infrastructure

The project is implemented in Java™ therefore it is operating system independent.

Hardware needs: A computer (an inek machine at the CEng department)
Software needs: An installation of Java Development Kit for compiling agents.

3.3 Responsibilities and Authority

System testing is a team effort and each team member is equally responsible for every action in the process.