Train Trax: Train Monitor for Positive Train Control Test Beds

Software Test Description

Version 1.0

Mm/dd/yy

Stephen Jalbert

Rashad Madyun

Corey Sanders

|  |  |  |  |
| --- | --- | --- | --- |
| Revision History | | | |
| Version | Date | Description | Author |
| 1.0 | 03/18/2016 | Initial Version. Created template for Software Test Description. Included content for testing MDU and the Train Navigation Database. | Stephen Jalbert  Rashad Madyun  Corey Sanders |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History 2](#_Toc437039535)

[1 Introduction 7](#_Toc437039536)

[1.1 Purpose of this document 7](#_Toc437039537)

[1.2 Scope of the development project 7](#_Toc437039538)

[1.3 Definitions, acronyms, and abbreviations 8](#_Toc437039539)

[1.4 Overview of document 11](#_Toc437039540)

[**2** **System architecture description** 11](#_Toc437039541)

[2.1 Overview of modules / components 11](#_Toc437039542)

[2.2 Structure and relationships 15](#_Toc437039543)

[2.3 User Interface 15](#_Toc437039544)

[2.4 Hardware Interfaces 17](#_Toc437039545)

[2.5 Communication Interfaces 17](#_Toc437039546)

[2.6 Software Interfaces 20](#_Toc437039547)

[3 Detailed Description of Components 20](#_Toc437039548)

[3.1 Component Template Description 20](#_Toc437039549)

[3.2 Hardware Device: Motion Detection Unit 21](#_Toc437039550)

[3.2.1 Hardware 21](#_Toc437039551)

[3.2.2 Firmware 25](#_Toc437039552)

[3.3 Software Program: Train Navigation Service 28](#_Toc437039553)

[3.3.1 Structural 29](#_Toc437039554)

[3.3.2 Flow 31](#_Toc437039555)

[3.3.3 Behavioral 31](#_Toc437039556)

[3.3.4 Calculating Orientation from Gyroscope Data 32](#_Toc437039557)

[3.3.5 Calculating Acceleration from Accelerometer Measurements 33](#_Toc437039558)

[3.3.6 Calculation Position 34](#_Toc437039559)

[3.3.7 Requirements Traceability 36](#_Toc437039560)

[3.4 Software Program: Train Navigation Database 37](#_Toc437039561)

[3.4.1 Structural 38](#_Toc437039562)

[3.4.2 Flow 40](#_Toc437039563)

[3.4.3 Behavioral 40](#_Toc437039564)

[3.4.4 Track Geometry Measurements 41](#_Toc437039565)

[3.4.5 Requirements Traceability 51](#_Toc437039566)

[3.5 Software Program: Train Monitor Terminal GUI 53](#_Toc437039567)

[3.5.1 GUI Overview 53](#_Toc437039568)

[3.5.2 Structural 54](#_Toc437039569)

[3.5.3 Flow 60](#_Toc437039570)

[3.5.4 Behavioral 61](#_Toc437039571)

[3.5.5 Requirements Traceability 62](#_Toc437039572)

[**4** **Reuse and relationships to other products** 64](#_Toc437039573)

[**5** **Design decisions and tradeoffs** 64](#_Toc437039574)

[5.1 Database Engine Selection for Train Navigation Database 64](#_Toc437039575)

[**6** **Pseudo code for components** 65](#_Toc437039576)

[6.1 Object Position Estimation Algorithm 65](#_Toc437039577)

[7 References 68](#_Toc437039578)

# Introduction

## Purpose of this document

The purpose of this document is to describe the tests performed to verify the design og the Train Trax Train Monitor system. It will detail each test and the hardware and software configuration necessary for each test.

## System Overview

Train Trax's primary purpose is to estimate the position of each train operating along the Positive Train Control Test Bed accurately enough to allow Train Operators schedule trains to run close enough to operation on the same section of track with minimal risk of collision. Additionally, Train Trax provides a means for Train Operators to easily control switches on the train track without the need to using any additional train control software. Train Trax is only a monitor for trains, not train control software. Furthermore, the development team is to assist the department with any modifications necessary to the Positive Train Control Test Bed to support proper operation of Train Trax, including the placement of markers on the track at pre-designated locations.

Train Trax consists of hardware that is equipped on either the train engine or rail cars to measure train movement. It also consists of software that will run on existing equipment within the department to graphically display train positions and to control movement.

A unit is attached to a rail car that is equipped with an Inertial Motion Unit (IMU) that measures the acceleration and angular velocity (rotational vectors) of the rail car as it is tugged by the train along the track. This unit, called a Motion Detection Unit, will send its collected measurements over WIFI to a train monitor terminal (i.e. computer) that will estimate the train’s position using numerical integration to solve for displacement kinematic equations. The resulting position is then displayed on the terminal as well as the layout of the track itself. RFID tags, whose position is already recorded in a database, will be used as the track markers and placed strategically throughout the track so that they can correct the position calculated from IMU measurements. Lastly, the monitor terminal displays representations of all of the switches on the track and allows the user to control them through a GUI that sends LOCONET messages to the track's switch controllers, which then control relays to change a switch’s state. Train control software, such as JMRI, is expected to be used to control/throttle the movement of the train via LOCONET messages to the Train Command Station.

## Document Overview

The remainder of the STD will provide a detailed description of each test used to prove system functionality.

# Referenced Documents

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

# Test Preparations

The preparations necessary for each test will be described in each test description.

# Test Descriptions

## MDU Hardware Test

### MDU Reading test

The purpose of this test case is to test the capability of the MDU Hardware to read from the IMU and RFID tag reader.

### Requirements Addressed

|  |  |
| --- | --- |
| Requirement Number | Requirement Text |
|  |  |
|  |  |

### Prerequisite Conditions

The MDU hardware has powered on, the base computer is running with the MDU console reader program is available to be run on the computer.

### Test Inputs

None

### Expected Test Results

The MDU reader console program should show reading outputs from both the IMU and RFID tag reader

### Criteria for Evaluating Results

None

### Test Procedure

|  |  |  |
| --- | --- | --- |
| Test Step Number | Test Step Description | Requirements |
| 1 | Begin the MDU reader program |  |
| 2 | Verify the MDU reader program displays output from the IMU |  |
| 3 | Move the MDU hardware over a RFID tag and verify that the MDU reader displays that the RFID tag was read |  |

### Assumptions and Constraints

This test does not

## Train Navigation Database Tests

### Purpose

The purpose of this section to describe the series of tests necessary to verify that all of the requirements for the Train Navigation Database subsystem of Train Trax are met. The Train Navigation Database is library that is used by other Train Trax subsystems to persist information necessary for system operation and improvement. It is primarily responsible for retaining information about the shape and size of the Positive Train Control Test Bed (Track Geometry). This is necessary so that Train Trax can correlate train movement with its position along the Test Bed tracks. It is also responsible for saving position estimates and raw measurements used to derive position estimates.

### Description

Testing is conducted in accordance to the Train Trax Test Plan. Testing for this subsystem is primarily conducted through a series of integration tests created to run from the Junit unit testing framework. However, some tests are also conducted by a TestDriver console program that guides the Tester through a series of steps for testing operations that require a human to be in the loop.

Test Results from the Test Driver and the automated integration tests should be recorded as the primary artifact for verification that testing requirements work correctly.

The subsections that follow provide detailed information on how to conduct testing for the Train Navigation Database subsystem.

### Required Tools

* JUnit 4
* Eclipse Mars (4.5)
* My SQL Server (5.5.47)
* Train Trax My SQL Table Creation Script
* Java 7
* MYSQL JDBC Driver (Connector/J 5.1.38)

### Test Procedures

#### Open A Train Trax Workspace

Test sequence describes what is necessary to get the eclipse configured to run testing for the Train Navigation Database.

|  |  |  |
| --- | --- | --- |
| Description | Expected Result | Pass/Fail |
| 1. Open Eclipse | The eclipse application should appear upon the screen. |  |
| 1. Examine the Project Explorer window. | All of the Train Trax Projects in the CPE656TL/source directory should appear in the Project Explorer in the left hand corner of the table.  No further steps are necessary if projects are present.  If they are not present, then proceed to the next step. |  |
| 1. Click on File->Import. | The Import window should appear. |  |
| 1. Click on ‘Existing Projects into Workspace’, then click Next. | The Import window should update to the ‘Import Projects’ screen. |  |
| 1. Click on the Browse button to the right of the ‘Select root directory’ text box. | A Browse window should appear. |  |
| 1. Browse to the location of the CPE656TL/source directory. Click OK. | A list of Train Trax Projects should appear in the Projects List Box. |  |
| 1. Click on the ‘Select All’ button. | All of the Train Trax Projects should have a check mark beside their listing in the Projects List Box. |  |
| 1. Click on the Finish Button. | All of the Train Trax Projects in the CPE656TL/source directory should appear in the Project Explorer in the left hand corner of the table. |  |

#### Running Integration Tests

A description is given on how to use Eclipse to run all of the integration tests defined for the Train Navigation Database.

|  |  |  |
| --- | --- | --- |
| Description | Expected Result | Pass/Fail |
| 1. Perform the steps described in the ‘Open A Train Trax Workspace’ test sequence. | All of the Train Trax Projects are loaded into the Eclipse workspace. |  |
| 1. Right click upon the TrainNavigationDatabase.Library.UnitTests project. | A context menu appears. |  |
| 1. Click on Run As -> Junit Test | The Junit tab should appear and all of the tests for the database should be run. |  |
| 1. Examine the results from the Junit tab. | All of the tests are run. This is indicated by the following ratio: (Tests Run / Tests Available). For example, if there are 83 tests then, it should report in the Runs field “83/83”.  The Errors field reports “0”.  The Failures field reports “0”. |  |

#### Running the Test Driver

A description is given on how to use Eclipse to run the Train Navigation Database Test Driver to complete human-in the-loop testing.

|  |  |  |
| --- | --- | --- |
| Description | Expected Result | Pass/Fail |
| 1. Perform the steps described in the ‘Open A Train Trax Workspace’ test sequence. | All of the Train Trax Projects are loaded into the Eclipse workspace. |  |
| 1. Right click upon the TrainNavigationDatabase.Library.TestDriver project. | A context menu appears. |  |
| 1. Click on Run As -> Java Application | A console window appears. |  |
| 1. Follow the steps described by the console window. | A sequence of tests is conducted where the program prompts the operator for input at stages.  When testing is complete, the program should return the following: A prompt reporting “Testing Complete” in the console window, the file path for the test results file, and the final outcome of the test driver testing: “Pass” or “Fail”. |  |

# Requirements Traceability

## Traceability from Test Case to requirement

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Test Case Name | Requirement Number | Requirement text |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

### Train Navigation Database

#### Integration Tests

These are a series of tests that test the complete implementation of the Train Navigation Database as it is used by clients of the library. The intent of these tests are to verify that all public interfaces to the subsystem function correctly prior to any effort of clients to integrate with the library (hence the term integration tests).

##### Track Geometry

The Train Navigation Database is responsible for recording information about the size and shape of the rail road tracks that consist of the Positive Train Control Test Bed. This is accomplished by storing information about the train track using a bottom-up hierarchy of tables where each table represents a component that is a fundamental building block of the track. The smallest building block are track points. These are individual measurements of important positions on the test bed track. Positions are ensured to be recorded at a fixed distance to ensure a certain level of fidelity for representing the shape of the points alone. Positions are measured in inches relative to a point on the track that acts as the origin for the coordinate system. The origin is the bottom, left-most corner at the top of table that the Test Bed sits upon. The depth from the origin represents the y coordinates. The width from the origin represents the x coordinates, and the height from the origin represents the z coordinates. All of the points are stored in the Track Points tables.

Information about the proximity of a given point to other points is captured with the Adjacent Points table. It records which points are neighbors to other points. Recording adjacent points allows each point to be treated as vertex in a graph and the Adjacent Points table acts as the adjacency list for that graph.

Prior to Train Trax, the system was organized into Track Blocks where each block of the track is a section of the track that has been wired so that current from the DCC signals could be used to detect whether or not trains are present in a given section (DCC Track Block Occupancy Detection). The Track Block table is used to record which Track Points correspond to one of these track blocks. A collection of Track Blocks is what is used to represent the entire track.

There are switches used on the track to control the movement on trains between different routes on the rail system. In practice, a track switch controls the next track block that a train enters once it crosses the switch. The Train Switch table is used to record the position of each switch on the track as well as the Track Blocks that it is responsible for controlling access to.

The tests that follow are used to verify that the Train Navigation Database is correctly controlling each Track Geometry table described.

###### Track Block Repository

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that track blocks can be found when searches against all of the values of the columns in the track block table are performed. | TND-3020 |
| TestFindWithBlockNameSearchCriteria | The test verifies that track blocks can be found according to the human friendly name associated with it. | TND-3020 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all track block values if there are not any values specified to find matches for. | TND-3020 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-3020 |
| TestRemove | The test verifies that track blocks can be removed if erroneous block entries are identified. | TND-3020 |
| TestFindById | The test verifies that track blocks can be found by the unique ID associated with each entry into the repository. | TND-3020 |
| TestUpdate | The test verifies that existing track block entries can be changed. | TND-3020 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-3020 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-3020 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-3020 |
| TestAdd | The test verifies that track blocks can be added to the repository. | TND-3020 |

###### Track Point Repository

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that track points can be found when searches against all of the values of the columns in the track point table are performed. This functionality is necessary for a client to be able to calculate orientation for a track point: We need absolute position of points. | TND-3010, TND-4011, TND-3060 |
| TestFindWithNameSearchCriteria | The test verifies that track points can be found according to the human friendly name associated with it. | TND-3010, TND-4011 |
| TestFindWithBlockIdSearchCriteria | The test verifies that track points can be found according to the repository ID for the track block associated with the point. | TND-3010, TND-4011 |
| TestFindWithTagNameSearchCriteria | The test verifies that track points can be found according to the RFID Tag identifier associated with it. | TND-3010, TND-4011 |
| TestFindWithTypeSearchCriteria | The test verifies that track points can be found according to the point type associated with it. A point type is a text description of a group of points used for similar purpose. | TND-3010, TND-4011 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all track point values if there are not any values specified to find matches for. | TND-3010, TND-4011 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-3010, TND-4011 |
| TestRemove | The test verifies that track points can be removed if erroneous point entries are identified. | TND-3010, TND-4011 |
| TestFindById | The test verifies that track points can be found by the unique ID associated with each entry into the repository. This functionality is necessary for a client to be able to calculate orientation for a track point: We need absolute position of points. | TND-3010, TND-4011, TND-3060 |
| TestUpdate | The test verifies that existing track point entries can be changed. | TND-3010, TND-4011 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-3010, TND-4011 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-3010, TND-4011 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-3010, TND-4011 |
| TestAdd | The test verifies that track points can be added to the repository. | TND-3010, TND-4011 |

###### Adjacent Point Repository

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that adjacent track points on the track can be found when searches against all of the values of the columns in the adjacent point table are performed. This functionality is necessary for a client to be able to calculate orientation for a track point: We need the next point that the train will cross in order to determine the change in heading. | TND-4000, TND-3060 |
| TestFindWithAdjacentPointIdSearchCriteria | The test verifies that adjacent track points to a track point of interest can be found according to the Track Point Repository ID associated with the neighbors. | TND-4000, TND-3060 |
| TestFindWithPointIdSearchCriteria | The test verifies that adjacent track points to a track point of interest can be found according to the Track Point Repository ID associated with the track point of interest.  We need the next point that the train will cross in order to determine the change in heading. | TND-4000, TND-3060 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all adjacent track point entry values if there are not any values specified to find matches for. | TND-4000, TND-3060 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-4000, TND-3060 |
| TestRemove | The test verifies that an adjacency relationship between two track points can be removed if erroneous point entries are identified. | TND-4000, TND-3060 |
| TestFindById | The test verifies that adjacency association between two track points can be found by the unique ID associated with each entry into the repository. | TND-4000, TND-3060 |
| TestUpdate | The test verifies that existing adjacent track point entries can be changed. | TND-4000, TND-3060 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-4000, TND-3060 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-4000, TND-3060 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-4000, TND-3060 |
| TestAdd | The test verifies that an adjacency relationship between track points can be added to the repository. | TND-4000, TND-3060 |

###### Track Switch Repository

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that track switches can be found when searches against all of the values of the columns in the track switch table are performed. | TND-3030 |
| TestFindWithSwitchNameSearchCriteria | The test verifies that track switches can be found according to the human friendly name associated with it. | TND-3030 |
| TestFindWithPassBlockIdSearchCriteria | The test verifies that track switches can be found according to the repository ID for the track block that the switch directs traffic to when in Pass mode. | TND-3030 |
| TestFindWithBypassBlockIdSearchCriteria | The test verifies that track switches can be found according to the repository ID for the track block that the switch directs traffic to when in ByPass mode. | TND-3030 |
| TestFindWithPointIdSearchCriteria | The test verifies that track switches can be found according to the ID in the track point repository that stores the position of the switch. | TND-3030 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all track switch values if there are not any values specified to find matches for. | TND-3030 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-3030 |
| TestRemove | The test verifies that track switches can be removed if erroneous switch entries are identified. | TND-3030 |
| TestFindById | The test verifies that track switches can be found by the unique ID associated with each entry into the repository. | TND-3030 |
| TestUpdate | The test verifies that existing track switch entries can be changed. | TND-3030 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-3030 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-3030 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-3030 |
| TestAdd | The test verifies that track switches can be added to the repository. | TND-3030 |

##### Train Measurements

[TBD]

###### Accelerometer Measurement Repository (Incomplete)

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that accelerometer measurements can be found when searches against all of the values of the columns in the accelerometer measurements table are performed. | TND-2010 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all accelerometer measurement values if there are not any values specified to find matches for. | TND-2010 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-2010 |
| TestRemove | The test verifies that accelerometer measurements can be removed if erroneous measurement entries are identified. | TND-2010 |
| TestFindById | The test verifies that accelerometer measurements can be found by the unique ID associated with each entry into the repository. | TND-2010 |
| TestUpdate | The test verifies that existing accelerometer measurement entries can be changed. | TND-2010 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-2010 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-2010 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-2010 |
| TestAdd | The test verifies that accelerometer measurements can be added to the repository. | TND-2010 |

###### Gyroscope Measurement Repository (Incomplete)

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that gyroscope measurements can be found when searches against all of the values of the columns in the gyroscope measurements table are performed. | TND-2020 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all gyroscope measurement values if there are not any values specified to find matches for. | TND-2020 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-2020 |
| TestRemove | The test verifies that gyroscope measurements can be removed if erroneous measurement entries are identified. | TND-2020 |
| TestFindById | The test verifies that gyroscope measurements can be found by the unique ID associated with each entry into the repository. | TND-2020 |
| TestUpdate | The test verifies that existing gyroscope measurement entries can be changed. | TND-2020 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-2020 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-2020 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-2020 |
| TestAdd | The test verifies that accelerometer measurements can be added to the repository. | TND-2020 |

###### RFID Detected Notification Repository (Incomplete)

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that RFID tag detected notifications can be found when searches against all of the values of the columns in the track switch table are performed. | TND-2030, TND-2031, TND-2032 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all RFID tag detected notification values if there are not any values specified to find matches for. | TND-2030, TND-2031, TND-2032 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-2030, TND-2031, TND-2032 |
| TestRemove | The test verifies that RFID tag detected notifications can be removed if erroneous notification entries are identified. | TND-2030, TND-2031, TND-2032 |
| TestFindById | The test verifies that RFID tag detected notifications can be found by the unique ID associated with each entry into the repository. | TND-2030, TND-2031, TND-2032 |
| TestUpdate | The test verifies that existing RFID tag detected notification entries can be changed. | TND-2030, TND-2031, TND-2032 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-2030, TND-2031, TND-2032 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-2030, TND-2031, TND-2032 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-2030, TND-2031, TND-2032 |
| TestAdd | The test verifies that RFID tag detected notifications can be added to the repository. | TND-2030, TND-2031, TND-2032 |

###### Train Position Repository (Incomplete)

|  |  |  |
| --- | --- | --- |
| **Junit Integration Test Method** | **Description** | **Met Requirements** |
| TestFindWithAllSearchCriteria | The test verifies that train position estimates can be found when searches against all of the values of the columns in the track switch table are performed. | TND-1000, TND-1010, TND-1020 |
| TestFindWithEmptySearchCriteria | The test verifies that a search defaults to a listing of all train position estimate values if there are not any values specified to find matches for. | TND-1000, TND-1010, TND-1020 |
| TestFindAll | The test verifies that a listing of all stored values can be provided. | TND-1000, TND-1010, TND-1020 |
| TestRemove | The test verifies that train position estimates can be removed if erroneous estimate entries are identified. | TND-1000, TND-1010, TND-1020 |
| TestFindById | The test verifies that train position estimates can be found by the unique ID associated with each entry into the repository. | TND-1000, TND-1010, TND-1020 |
| TestUpdate | The test verifies that existing train position estimate entries can be changed. | TND-1000, TND-1010, TND-1020 |
| TestUpdateWithInvalidId | The test verifies that the repository handles when an update to a non-existent entry is requested. | TND-1000, TND-1010, TND-1020 |
| TestRemoveWithInvalidId | The test verifies that the repository handles when removal of a non-existent entry is requested. | TND-1000, TND-1010, TND-1020 |
| TestFindWithInvalidId | The test verifies that the repository handles when searching for a non-existent entry is requested. | TND-1000, TND-1010, TND-1020 |
| TestAdd | The test verifies that train position estimates can be added to the repository. | TND-1000, TND-1010, TND-1020 |

##### Test Driver

### [TBD] Table of Test Driver tests here

## Traceability Matrices

### Train Navigation Database

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Requirement | Testing Area | | | | | | | |
|  | Track Block Repository | Track Point Repository | Adjacent Point Repository | Track Switch Repository | Accelerometer Measurement Repository | Gyroscope Measurement Repository | RFID Tag Detected Notification Repository | Train Position Repository |
| TND-1000 |  |  |  |  |  |  |  | X |
| TND-1010 |  |  |  |  |  |  |  | X |
| TND-1020 |  |  |  |  |  |  |  | X |
| TND-2000 |  |  |  |  | x | x | x |  |
| TND-2010 |  |  |  |  | x |  |  |  |
| TND-2020 |  |  |  |  |  | x |  |  |
| TND-2030 |  |  |  |  |  |  | x |  |
| TND-2031 |  |  |  |  |  |  | x |  |
| TND-2032 |  |  |  |  |  |  | x |  |
| TND-3000 | x | x | x | x |  |  |  |  |
| TND-3010 |  | x |  |  |  |  |  |  |
| TND-3020 | x |  |  |  |  |  |  |  |
| TND-3030 |  |  |  | x |  |  |  |  |
| TND-3060 |  | x | x |  |  |  |  |  |
| TND-4000 |  |  | x |  |  |  |  |  |
| TND-4011 |  | x |  |  |  |  |  |  |
| TND-5000 |  |  |  |  |  |  |  |  |

# Notes

# Appendixes