Quality Assurance and Test Plan

Thomas Bui, Justin Carter, Patrick Flaherty, Wesley Miller Philip Seitz

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to outline the procedure for testing the Train Control Signaling System software, including lists of all unit tests to be run with the conditions for passing said tests.

1.2 Scope

The purpose of this project is to design and implement a new Centralized Traffic Control Center and Signaling System for Light Rail Transit system. Due to the safety-centric nature of this system, the testing plan will attempt to provide as near to total coverage as possible.

1.3 Definitions, Acronyms, and Abbreviations

The following are a list of terms, each followed by synonyms and abbreviations. A definition of each of the terms then follows:

- Train Control Signaling System - TCSS
  - The Train Signaling System is the complete deliverable which this SRS shall specify.
- Centralized Traffic Control - CTC
  - The Central Office is one of the major modules of the TCSS. The CTC is able to track the locations of the trains in the railway, as well as communicate to these trains important and vital information.
- Operation Control Center - OCC
  - See *Centralized Traffic Control*
- Central Office
  - See *Centralized Traffic Control*
- Track Controller
  - This is a major module of the TCSS. The Track Controller is responsible for sending important information to both the CTC and the Train Controller (Through the Track and Train Models). It also is responsible for controlling the state of the tracks, such as track switching and railway crossing. This runs a PLC and is configurable on a per-Track Controller basis.
- Train Controller

  - This is a vital *(See Vital)* component of the TCSS. It is responsible for regulating the speed of the train. It is also responsible for controlling various other features of the Train Model. It is directly interface-able by the train driver, and will be capable of being controlled in both automatic and manual modes.

- Track Model
  - This is a major component of the TCSS. It is a digital representation of the track with which the Train Model rides over, and the model which the Track Controller is responsible for. This is configurable via a formatted csv file that will allow the user to simulate and control over any track diagram.
- Train Model
  - This is a major component of the TCSS. It is a digital representation of the Train with which the Train Controller is responsible for. Furthermore it is a representation of the effect of Newtonian physics on the TCSS decision making.
- Port Authority of Allegheny County - PAAC
  - A public transit agency within Pennsylvania.
- Programmable Logic Controller - PLC
  - This is a logic controller that is specially programmable on a per-device basis. It is often adapted for the control of systems and devices that require a high reliability and ease of programming.
- Vital
  - Safety-Critical
- Graphic User Interface - GUI
  - An electronic user interface displayed through a screen so that a user may interact with software.

1.4 References

- Blackpool Flexity Tram Specifications, Bombardier Inc., 2009. Contained in professor-supplied Project Information directory.

1.5 Overview

The rest of this document will go into more detail in how tests will be conducted. Section 2 will discuss Quality Management issues such as team member roles, problem reporting procedures, and risks and assumptions. Section 3 will discuss testing strategies, section 4 will outline environment requirements, section 5 is features not to be tested (there are none), and section 6 lays out tests to be run for each individual module.

2 Quality Management

2.1 Resources, Roles, and Responsibilities

Name	**Role/Responsibility**
Thomas Bui	Track Controller Developer
Justin Carter	Track Model Developer
Patrick Flaherty	Train Controller Developer
Wesley Miller	Train Model Developer
Phil Seitz	CTC Developer

The software developer is responsible for writing the unit test cases for each testable component of the software. The author shall be responsible for writing the module test cases document. A developer other than the author shall be responsible for verifying the completeness and correctness of the unit tests and module test document. The same developer is responsible for executing the unit test program and module test and composing a test report for each. The software team is collectively responsible for deciding on actions to be taken in the event that one or more test cases fail.

2.2 Schedules

TeamGantt and Microsoft Azure DevOps will be used to plan sprints and deadlines for development and testing.

2.3 Control Procedures

A GitHub repository will hold the code base for the TCSS, allowing version control to be managed by through Git's commit system.

2.3.1 Problem Reporting

Git and Github will be utilized for defect tracking.

2.3.2 Change Request

Changes and version control will be managed through GitHub in the form of pull requests and merging of branches.

2.4 Dependencies

- The host system for testing the TCSS shall utilize the Windows 10 operating system.
- The host system for testing the TCSS shall have the specified version of the JRE installed.

2.5 Risks and Assumptions

The TCSS is a large system, thus it is impossible to test every possible configuration. As a result, the assumption will be made that the designed test suite will be comprehensive enough to cover enough cases to confirm acceptable functionality. This assumption presents a risk of under testing with regards to the number of configurations testing.

3 Testing Strategy

3.1 Unit Testing

Unit testing shall be conducted by the developer during code development to ensure proper functionality. The following are examples areas of the project that must be unit-tested and signed-off before component testing:

- Loading a PLC File
- Loading a Track Layout
- Initializing a train

3.2 Component Testing

Component Testing shall occur after all unit tests have been run. Component testing shall be run by testers for each individual module separately without integrating with other modules. The following are example areas of the project that must be component-tested and signed-off before Performance and Stress Testing:

- Parsing a PLC file
- View Block attributes

3.3 Performance and Stress Testing

Performance and Stress Testing shall occur after component testing has been completed. This shall be done when all modules have been integrated together. The following are example areas of the project that must be component-tested and signed-off before Regression Testing:

- Test the interfaces between modules heavily
- Add as many trains as possible to the system
- Break rails and trains and see how the system responds

3.4 Automated Regression Testing

Automated Regression testing shall be performed to verify that previously tested features and functions do not have any new defects introduced, while correcting other problems or adding and modifying other features. Important deliverable required for acceptance into Beta Testing include:

- Application TCSS.EXE
- All seperate module .EXE files
- Installation instructions
- All documentation

3.5 Beta Testing

Once Performance and Stress Testing have been completed, beta testing shall begin. Beta Testing shall be tested by the intended audience for the software. Once the experiences by the users are forwarded back to the developers, the developers will make final changes for User Acceptance Testing.

3.6 User Acceptance Testing

User Acceptance Testing shall be the final stage of testing. The software shall be tested in the real world by the intended audience. The goal of User Acceptance Testing is to ensure the software can both handle real-world tasks and perform up to development specifications.

4 Environment Requirements

4.1 Hardware Requirements

This section is not applicable.

4.2 Software Requirements

- Windows Operating System® by Microsoft®, version 10, from Microsoft®
  - The host system of the TCSS must operate within a Windows operating system for the purpose of file navigation and management.
- Java^{TM} Runtime Environment (JRE) by Oracle®, version 8, from Oracle®.
  - The host system of the TCSS must have the specified installation of the JRE to execute the program.

4.3 Tools Required

- JUnit 5
  - Used for designing and executing tests
- IntelliJ IDEA Commmunity Edition 2019.2.3
  - IDE used to build project and manage packages.
- Git
  - Used for version control, defect tracking, and development workflow

5 Features Not To be Tested

This section does not apply.

6 Test Procedure

6.0.1 Central Office

Table 2: CTC test cases

Test Case	Inputs	Expected Outputs	Pass/Fail	Failure Description	Tester	Date
Send a Maintenance Request						
Request sent to an empty block	1) Click on block ready for maintenance 2) Click on maintenance button, select a start time and duration, and send request.	Block is flagged for maintenance at designated time, and it is reopened after a successful repair	FAIL	The request is successfully transmitted and the block is successfully close but the block never reopens.	PATRICK	12/11/19

Continuation of CTC						
Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Request sent to an occupied block	1) Click on block ready for maintenance 2) Click on maintenance button, select a start time and duration, and send request.	Block is flagged for maintenance at designated time, and it is reopened after a successful repair (request will delay until occupied block opens)	FAIL	The request is successfully transmitted and the block is successfully close but the block never reopens.	PATRICK	12/11/19
Dispatch a Train						
Send a Dispatch request in Manual Mode	1) Start a new dispatch and select manual mode 2) Add stops to the schedule and choose an arrival time for the last station 3) Dispatcher confirms dispatch.	Dispatch request is delayed to be sent until the appropriate calculated time based on the schedule and arrival time. A suggested speed and authority are sent to the starting block at the correct time	PASS		JUSTIN	12/11/19
Send a Dispatch Request in Automatic Mode	1) Start a new dispatch and select automatic mode 2) Import a schedule file for the current dispatch 3) Dispatcher confirms dispatch.	Dispatch request is delayed to be sent until the appropriate calculated time based on the schedule and arrival time. A suggested speed and authority are sent to the starting block at the correct time	FAIL	When timer starts, an error is thrown with the Wayside Controller	JUSTIN	12/11/19
Viewing Map Information						

Continuation of CTC						
Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Updating CTC view of train system	1) Block occupancy, switch conditions, and light state returned by track controllers 2) Values in track model object in CTC updated to new values 3) GUI of map view updated based on new track model values	After updating the internal representation of the track layout, the GUI view of the track is updated to view the current track conditions	FAIL	Map does not update, however the background data does update properly and can be displayed through viewing a train or track block	JUSTIN	12/11/19
Updating Train Locations	1) New occupied block locations compared to old locations 2) The trains assigned to each dispatch are assigned to a current occupied block based on CTC calculations	Train locations of each train in each dispatch are updated to reflect the current track conditions	PASS		JUSTIN	12/11/19
Viewing Current Block Info	1) Dispatcher selects a track section to view 2) Dispatcher selects a specific block from that section to view 3) Formatted block information is displayed to the screen	Formatted block information is added to the GUI and displayed on the screen	PASS		JUSTIN	12/11/19

Continuation of CTC						
Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Viewing Current Train Info	1) Dispatcher selects an active dispatch to view 2) Dispatcher selects the train from that dispatch to view 3) Formatted train information is displayed to the screen	Formatted train information is added to the GUI and displayed on the screen	PASS		JUSTIN	12/12/19

6.0.2 Track Controller

Table 3: Track Controller test cases

Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
The CTC sends suggested speed and dispatch						
Get suggested speed and dispatch	1) Track Controller calls the sendDispatch method with arguments of suggested speed and authority	The Track Controller stores the values locally and also passes them on to the Track Model	PASS		PHIL	12/12/19
The Track Engineer uploads a PLC file						
Upload a PLC file	1) Track Controller shall parse the PLC file. 2) Assert that the array list returned is not null.	The Track Controller shall update the values for switches, lights, and railroad crossing	PASS		PHIL	12/12/19
The Track Engineer toggles a switch						
Track Engineer changes a switch	1)Track Engineer changes the switch positioning on a block	The switch positioning shall update.	PASS		PHIL	12/12/19
The Track Engineer toggles a railroad crossing						

Continuation of Track Controller						
Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Track Engineer changes railroad crossing	1)Track Engineer changes the railroad crossing positioning on a block	The railroad crossing positioning shall update.	PASS		PHIL	12/12/19
The Track Engineer toggles lights						
Track Engineer changes lights on the wayside controller	1)Track Engineer toggles the lights	The lights shall update to red or green.	PASS		PHIL	12/12/19
Close block for maintenance						
CTC sends a valid speed and authority to the Track Controller	1)Track Controller sends authority to Track Model to close a block 2) Track Model returns boolean back to Track Controller and updates block value	The block will be closed for maintenance	FAIL	Able to close, but not open.	PHIL	12/12/19

6.0.3 Track Model

Table 4: Track Model test cases

Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Track Builder uploads track file						
Invalid Track File Input	1) Formatted .xlsx file for building track	System should inform the user (track builder) that the inputted file is invalid along with a reason	FAIL	Track file is automatically uploaded into the system	PHIL	12/11/19

Continuation of Track Model

Test Case	Inputs	Expected Outputs	Pass/Fail	Failure Description	Tester	Date
Valid Track File Input	1) Formatted .xlsx file for building track	System allow the user to confirm that the track built by the system is correct, then continue with execution	PASS		PHIL	12/11/19
Train Model requests next block in route						
Train Model requests properties of next block	1) Train model 2) Current block object occupied by Train Model 3) Track object	The next block along the train's path should be returned, properly accounting for the train's current direction of travel	PASS		PHIL	12/11/19
Track Controller sends suggested speed and authority to block						
Track Controller gives command to initialize train	1) Train calculates current forces on it based on physical states such as grade 2) Train Model calculates its acceleration based on the force produced by the power command 3) Train Model calculates its current velocity based on acceleration	Train Model will update its current acceleration and speed fields	PASS		PHIL	12/11/19

Continuation of Track Model						
Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Track Controller gives command to close block	1) Train calculates current forces on it based on physical states such as grade 2) Train Model calculates its acceleration based on the force produced by the power command 3) Train Model calculates its current velocity based on acceleration	Train Model will update its current acceleration and speed fields	PASS		PHIL	12/11/19
Track Controller sends suggested speed and authority to dispatch train	1) Train calculates current forces on it based on physical states such as grade 2) Train Model calculates its acceleration based on the force produced by the power command 3) Train Model calculates its current velocity based on acceleration	Train Model will update its current acceleration and speed fields	PASS		PHIL	12/11/19
The Track Controller controls peripheral track features						
Control Railroad Crossing	1) Block on which RXR is located 2) Boolean input to specify commanded state of railroad crossing signal (up/down)	Railroad crossing state reflects commanded state				

Continuation of Track Model						
Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Control Switches and Wayside Lights	1) Block on which switch is located 2) Boolean input to specify commanded orientation of switch (straight/curved) 3) Boolean input to specify state of lights (red/green)	State of switch and wayside lights reflects commanded state				
Murphy causes track failure						
Create block failure	1) Murphy calls method on specific track block to set failure state	Block state indicates failure	FAIL	Murphy implementation is not complete for each module	PHIL	12/11/19

6.0.4 Train Model

Table 5: Train Model test cases

Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
The Track Model sends a new Suggested Speed and Authority						
Without antenna failure	1) Track Model calls the passCommand method with arguments of suggested speed and authority	The Train Model stores the values locally and also passes them on to the Train Controller	PASS		PATRICK	12/11/19
With antenna failure	1) Track model attempts to call the passCommand method	Values for Suggested Speed and Authority remain unchanged in both the Train Model and Train Controller	PASS		PATRICK	12/11/19
The Track Model tells the Train Model to update the number of passengers						

Continuation of Train Model

Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Train has no passengers	1) Train Model tells the Track Model how much free space it has 2) Track Model adds a random number of passengers equal to or less than that number	Train Model will update its passenger field to that number and update mass	PASS		PATRICK	12/11/19
Train has some passengers	1) Train Model removes a random number of passengers less than or equal to its current number of passengers 2) Train Model tells Track Model how much free space it has 3) Track Model adds a random number less than or equal to the free space	Train Model will update its passenger field to reflect both changes and update mass	FAIL	The train model does not correctly change the number of passengers after it already has some. Unclear if this is caused by the track modules or the train modules.	PATRICK	12/11/19
The Train Controller sends a power command to the train						
No engine failure	1) Train calculates current forces on it based on physical states such as grade 2) Train Model calculates its acceleration based on the force produced by the power command 3) Train Model calculates its current velocity based on acceleration	Train Model will update its current acceleration and speed fields	PASS		PATRICK	12/11/19

Continuation of Train Model

Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
With Engine Failure	1) Train takes power command but makes no updates to its acceleration	Train will not accelerate	PASS	Train still changes acceleration if it was moving when the engine failed. e.g. to decelerate from friction	PATRICK	12/11/19
The Train Controller activates the service brake.						
Without brake failure	1) Train Controller calls method to activate service brake	Train Model begins to decelerate at max safe rate (see Blackpool specifications document)	PASS		PATRICK	12/11/19
With brake failure	1) Service brake failure state set to true 2) Train controller calls method to activate service brake	Train's speed and acceleration go unchanged				
The Train Controller activates the emergency brake.						
Without brake failure	1) Train Controller calls method to activate emergency brake	Train Model begins to decelerate at max safe rate (see Blackpool specifications document)	PASS		PATRICK	12/11/19
With brake failure	1) Emergency brake failure state set to true 2) Train controller calls method to activate emergency brake	Train's speed and acceleration go unchanged	PASS	Acceleration still changes as with engine failure above due to environment but not from power command.	PATRICK	12/11/19
The Train Controller opens or closes a door						

Continuation of Train Model

Test Case	Inputs	Expected Outputs	Pass/Fail	Failure Description	Tester	Date
Train Controller calls method to open or close a certain door	1) Method is called with arguments indicating which door and the desired state of the door	Train Model will update its local door fields	PASS		PATRICK	12/11/19
Train Controller sets state of lights						
Train Controller sets lights	1) Train Controller calls method to set state of lights	Train Model sets its lights field to the given value	PASS		PATRICK	12/11/19
Murphy sets a failure state						
Murphy sets failure state	1) Murphy calls the method to set a chosen failure state to a given boolean	Train Model updates the appropriate failure state field	PASS		PATRICK FLAHERTY	12/11/19

6.0.5 Train Controller

Table 6: Train Controller test cases

Test Case	Inputs	Expected Outputs	Pass/Fail	Failure Description	Tester	Date
The Train Driver sets a new setpoint speed						
Current Speed is less than setpoint speed or suggested speed. E-Brake is False	1) Suggested Speed 2) Authority 3) Brake Status 4) Current Speed 5) Engine Status	The Train Controller Outputs a positive power command to the train model.	PASS		WESLEY	12/11/19

Continuation of Train Controller

Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Current Speed is more than setpoint speed or suggested speed. E-Brake is false	1) Suggested Speed 2) Authority 3) Brake Status 4) Current Speed 5) Engine Status	Service break is set to true.	PASS		WESLEY	12/11/19
E-Brake is true	1) Suggested Speed 2) Authority 3) Brake Status 4) Current Speed 5) Engine Status	The Train controller outputs a power command of 0. E-Brake remains true.	FAIL	Power command does not always return zero.	WESLEY	12/11/19
Engine Failure	1) Suggested Speed 2) Authority 3) Brake Status 4) Current Speed 5) Engine Status	The Train controller outputs a power command of 0. E-Brake is set to true.	FAIL	The train controller outputs the same value as prior, and does not adapt for the engine status	WESLEY	12/11/19
The Train Model is operating in automatic mode and E-Brake is false						
Authority is greater than 1, Current Speed is less than suggested speed	1) Suggested Speed 2) Authority 3) Current Speed	The Train Controller Outputs a positive power command and the train model reacts accordingly.	PASS		WESLEY	12/11/19
Authority is greater than 1, Current Speed is greater than suggested speed	1) Suggested Speed 2) Authority 3) Current Speed	The Train Controller Outputs a S-Brake command and the train model reacts accordingly.	FAIL	The train controller does not always immediately send an sbrake command to the train	WESLEY	12/11/19

Continuation of Train Controller						
Test Case	**Inputs**	**Expected Outputs**	**Pass/Fail**	**Failure Description**	**Tester**	**Date**
Authority is less than 1	1) Suggested Speed 2) Authority 3) Current Speed	The Train Controller Outputs a S-Brake command and the train model.	PASS		WESLEY	12/11/19
Engine Failure	1) Suggested Speed 2) Authority 3) Brake Status 4) Current Speed 5) Engine Status	The Train controller outputs a power command of 0. E-Brake is set to true.	FAIL	The train controller does not always output a power command of 0	WESLEY	12/11/19
E-Brake is True						
E-Brake remains true	None	The Train controller outputs only an E-Brake signal to the the train model.	PASS		PATRICK	12/11/19
E-Brake is turned off	E-Brake is Toggled Off	The Train Controller returns to the operating mode it was in before the E-Brake was activated and behaves appropriately.	PASS		WESLEY	12/11/19
Fault Detected						
Any fault is detected by the train model	Fault Notification	The Train Controller sets E-Brake to true, updates the display accordingly, and engages the E-Brake of the train model.	FAIL	The train controller does not properly detect failure states of the train model	WESLEY	12/11/19