COE 1186 Software Engineering: Train Model

Software Testing Plan

Version 2.0.0

April 22, 2015

Originator and Author:

Justin Ying

Team Members:

Trevor Leong

Shalin Modi

Phil Scott

Zachary Ovington

Prepared for

Software Engineering

Instructor: Joseph A. Profeta III, Ph.D

Spring 2015

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 3/25/2015 | Initial Creation | Justin Ying | -- |
| 4/22/2015 | Revision 1 | Justin Ying | Includes specific actions that should be tested. Now follows the IEEE standard test plan in format and included information. |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Test Plan has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Justin Ying | Originator | 4/22/2015 |
|  | Joseph A. Profeta | Instructor |  |
|  |  |  |  |

**Table of Contents**

1. Test Plan Identifier
2. References
3. Introduction
4. Test Items
5. Software Risk Issues
6. Features to be Tested
7. Approach
8. Item Pass / Fail Criteria
9. Test Deliverables
10. Responsibilities

**TEST PLAN IDENTIFIER: CTC Module Version 4.0 Test Plan**

**REFERENCES:**

* Component Communication Document. Shalin Monks, 2015
* IEEE Test Plan Template. <http://gerrardconsulting.com/tkb/guidelines/ieee829/main.html>
* Zachary Ovington’s Test Plan. TrainModelTestPlan.docx

**INTRODUCTION:**

This section is a summary of the Software Test Plan for the CTC module 1.0.0 as part of Train Control System 1.0.0. The purpose, scope, and overview of the test plan is given below, along with any relevant acronyms.

**Purpose**

This document defines the test plan that will be used to verify the functionality of the CTC module. The intended audience of this document is Joseph Profeta, the Port Authority of Allegheny County, other members of the Shalin Monks software development group, and any developers who may maintain or expand on this software.

**Scope**

Subsequent sections detail test plan specifics for verifying CTC functionality, such as the testing frameworks that are used, equivalence partitions for tests, and examples of specific test cases.

**Definitions, Acronyms, and Abbreviations**

1.3.1 Authority: number of blocks the train is permitted to travel

1.3.2 Block: a section of a railway line which maps to a single block signal

1.3.3 GUI: Graphical User Interface

1.3.4 JSON: JavaScript object notation

1.3.5 CTC: Centralized Traffic Control

1.3.6 UI: User Interface

1.3.7 OS: Operating System

1.3.8 Suggested Speed: the speed the CTC sends to a wayside controller

1.3.9 Wayside Controller: controller associated with a block of track

1.3.10 Dwell: the length of time a train will stay at a station

1.3.11 QUnit: Javascript testing framework

1.3.12 JUnit: Java testing framework

1.3.13 View Binding: the act of associating part of a controller with the view in such a way that the view is automatically updated when the controller changes.

1.3.14 Execution (Executing): term that describes when simulation has started.

**TEST ITEMS:**

The following is a list, by version, of the items to be tested:

* CTC Server version 4.0

Includes static file server, dynamic request handler, message handler

* CTC Client version 4.0

Includes HTML, Javascript, CSS, manually generated JSON files

**SOFTWARE RISK ISSUES:**

There are several parts of the project that are not within the control of the Shalin Monks but have direct impacts on the operability of the CTC. Problems with the below libraries and software may result in software failure, but testing these libraries is not within the scope of the project.

* jQuery Javascript Library
* KnockoutJS Javascript Library
* Browser used to display the CTC

**FEATURES TO BE TESTED:**

The following is a list of the features to be focused on during testing of the application:

* Creating a train
* Scheduling a train
* Updating track properties
* Displaying updated train position
* Searching for a train
* Displaying station throughput metrics
* Importing a schedule
* Exporting a schedule
* Executing a schedule

**SPECIFIC ACTIONS TO BE TESTED:**

* The initial page shall display Track Status.
* Clicking on a block of track in the table should display the block status in the sidebar.
* Selecting a value for the maintenance drop down shall change the value and persist.
* Typing specific queries of the correct format shall show only relevant blocks in the table.
* Clicking on Crossings and Switches tab shall display the Crossings and Switches page.
* Clicking on Trains and Scheduling tab shall display the Trains and Scheduling page.
* Clicking on Track Status tab shall display the Track Status page.
* The Crossings and Switches page shall display information about crossings and switches for the current track model.
* Clicking on Add Train shall add a train to the list of trains with default values.
* Selecting a train from the list of trains shall display train information in the sidebar.
* Train information shall be updated as the user edits the information in the train information sidebar.
* Clicking Add Stop in the train information sidebar shall add a stop to the list of stops with the specified values.
* Clicking Delete Train shall remove a train from the trains list as long as the schedule is not executing.
* Clicking close shall close the train information sidebar and save all changes.
* Clicking Import Schedule shall display a modal where the user can enter a JSON schedule.
* In the Import Schedule modal, clicking import shall import the specified schedule and populate all fields automatically.
* Clicking Export Schedule shall display a modal with the JSON representation of the current schedule.
* Clicking Execute Schedule shall display a modal with schedule execution settings.
* In the Execute Schedule modal, clicking “execute schedule” shall start system execution at the specified rate.
* While the schedule is executing, changing the time rate modifier shall update the speed of simulation.
* Clicking “Current Time” on the toolbar shall display the Execute Settings modal.
* While the schedule is executing, play and pause buttons shall be visible next to the “Current Time” display on the toolbar.
* The play and pause buttons on the toolbar (visible during execution) shall play and pause the simulation, respectively.

**APPROACH:**

The following is the acceptance level test approach for the CTC module.

**Unit testing**

The CTC’s functionality is tied closely with the functionality of its UI and external stimulus from the user or other modules. Furthermore, the asynchronous nature of Javascript and inter-module communication makes using standard unit testing libraries difficult. As a result, all tests will be component and interface tests.

**Component testing**

Component tests will test the features listed in the previous section. Because the CTC’s functionality is tied closely with the functionality of its UI, these tests will be verified through UI stimulus to ensure that components function as intended. Components to be tested via interface tests are KnockoutJS view models representing the track, switches, crossings, trains, and time.

**Interface Testing**

Interface testing will be completed manually through the use of the CTC client UI and also the MessageTester Java class, which can simulate sending messages from other modules to test handlers in the CTC. This testing will ensure that both the user interface and the message interface available to other modules functions correctly.

**ITEM PASS / FAIL CRITERIA**

All component and interface tests are expected to pass, since they are all necessary for software functionality. If a component or user interface test fails, CTC development will continue until the tests pass. If a module interface test fails, the team will be consulted to resolve the failure.

**ITEM PASS / FAIL CRITERIA**

* Any component test failures that were not resolved by delivery
* Interface test cases and results

**RESPONSIBILITIES**

The entire development team will be responsible for ensuring that all interface tests pass. The CTC developer will create individual unit tests for handling requests over the interface that the CTC implements. Furthermore, the CTC developer is responsible for ensuring that all component and unit tests relating to the CTC pass.