Train Trax: Train Monitor for Positive Train Control Test Beds

System Test Plan

Document

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author |
| 1.0 | 10/22/2015 | Initial Version. | Stephen Jalbert  Rashad Madyun  Corey Sanders |
| 1.1 | 11/9/2015 | Updated the excluded software list. Clarified description of testing procedures. | Stephen Jalbert |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 2](#_Toc434736521)

[Introduction 5](#_Toc434736522)

[Glossary 5](#_Toc434736523)

[Tested Features 6](#_Toc434736524)

[Features Not Tested (Per Cycle) 6](#_Toc434736525)

[Test Items 6](#_Toc434736526)

[Testing Strategy and Approach 7](#_Toc434736527)

[System Testing 7](#_Toc434736528)

[Integration Testing 8](#_Toc434736529)

[Boundary Testing 9](#_Toc434736530)

[Motion Detection Unit 9](#_Toc434736531)

[Train Navigation Engine 9](#_Toc434736532)

[Train Terminal Display UI 10](#_Toc434736533)

[Train Navigation Database 10](#_Toc434736534)

[Unit Testing 10](#_Toc434736535)

[Description of Functionality 10](#_Toc434736536)

[Features 10](#_Toc434736537)

[Use cases 11](#_Toc434736538)

[Arguments for Tests 11](#_Toc434736539)

[Expected Output 11](#_Toc434736540)

[Specific Exclusions 11](#_Toc434736541)

[Dependencies 11](#_Toc434736542)

[Success/Failure Criteria for Test Cases 12](#_Toc434736543)

[Entrance and Exit Criteria 13](#_Toc434736544)

[Test Suspension Criteria and Resumption Requirements 13](#_Toc434736545)

[Test Deliverable and Status Communication Vehicles 13](#_Toc434736546)

[Testing Tasks 13](#_Toc434736547)

[Hardware and Software Requirements Problem Determination and Correction Responsibilities 14](#_Toc434736548)

[Staffing and Training Needs/Assignments 14](#_Toc434736549)

[Test Schedules 14](#_Toc434736550)

[Risks and Contingencies 15](#_Toc434736551)

[References 15](#_Toc434736552)

# Introduction

In a real-word train environment, trains often carry very valuable assets: people, oil, merchandise, etc. It is important for rail system to be able to track the location of each train in order to prevent collisions and to monitor the state of trains in the event of attack.   
The computer engineering department owns a Positive Train Control Test Bed that is intended to mirror a typical train environment. The purpose of the train track is to be a teaching tool for instructing students on creating safety critical software. It is desired for the department Positive Train Control Test Bed to be able to track the location in for each train for this reason. Like subway trains, the department Positive Train Control Test Bed is completely indoors, so a Global Position System (GPS) is not possible.   
The purpose of this document is to describe how to test that the Train Trax Train Monitor correctly assists the department with tracking trains as they move along the Positive Train Control Test Bed. It describes how testing will be conducted to ensure measure the quality of the system in both meeting the requirements discussed in the software requirements specification as well as its construction.

# Glossary

**Positive Train Control Test Bed**  
A model train system designed to scale to represent actual railway systems. Its purpose is to facilitate the testing, design, and training of train control systems without the risk of associated performing these activities on live trains, such as bodily injury and costs for scheduling and operating full scale trains.  
  
**Train**  
To-scale model of a commercial train engine. It is the primary vehicle used to move along the test bed.   
  
**Rail Car**  
Simple wheeled container that is attached to the train to carry cargo.   
  
**Track**  
The track is a pair of metal rails that the train runs on top of to move. It provides both power and control signals to the train. It is divided into different physical pieces called sections to simplify its assembly.   
  
**Track Section**  
Segment of track that is designed to link with other segments to create the track.  
  
**Track Marker**  
Special hardware placed at different spots on the track to highlight places of interest on the track. Examples of train markers include RFID tags that are read by the train as it moves along the track, and track sections that signal when one or more trains are present.  
  
**Train Controller**  
A hardware device that is attached to the track that translates requests from operators to control the train to control signals that the train understands.   
  
**Train Monitor Terminal**  
The display equipment, such as a laptop, used by the system visually display to operators information about the test bed.   
  
**Track Switch**  
Devices on the track to control the direction of train engine movement by changing the sections of track that are connected together.   
  
**Train Operator**  
A person or machine that controls one or more of the trains on the Positive Train Control Test Bed.   
  
**Train Technician**  
A person who maintains the Positive Train Control Test Bed.   
  
**Railway System Owner**  
The entity that owns Positive Test Control Test Bed.   
  
**Train Monitor Development Team**  
A group of people who have been commissioned by the Railway System Owner to create a system for tracking the movement of trains along the railways system real time.

# Tested Features

* Report the current position of each train on the rail system.
* Report the history of each train’s movements along the rail system.
* Control switches on the rail system.
* Collect information to describe the shape and geometry of the track.
* Collect raw measurements used to estimate each train’s position.
* Alert when trains are too close together.
* Alert when there is a system failure.

# Features Not Tested (Per Cycle)

N/A

# Test Items

|  |  |
| --- | --- |
| **Test Case** | **Associated Requirements** |
| Test Acceleration Measurement | MDU-1010, MDU-1030, MDU-2010 |
| Test Rotation Measurement | MDU-1020, MDU-2020 |
| Test Train Marker Detection | MDU-1040 |
| Test Report of Train Position | TNE-1000, TNE-2000, TNE-9000, TND-5000, GUI-1000, GUI-3000 |
| Test Track Switch Control | TNE-4000, GUI-4000, GUI-6000 |
| Test Train Travel Path Display | TNE-1000, TND-1000, TND-2000, TND-5000, GUI-5000 |
| Test Persistence of Positive Train Control Test Bed Information | TND-1000, TND-1010, TND-1020, TND-2000, TND-2010, TND-2020, TND-2030, TND-2031, TND-2032, TND-2040, TND-3000, TND-3010, TND-3020, TND-3030, TND-3040, TND-3050, TND-3060, TND-4000, TND-4010, TND-4011, TND-5000 |
| Test Reporting of Track Geometry | TNE-3000, TND-3000, GUI-2000 |
| Test Saving of Track Geometry | TNE-3000, TND-3000, TND-4000, GUI-2000, GUI-9000, GUI-10000 |

Table 1 Requirements Traceability Matrix

# Testing Strategy and Approach

## System Testing

The purpose of system testing is to verify that the system's functions correctly to meet requirements. It involves running the system along on the customer's Positive Train Control Test Bed or replaying collected data from operating the system on the test bed. It is important that the primary use cases for the system be reflected also in these tests.

Test Report of Train Position:

* Compare estimated train position against observed train position from video using visible markers of know position as reference points.

Test Switch Control:

* Verify that switches change state after request for change from monitor
* Verify that LocoNet traffic is generated correctly when controlling switches from the terminal.

Test Acceleration Measurement:

* Rotate the unit along each axis of the device and verify that it reports correctly gravity along the expected axis.

Test Rotation Measurement:

* Place the device in a fixed position. Rotate the device a fixed amount of degrees along each device axis. Verify that the measured change in rotation matches the actual rotation of the device.

Test Train Marker Detection:

* Position the rail car on the track. Move the rail car along a known train marker. Verify that the device announces that the car crossed the marker.

Test Persistence of Positive Train Control Test Bed Information:

* Verify that saved items can be read after restarting the system.

Test Train Travel Path (Position History) Display / Test Reporting of Track Geometry

* Run train along a loop of track where the shape and sequence of train markers is already known
* Verify that sequence of markers read by Motion Detection Unit matches expected sequence.
* Have Tester verify that path of train shown by the Train Monitor GUI matched the shape of the test track loop.

Test Saving of Track Geometry

* Run train along a loop of track where the shape and sequence of train markers is already known
* Have Tester verify that path of train shown by the Train Monitor GUI matched the shape of the test track loop.
* Have Tester save the layout.
* Restart the Train Monitor GUI
* Load the layout.
* Verify that the path of train show by the Train Monitor GUI still matches the shape of the test track loop.

## Integration Testing

The purpose of integration testing is to identify errors with how the main components of the system interact with each of other and the Positive Train Control Test Bed. The plan for testing each main component is described below.

### Boundary Testing

One cross-cutting concern for testing is checking how the system handles the range of values that are input into the system. It is important to test how Train Trax responds specifically to input that is provided either directly from or from observing its environment. For this purpose, the following type of boundary testing will be conducted where necessary:

* negative values
* zero
* positive values
* maximum value
* maximum value +1
* minimum value
* minimum value-1

### Motion Detection Unit

Create a listener program that will record all of the radio messages sent by the motion detection unit, save them, and decode them. Compare the decoded output against expected test values.

Test cases include:

* Test Bounds of Gyroscope Measurements
* Test Bounds of Accelerometer Measurements
* Test when Gyroscope is unresponsive.
* Test when Accelerometer is unresponsive.
* Test when RFID Reader is unresponsive.
* Test when Optical Sensor is unresponsive.

### Train Navigation Engine

Feed in previously recorded measurements from the track and synthetic data generated from ideal calculation of the track to simulate input from the Motion Detection Unit. Compare messages output to LocoNet to confirm control behavior of Engine. Compare return values from Engine calls to against expected values from recorded measurements based on observations of test rail car from video recorded when each recorded test sample was collected.

Test cases include:

* Test when Motion Detection is unresponsive.
* Test when Motion Detection Unit reports an error.
* Test reporting of an unknown track marker ID
* Test reporting of an unknown train ID
* Test when Train Navigation Database is unresponsive.
* Test when Train Controller is unresponsive.
* Test when Train Controller reports an error.
* Test Bounds of Track Marker (RFID tag) positions.

### Train Terminal Display UI

Have Tester confirm that expected visual behavior of the display is what is observed. Use a Train Navigation Engine Driver and Train Database Driver to feed input to the display and confirm output. Each Driver emulates the behavior and functions of the system component it represents.

Test cases include:

* Test when Train Navigation Database is unresponsive.
* Test when Train Navigation Database reports an error.
* Test when Train Navigation Engine reports an error.
* Test when the Train Navigation Engine cannot be located.
* Test Bounds of Train Position Estimates
* Test Bounds of Track Marker (RFID tag) positions.
* Test Bounds of Track Layout

### Train Navigation Database

Write test values to the database. Restart the system. Confirm that values written to the database can be read back. Remove values from the database. Restart the system. Confirm that values no longer exist in the database.

Test cases include:

* Test when cannot connect to backend Database.
* Test when cannot read from backend Database.
* Test when cannot write to backend Database.

## Unit Testing

The purpose of unit tests is to identify as many errors as possible with the implementation of system. Unit Tests should be created to test all of the operations performed by each object in the software. They will also be created to verify that each component of the software behaves according to its responsibilities described in the software design document. Automated unit testing frameworks, such as JUnit and Arduino Unit will be used perform all of the unit tests created. EclEmma will be used to verify the amount of code coverage performed by the tests. 100% structural and branch Code Coverage is required for the product to ensure that all code in the software of the system is being exercised.

# Description of Functionality

## Features

* Report the current position of each train on the rail system.
* Report the history of each train’s movements along the rail system.
* Control switches on the rail system.
* Collect information to describe the shape and geometry of the track.
* Collect raw measurements used to estimate each train’s position.
* Alert when trains are too close together.
* Alert when there is a system failure.

## Use cases

* Monitor train
* Learn geometry of railway
* Control track switches
* Detect movement of location markers

# Arguments for Tests

* Simulated IMU messaging
* Recorded sensor measurements
* Simulated RFID tag detection
* Video of train movement
* Simulated train navigation Engine messages
* Measured position of objects on train track /test track
* User control of system

# Expected Output

* Train Position Estimation
* LocoNet message traffic
* Visual display from UI
* Radio traffic from motion detection unit
* Content of train database.

# Specific Exclusions

* Any third party source code or libraries with sufficient user base are exempted from unit testing. This is because it is assumed that the authors of the third party have independently conducted testing for them and the user base of sufficient size that significant errors have been detected and resolved.
* Excluded source code and libraries:
* Arduino Development Environment
* Junit
* Java Development Environment
* Eclipse
* Eclemma
* Octave
* Android Development Environment
* Standard Arduino Libraries
* Standard GNU C Libraries

# Dependencies

* Camera
* Video Software
* Motion Detection Unit Test Driver
* Navigation Engine Test Driver
* Train Monitor GUI Test Driver
* Train Database Driver
* Train Database Client Software
* Wireshark
* JMRI Train Control Software
* Positive Train Control Test Bed
* JUnit
* EclEmma code coverage tool
* Arduino Unit (https://github.com/mmurdoch/arduinounit)
* Test Computer
  + Windows or Linux

# Success/Failure Criteria for Test Cases

* Test Acceleration Measurement
  + All reported acceleration measurements match expected acceleration measurements.
* Test Rotation Measurement
  + All reported rotation measurements match performed rotations.
* Test Train Marker Detection
  + The train marker is reported matches the test the train marker that the tester was instructed to move the rail car across.
* Test Report of Train Position
  + The final position reported matches within a tolerance the final position that the rail car is moved.
* Test Track Switch Control
  + The track switch that is changed matches the switch that changes on the Positive Train Control Test Bed.
* Test Train Travel Path Display
  + Verify that the path rendered by the Train Monitor Terminal GUI matches the shape of the track configured for the train to move across.
* Test Persistence of Positive Train Control Test Bed Information
  + Verify that the information reported by the Train Monitor GUI when the train is moving is the same information reported when reviewing the travel history after restarting the GUI.
* Test Reporting of Track Geometry
  + Verify that the path rendered by the Train Monitor Terminal GUI matches the shape of the track configured for the train to move across.
* Pass/Fail Criteria for the Complete Test Cycle:
* A test cycle is considered complete when all of the system tests and unit tests have been completed and all have passed. Anything else is considered a failure

# Entrance and Exit Criteria

* System testing must be conducted before any release or when changes to the tests have been made. System testing is considered complete when the Test Manager verifies that all tests passed. If testing is being done because of changes to the tests, the Moderator must also verify that all tests passed. All System tests must pass before a release can be completed.
* Unit Test and Integration Testing must be conducted before any change to the code can be approved. Testing is considered complete when the Moderator verifies that all tests passed.

# Test Suspension Criteria and Resumption Requirements

* When conducting any of the types of testing, the tester must execute all of the test cases available. A tester may stop execution of a test case on the first step that fails. After all of the test cases have been executed, the tester must suspend testing until notified by a member of the team that corrections have been made. Once notified, testing will be performed for all test cases again starting with all of the test cases that failed.

# Test Deliverable and Status Communication Vehicles

* The state outcome of testing should be communicated to the team with a test report. For system tests, this will be a copy of the Test Description Document that reports the purpose for the test and the target release version and the version control ID for the specific version of code tested. The test description versions should be associated with the target release both in the version control system and any associated change request that prompted the testing.
* For unit tests and integration tests, this should be an automated test report created by the code coverage tool. The automated report should be attached to the change request associated with the features or fixes implemented and saved in the version control system.

# Testing Tasks

* Conducting unit testing of software
* Conducting integration testing of software
* Conducting system testing of software
* Updating unit tests of software
* Updating integration tests of software
* Updating system tests
* Verifying unit tests
* Verifying integration tests
* Verifying that system tests
* Verifying system test traceability to requirements

# Hardware and Software Requirements Problem Determination and Correction Responsibilities

* Every time that a change to the code base is completed, the unit tests and integration tests for the code base must be updated and reviewed. The change cannot be considered complete until all unit tests passed. The Test Manager must be included with each review and must make changes to the system tests if necessary and place those tests under review.

# Staffing and Training Needs/Assignments

* Developers are responsible for conducting and updating unit and integration tests before checking in changes.
* The Moderator is responsible for running unit tests and integration tests before declaring an issue complete. He also is responsible for verifying that unit test and integration tests are correctly updated.
* The Test Manager is responsible for running complete test cycles. He also maintains system tests and verifies that system tests trace back to requirements.

# Test Schedules

* Release 2/1/2016
  + Report the current position of each train on the rail system.
  + Collect raw measurements used to estimate each train’s position.
* Release 3/1/2016
  + Control switches on the rail system.
  + Report the history of each train’s movements along the rail system.
  + Collect information to describe the shape and geometry of the track.
* Release 4/1/2016
  + Alert when trains are too close together.
  + Alert when there is a system failure.
* Release 4/22/2016
  + Final Release Testing
  + Customer Acceptance Testing

# Risks and Contingencies

* Risks are addressed according to the procedure described in Team Operation Document.

# References

* Carnegie Mellon University System Test Plan Template. <http://www.sei.cmu.edu/productlines/ppl/system_test_plan_template.html>
* Train Trax Team Operation Document
* Train Trax Software Requirements Specification Document
* Train Trax Software Design Document