Train Trax: Train Monitor for Positive Train Control Test Beds

Software Design Document

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author |
| 1.0 | 10/25/2015 | Initial Version. Created temporary template for software design specification. | Stephen Jalbert  Rashad Madyun  Corey Sanders |
| 1.1 | 11/9/2015 | Added definitions to document. Updated document with overview information about the system. Updated the Train Navigation Database design section with details of track geometry collection. | Corey Sanders |
| 1.2 | 11/16/2015 | Created an outline for how each component of Train Trax is to detail its design.  Added information about all of the interfaces planned to be used with the system.  Updated the scope to describe the flow of the system.  Added details of the design of the Train Navigation GUI, Train Navigation Service, and Train Navigation Database.  Improved details used in the SDD Design overview.  Renamed document from the “Software Design Specification” to the “Software Design Document”.  Improved the labeling of sections. Added Captions to Figures.  Reorganized the structure of the design document. | Rashad Madyun  Corey Sanders |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Revision History](#h.gjdgxs)

[**1. Introduction**](#h.1fob9te)

[1.1 Purpose of this document](#h.3znysh7)

[1.2 Scope of the development project](#h.2et92p0)

[1.3 Definitions, acronyms, and abbreviations](#h.tyjcwt)

[1.4 References](#h.3dy6vkm)

[1.5 Overview of document](#h.1t3h5sf)

[**2. System architecture description**](#h.4d34og8)

[2.1 Overview of modules / components](#h.2s8eyo1)

[2.2 Structure and relationships](#h.17dp8vu)

[2.3 User interface issues](#h.3rdcrjn)

[**3. Detailed description of components**](#h.26in1rg)

[3.1 Component template description](#h.lnxbz9)

[3.2 Motion Detection Unit](#h.35nkun2)

[3.3 Train Navigation Service](#h.1ksv4uv)

[3.4 Train Position Database](#h.44sinio)

[3.5 Train Monitor Terminal GUI](#h.2jxsxqh)

[**4.0 Reuse and relationships to other products**](#h.z337ya)

[**5.0 Design decisions and tradeoffs**](#h.3j2qqm3)

[**6.0 Pseudo code for components**](#h.1y810tw)

[**7.0 Appendices**](#h.4i7ojhp)

# Introduction

## Purpose of this document

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 the design for the Train Trax Train Monitor to assist the department with tracking trains as they move along the Positive Train Control Test Bed. It will cover the design for both the desktop application and the embedded system software.

## Scope of the development project

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

A unit is attached to a rail car that is equipped with an Inertial Motion Unit (IMU) that measures the acceleration and rotational speed of the rail car as it is tugged by the train along the track. This unit will send its collected measurements over WIFI to a monitor terminal (i.e. computer) that will estimate the train’s position and display it as well as the layout of the track itself. RFID tags 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 train controller.

## Definitions, acronyms, and abbreviations

**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**

A 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**

A 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 train technician is 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.

**Position**

A description of where a given object is located on the Position Train Control Test Bed. It uses a relative coordinate system based on the distance from a fixed point on the table.

## References

IEEE Standard 1016: Software Design Specification

## Overview of document

The remainder of the SDD will provide an overview of the system architecture and then describe the detailed design of each of the system components.

# **System architecture description**

## Overview of modules / components

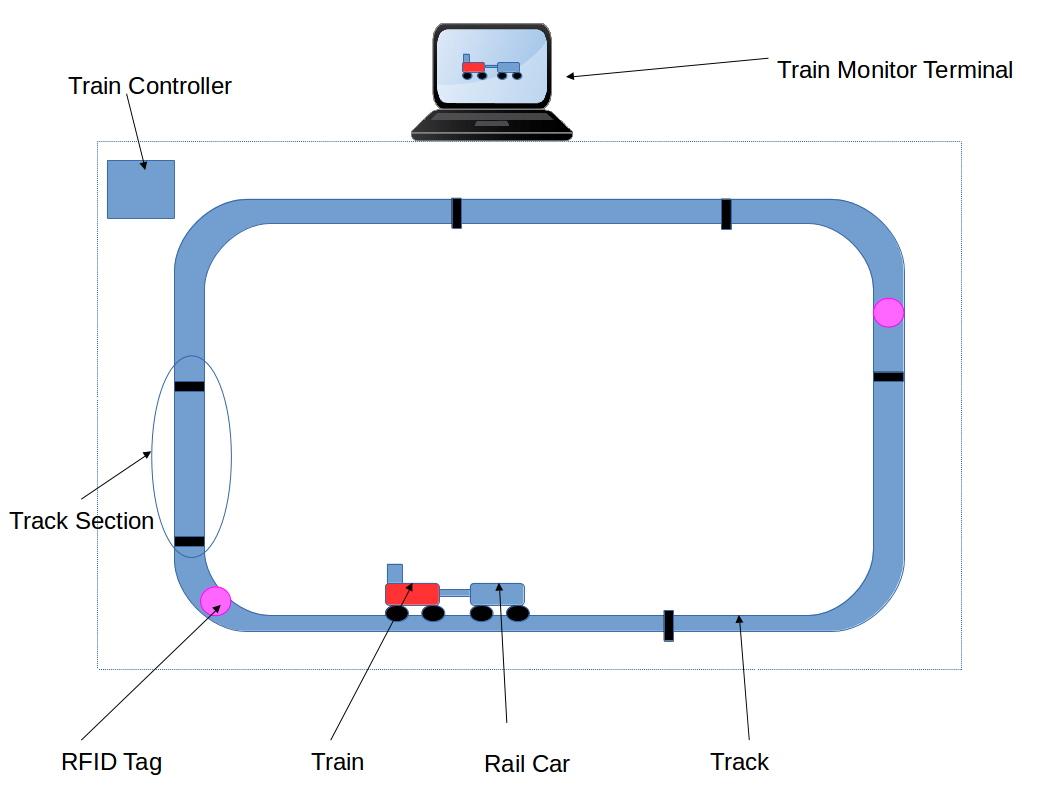


Figure Positive Train Control Test Bed

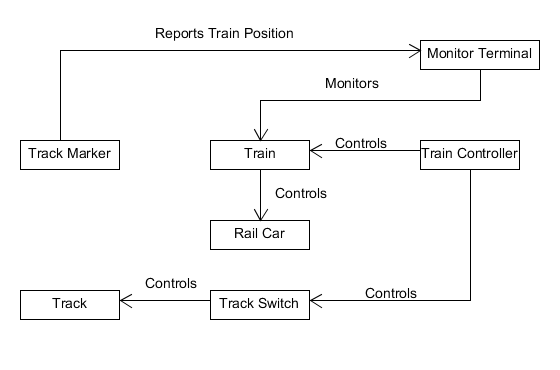


Figure Control Flow of Positive Train Control Test Bed

When the track crosses a track marker, information about the marker that was crossed is relayed to the Monitor Terminal so that it can update the train’s last known position based on the known position of the track marker. Since the rail cars are attached to the train, the train controls where the rail cars move. The Train Controller controls the speed of the train the direction that it moves along the track: either backward or forward. The Train Controller also controls track switches which in turn change the configuration of the track so that the path that the train moves along the track can be controlled. In the existing Positive Train Control Test Bed, Train Markers are actually the sections of track themselves. When a train is on the track, the Train Controller detects the current draw and sends out messaging to report that at least one train is on the section of track.

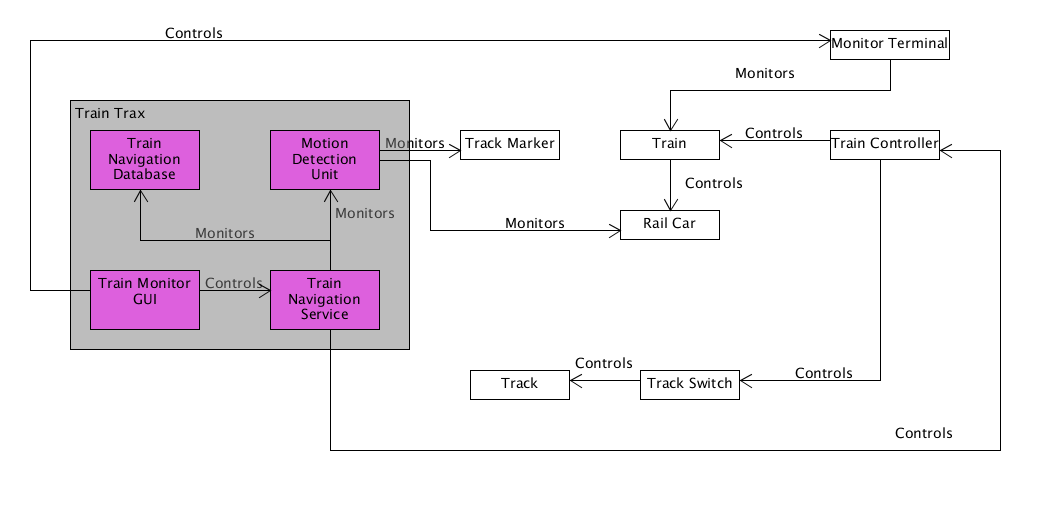


Figure Control Flow of Train Trax Components with Positive Train Control Test Bed

The Train Trax project consists of four top level components: the Motion Detection Unit, Train Navigation Service, Train Navigation Database, Train Monitor Terminal GUI. The Motion Detection Unit is the hardware that is used to measure train movement. It uses sensors to measure acceleration, and orientation of the train as well as crossing of track markers. The Train Navigation Service is a background service that is used to determine the position of each train and to control switches on the track. The Train Navigation Database stores all of the navigation information collected for the track and trains, including details on the geometry of the track (location of switches, sections of track, etc.). The Train Monitor Terminal GUI is the primary display that Train Operators use to interact with the system. It displays the train position of trains on the track and processes requests from users to change track switches.

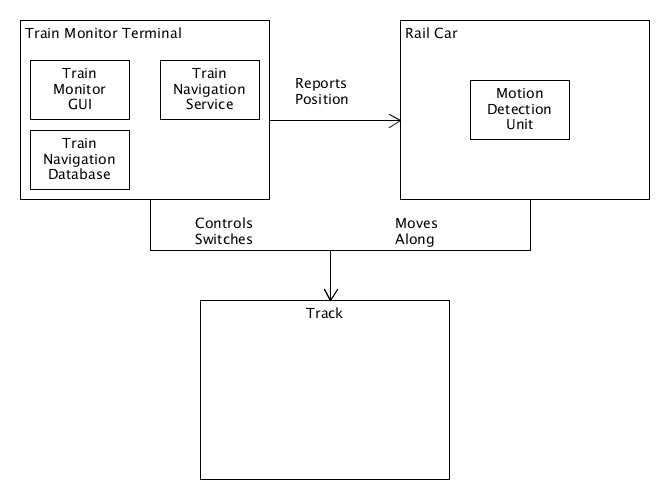


Figure Placement of System Components Within the Positive Train Control Test Bed

Train Trax integrates into the existing Positive Train Control Test Bed to function. The Train Navigation Service, Train Navigation Database, and Train Monitor GUI operate as software packages that run from the Train Monitor Terminal. They work together to allow the Train Monitor Terminal fulfill its responsibility to track trains and control switches on the track. Since there is very little space on the train engine itself, the movement of the train must be observed through an attached rail car instead. The Motion Detection Unit is equipped onto the Rail Car so that it can measurement movement of the rail car as it moves along the track.

## Structure and relationships

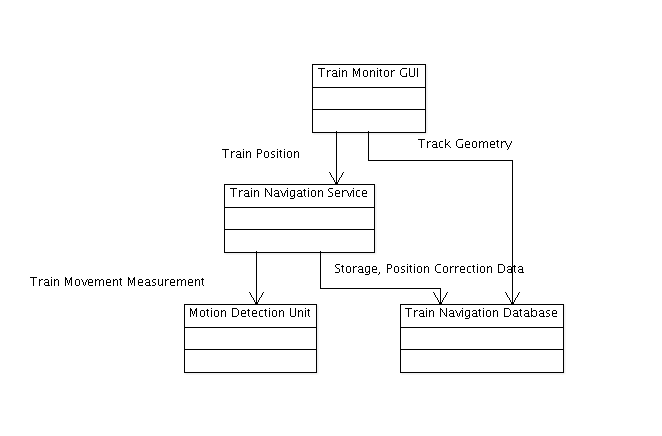


Figure Train Trax system relationships

The Train Navigation Service depends on the Motion Detection Unit to provide the raw measurements of train movement for the navigation calculations for each train. It also depends on the Train Position Database to save its calculated train position estimates and to lookup the positions of Track Markers. The Train Navigation GUI depends on the Train Position Database to render the history of each train's movement, and for track geometry information, such as the location of switches, track sections, and track markers. The GUI also depends on the Train Navigation Service to control track switches and to determine the current position of each train.

## User Interface

The following section describes an overview of the graphical user interface for the Trax Train Monitor project. The graphical user interface will consist of all windows that are required for the user to enter, store, and view information associated with the Trax system. A detailed design description of functionality and an outline of the user interface will be provided in Section 3.5 and for all of the views that will make up the graphical user interface including how most of the UI views will look, the functionality that will be executed after the interaction with the UI (i.e. a button push), and the communication between modules.

This section will present the main principles of the product's user interface. Use the personas defined in section 2.1 of your SRS to make specific examples. This section should not touch on technical details. You may want to include sketches and specific text messages.

## Hardware Interfaces

|  |  |  |
| --- | --- | --- |
| Type | Description | Purpose |
| RS-232 | Popular protocol for serial communication across a DB-9 serial cable. | Used to Program Motion Detection Unit |
| Serial Peripheral Interface (SPI) | Three-wire serial protocol used to connect a CPU to peripherals. | Used to connect Wireless Ethernet Module to Motion Detection Unit |
| Inter-Integrated Circuit (I2C) | Two-wire serial protocol used to connect a CPU to peripherals | Used to connect IMU and Optical Sensor to Motion Detection Unit. |
| Universal Serial Bus (USB) | Popular four-wire serial protocol used for connection devices to PCs. | Used to connect RFID reader to Motion Detection Unit. |

## Communication Interfaces

|  |  |  |
| --- | --- | --- |
| Type | Description | Purpose |
| IEEE 802.11 abgn Wireless Ethernet | Popular 2.4 GHz Radio Physical and Datalink Protocols for Exchanging Information Between Machines | Establishes communication with the Train Controller.  Delivers real-time measurements of train movement from the Motion Detection Unit |
| Transmission Control Protocol (TCP) / Internet Protocol (IP) | Standard Transport / Network Protocols for transferring data across the Internet and within Local Networks. | Transport for measurements from the Motion Detection Unit.  Transport for communication with the Train Navigation Database. |
| LOCONET | Ethernet-like messaging protocol used to monitor and control model train systems. | Used for communication with the Train Controller. |

## Software Interfaces

|  |  |  |
| --- | --- | --- |
| Type | Description | Purpose |
| Java Runtime API | Standard libraries provided by the Java Runtime Environment | Used to assist in Train Position Calculations By the Train Navigation Service  Used to render graphical displays for the Train Navigation GUI.  Provides interfaces for retrieving train measurements and other network data from Motion Detection Unit |
| Structured Query Language (SQL) | Standard language used to query and interact with databases. | Used for communication with the Train Navigation Database to access and store information. |
| Java Database Connectivity (JDBC) | Standard interface for connecting Java to a database. | Used to provide SQL access to the Train Navigation Database to the Train Navigation Service and the Train Navigation GUI. |

# Detailed description of components

## Component template description

Each system component of Train Trax provides the following information to deliver different representations of how each component is architected:

* Structural
* Identifies the primary entities that used to create the component, and the responsibilities for each.
* Should include Class Diagrams
* Behavioral
* Describes how subcomponents interact with each other for each of the relevant use case scenarios.
* Flow
* Describe control flow: which subcomponents are responsible for controlling other subcomponents; which subcomponents are responsible for monitoring other subcomponents.
* Describe data flow: what type of data is exchanged between subcomponents.

## Motion Detection Unit

## Train Navigation Service

The Train Navigation Service is the heart of the Train Train Trax. It is responsible for estimating the location of each train on the test bed. Also it is responsible for controlling the direction that trains move along the track by controlling the switches on the track. The Train Navigation Service is organized as a hierarchy of classes where there is one primary object acting as the root of the hierarchy that coordinates all of the actions of the support classes to achieve all of the duties of the service.

### Structural

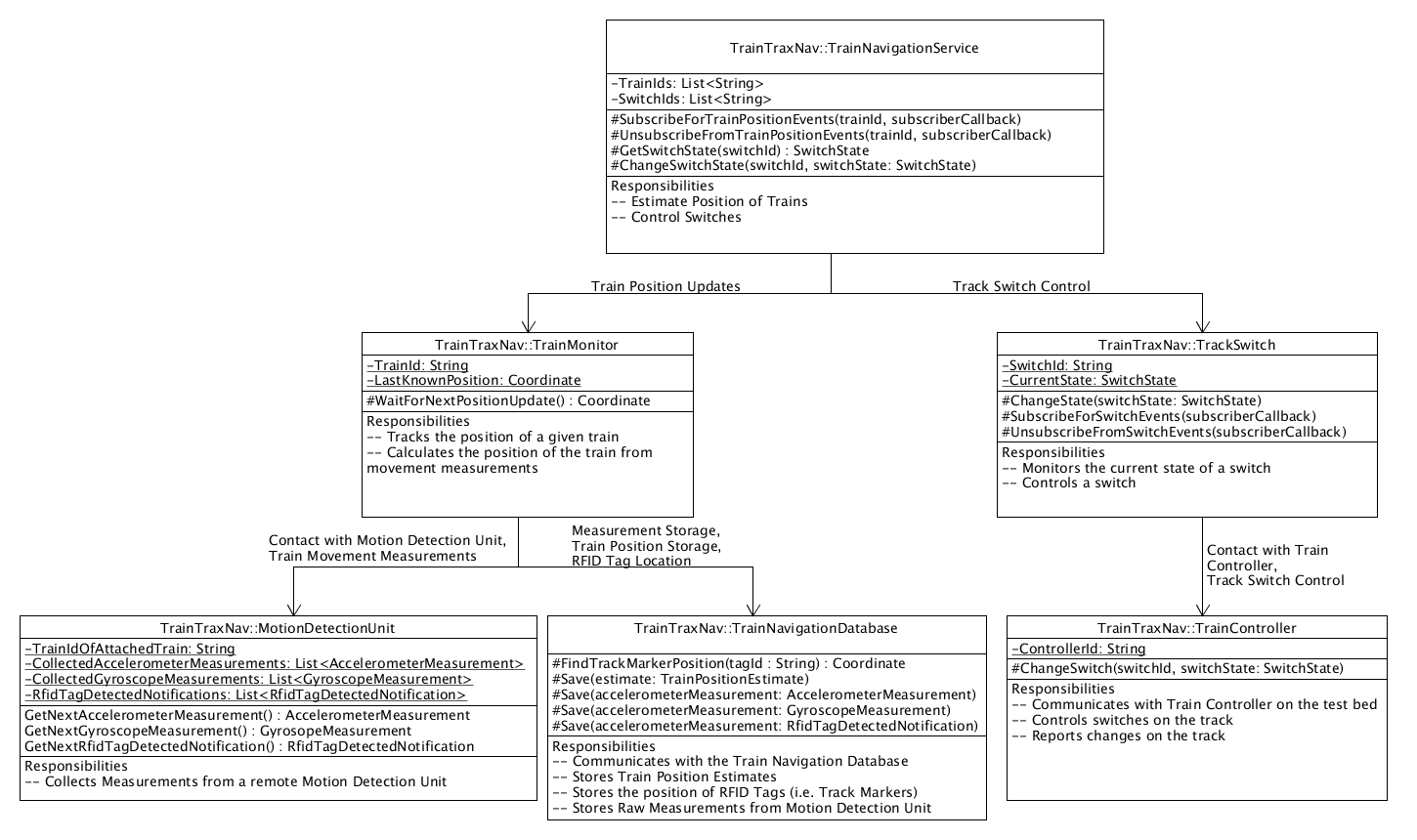


Figure Train Navigation Service Entity State Diagram

As shown in Figure 6, the TrainNavigationService is the primary object responsible for the responsibilities of the service and all subsequent classes support this class. The TrainNavigationService class acts as the front end of the service. It is the primary object that developers using the service interact with. It is responsible allowing uses to control track switches and reporting train position estimates to external locations.

The TrainMonitor class is responsible for tracking / estimating the position of a single Train on the Positive Train Control Test Bed.

The MotionDetectionUnit class is a class that represents a single Motion Detection Unit hardware device that is equipped onto a rail car that is attached to a given train. It is responsible for coordinating all information exchanged between the target Motion Detection Unit and the service.

The TrackSwitch class is a class that represents a single switch on the Positive Train Control Test Bed. It is responsible to keeping track of the current state of the switch and changing the state of the switch upon request.

The TrainController class is a class that represents represents a single Train Controller equipped on the Positive Train Control Test Bed. It is responsible for coordinating all information exchanged between the Train Controller and the service.

The TrainNavigationRepository is a datastore for all of the information that pertains to tracking trains around the Positive Train Control Test Bed, and about the track of the test bed itself. This is the primary object that interacts with any database used by Train Trax.

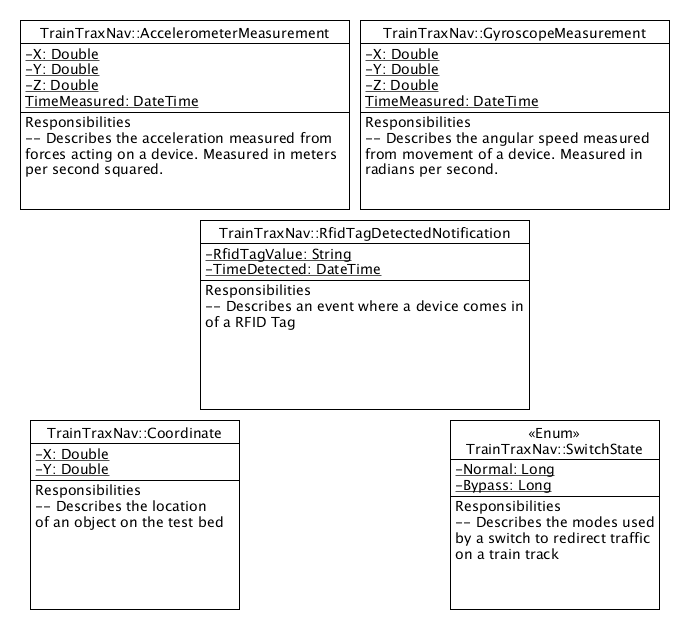


Figure Train Navigation Service Data Classes

The classes above represent all of the data that gets exchanged between classes. The AccelerometerMeasurement class represents a single sample measured from an accelerometer on the Motion Detection Unit.

The GyroscopeMeasurement class represents a single sample measured from a gyroscope on the Motion Detection Unit.

The RfidTagDetectedNotification class represents a single instance in time where the Motion Detection Unit came in contact with a RFID tag.

The Coordinate class represents the position of an object on the test bed.

The SwitchState class represents how a given switch on the track is configured to direct train traffic along the track.

### Flow

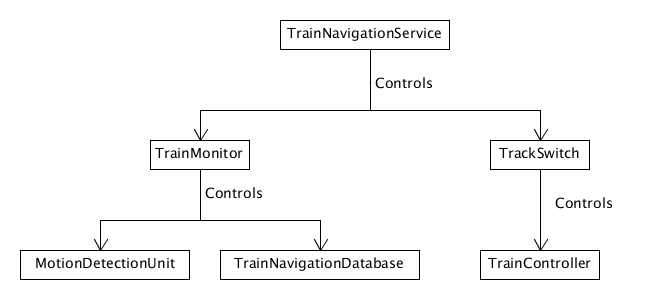


Figure Train Navigation Service Control Flow Diagram

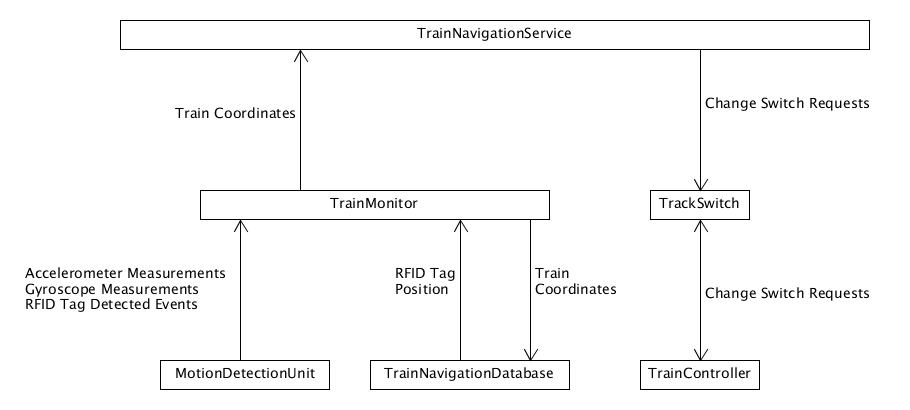


Figure Train Navigation Service Data Flow Diagram

### Behavioral

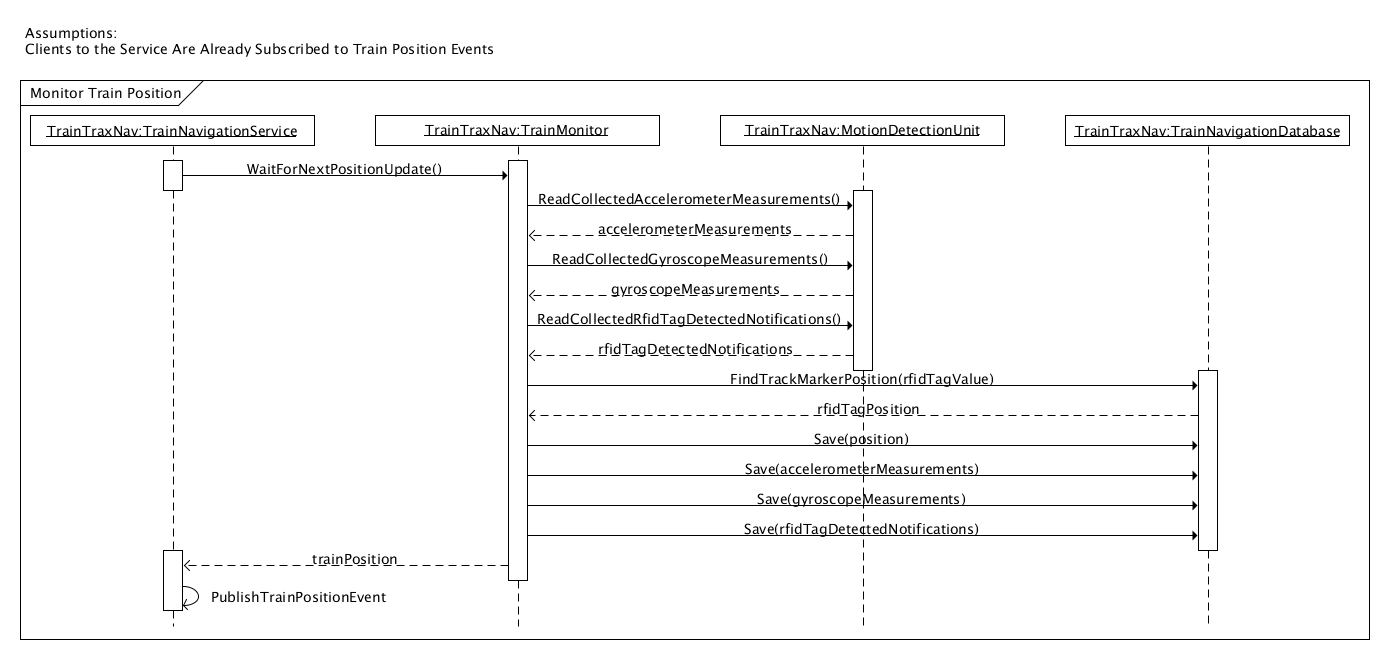


Figure Sequence Diagram for Monitoring a Train’s Position on the Test Bed

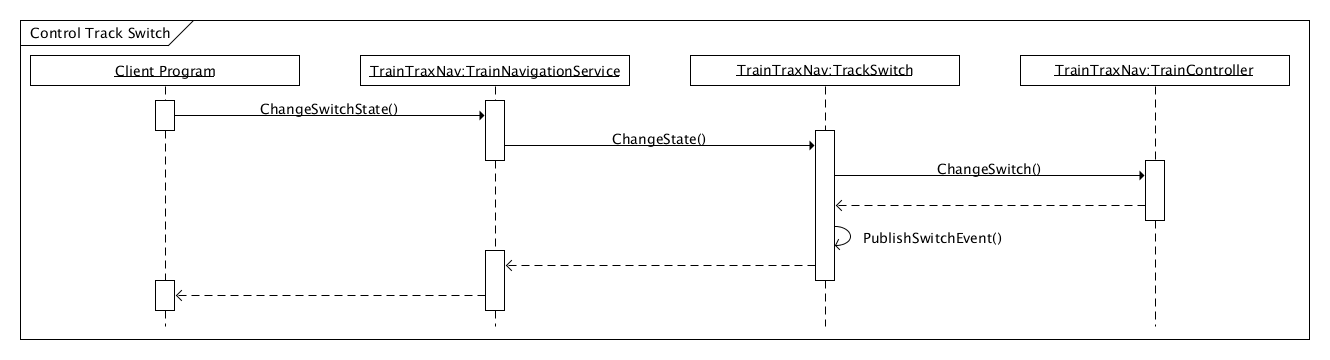


Figure Sequence diagram for Controlling a Switch on the Test Bed

## Train Navigation Database

The Train Navigation Database is used to store all track layout and positioning information for calculating a train’s position. It also stores all of the train positions reported by Train Trax to create a history of movement of each train over the period of time observed. It also provides the location of each RFID tag (i.e. Track Marker) placed on the track so that train position corrections can be made each time a train crosses a RFID tag.

### Track Geometry Measurements

#### Overview

The primary data that is stored in the Train Navigation Database is track geometry information. Track Geometry information is all of the information necessary to describe the shape and size of the track. It also includes a description of all of the relevant points of the track that are essential to making it function, such as identification of sections of track and the location of switches, power junctions, and track markers.

All of this information is necessary for the database to have enough information for the Train Navigation Service to correct its estimates of train positions as they cross track markers. Furthermore, it is necessary so that there is enough information for Train Operators to have context of where each train is located relative to both the track and each other.

Since this data is not already available, part of the design is to collect the track geometry data required for the database. The sections below describe details on the type of track geometry information collected as well as the process of collecting it.

#### Coordinate System

Train Trax uses a relative coordinate system based on the distance from a fixed point on the table. In particular, it is the bottom left corner of the table as the origin (0,0). This was selected because if we have this quality of granularity in measurement. We can still choose to relay the location of the train according to a grid if needed to simplify the problem, but we cannot get further accuracy if all that we collected was grid information.

#### Types of Data Collected

* Angle of Train from bottom corner of table
* Section of track
* Nearest junction
* Position within track
* Position
* Segments that a junction joins

#### Objects to Measure

* RFID Tags
* Switches
* NOTE: If we record just these things, then we know exactly where each section of track begins/ends. So we can use our train position data to estimate the geometry of each section of track when we begin position estimation.

#### Track Geometry Data Collection Procedure

Before you start geometry collection, partition the table into 4 sections along the length of the table where the Test Bed is located. Each section will be a phase of where data is collected. This intended to reduce the amount of work necessary for data collection at any one time, and to give us the opportunity to refine the geometry collection procedure before all of the table has been measured.

##### Tools

* Spreadsheet
  + Used to record the measurement data
* Laser Range Finders
  + Position Measurement
* 3 Pedestals
  + Placement of Laser Range Finders
  + Ensures that Range Finders are high enough to avoid any interference from objects on the track.
* Fabric Tape Measure
  + Verifies the accuracy of the laser range finder measurements
* Measurement Rail Car
  + Rail car that has been fitted with two polls each with a marker on the top that is easy for the range finders to align and measure.
  + We need two markers so that we can estimate the orientation of the rail car.

##### Setup

* Divide table into 4 Regions Along the Length of the Table
* Mark the Beginning and End of Each Region
* Place RFID Tags On Each End of Each Section of Track that is within the region.
  + If there is a section of track that overlaps regions. Only add a tag for the end is with the region.
  + If there happens to be an end that overlaps the region of interest and another region, proceed with adding the tag.
* Select the Corner of the table that will be the origin of the coordinate system.
  + It is recommend to the bottom left corner so that all measurements are positive.
* Place pedestals on the corners of the table that are not the origin.
  + Pedestals are selected because the laser range finders must be level and they need to be high enough for us to be able to measured positions on the track without any risk of interference from objects on the train track.
* Mark on each pedestal. This will be the point where laser range finders will measure from.
  + It is recommended that you try to mark points as close to the table as possible.
* With the fabric tape measure, measure the distance from the origin of each of the measuring points.
  + You can also use a one of the laser range finders to measure distance at that point as well if desired.
  + This is necessary to make any last adjustments.
* Select one of the markers on the measurement rail car to be the primary marker.
  + The primary marker is the marker that is always aligned with the position of the object of interest when measuring.

##### Measurement

**Procedure for a Given Partition of the Test Bed**

When searching for an object of interest that has not been measured yet, search from top-to-down and from left-to-right. The x-axis is the edge of the table that is along the length of the table and touches the origin. The y-axis is the edge of the table that is along the width of the table and touches the origin.

**Measuring the Position of an Object on the Track**

* Place or locate the object on the track that you want to measure.
* With the fabric tape measure, measure the distance from the object to the edge of the table that is perpendicular to the x-axis.
* With the fabric tape measure, measure the distance from the object to the edge of the table that is perpendicular to the y-axis.
* Move the measurement rail car into position so that the primary measurement marker of the car is aligned with the position on the track that is closest to the object of interest.
* Adjust the laser range finders on each pedestal so that the finders can measure the distance from the pedestal measurement point to a given measurement rail car measurement marker.
* Record the measurements of the distance from each range finder for each rail car measurement marker.
* Record the section of track that the object belongs to.
* If not already recorded, record the junctions that connect to the section of track.
* If not already recorded, record the two sections of track that are adjacent to the section of track that the object of interest is on.
* If measuring a RFID tag, record the end of the track section that it belongs to by recording the junction that it is closes to.
  + In the event, that the tag is not on an end of the track, record ‘NA’ so that it is known to be not be an end.

## Train Monitor Terminal GUI

### GUI Overview

The Train Trax system GUI will consist of a main menu that includes submenus to enter information. On the main menu the user will be able to select an option to Monitor the train. The Monitor Train window will display a diagram of the track with an icon of the train’s position and speed, and an icon of all switches on the track. From this window the user also has the option to control track switch changes, and an option to display a train’s position and speed history in database format. If the Train History option is selected from the Train Monitor window, the train’s position and speed information over a specified amount time will displayed in a excel like table format. If the Control Track Switch changes are selected, the state of the switch will be changed and the Monitor Train window updated to reflect the latest state.

### Views

#### Main Menu View

INSERT FIGURE HERE

Figure TRAX UI Main Menu View

|  |
| --- |
| **MainMenuView** |
| - trainMonitorButton : boolean = false  - trackLayoutButton : boolean = false |
| + displaytrainMonitorView()  + discovertrackLayoutView() |

Table Class Diagram for Main Menu View

|  |  |
| --- | --- |
| **Detailed Design for MainMenuView** | |
| **Attributes** | trainMonitorButton = false |
| **Operations** | displaytrainMonitor()  begin;  if(trainMonitorButton)  {  Create New Window;  GetTrackLayout from database;  Draw Diagram of Track;  Add state and location of switches to track;  Compute location of train;  Add Train location icon to display its current position;  Display speed of train;  }  End; |

Table Detailed Design for Main Menu View

#### Train Monitor Window View

INSERT FIGURE HERE

Figure TRAX UI Main Train Monitor View

|  |
| --- |
| **TrainMonitorView** |
| - trainHistoryButton : boolean = false  - numswitches : int= 0  - switchstate ; boolean = false |
| + displaytrainHistoryView()  + changeSwitchState() |

Table Class Diagram for Train Monitor View

|  |  |
| --- | --- |
| **Detailed Design for TrainMonitorView** | |
| **Attributes** | trainHistoryButton = false, switchPressed = false, numswitches = 0, switchstate=false |
|  | displaytrainHistoryView ()  begin;  if (trainHistoryButton)  {  Create New Window;  Get Train ID from Train Navigation System;  Lookup ID in Database;  Get Time/Position/Speed value  Display each interval in table for train  }  End; |
|  | changeSwitchState()  if(switchPressed)  {  begin;  Prompt user with dialog box to verify they want a switch change  If (no) exit;  For numswitches  Get Current Value of Switch (switchstate)  If True  Set Switchstate to false;  Else  Set Switchstate to true;  If Switchstate true  Switchicon is highlighted;  }  End; |

Table Detailed Design for Train Monitor View

#### Display Train History View

INSERT FIGURE HERE

Figure TRAX UI Display Train History View

|  |
| --- |
| **TrainHistoryView** |
| - trainMonitorButton : boolean = false  - trainIdList  - trainPositionTable  - timeRangeControl  - trackControl |
| + displaytrainMonitorView()  + showTrainPositionTable(trainId, dateRange)  +showPathTrainTraveled(trainId,dateRange) |

Table Class Diagram for Train History View

|  |  |
| --- | --- |
| **Detailed Design for TrainHistoryView** | |
| **Attributes** | trainMonitorButton = false, trackLayoutButton = false, |
|  | displaytrainMonitor()  begin; |
|  | displaytrackLayout()  begin; |

Table Detailed Design for Train History View

### ViewModels

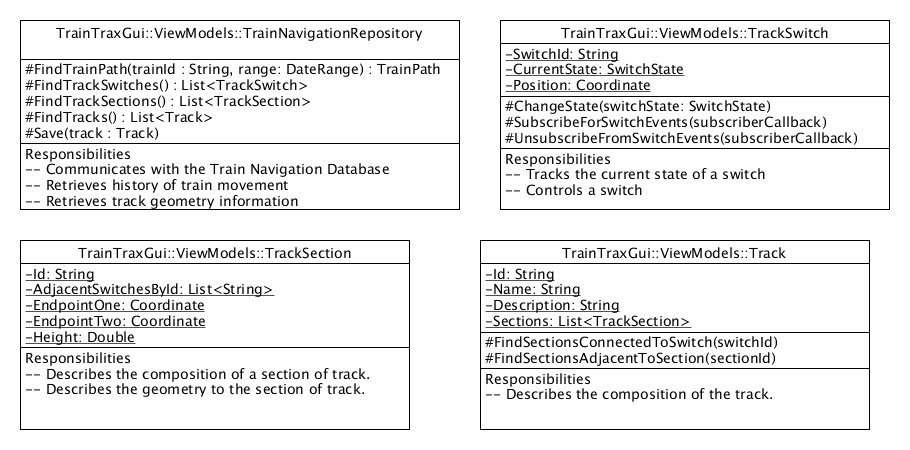


Figure Train Monitor Terminal GUI Entity State Diagram for ViewModels

The TrackSwitch class represents a single switch on the Positive Train Control Test Bed. It is responsible for reporting to users the current state of the switch. It also processes requests from users to change the state of the switch.

The Train class represents all of the information known about a given train on the Positive Train Control Test Bed. It is responsible for reporting information, such as it's current position and speed.

The TrackSection class represents a single section of track used to construct the Positive Train Control Test Bed. It include information such as an identifier for the section, other sections that it is connected to, and switches and track markers that are on it.

The Track class represents all of the information known about the complete track of the Positive Train Control Test Bed.

The TrainNavigationRepository is a datastore for all of the information that pertains to tracking trains around the Positive Train Control Test Bed, and about the track of the test bed itself. This is the primary object that interacts with any database used by Train Trax.

# **Reuse and relationships to other products**

All software for the Train Trax project will be new development and no reuse will occur.

# **Design decisions and tradeoffs**

Use this section to motivate any decisions that will help the reader understand the design that your team is using. This section can also capture good ideas that were abandoned and the reasons for leaving them out of the design.

## Database Engine Selection for Train Navigation Database

Decision: Database Selection

Description:

The Train Navigation Database requires a database service to handle saving and reading information.

Selection Criteria:

Longevity: Database must be believed to be able to be supported at least 2-3 years after the project.

Support: Reliable vehicles for the team to obtain information for questions that arise while implementing and supporting Train Trax

Ease of use: How easy it is for a new developer to understand how to setup and use the database and the software that interface with it.

Scale: (1(Poor) – 5 (Great))

Alternatives:

Alernative 1: MySql

Longevity: 5. Active community. Used in many commercial applications

Support: 5. Well documentation for using database and driver. Commercial options also available to get support.

Ease of use: 4. Setup is needed to configure the database before being able to use.

Alternative 2: SQLite

Longevity: 5. Active community. Used in many commercial applications

Support: 3. There is a lot of documentation available for setup and using the C interface and the is an active community for that. No activity communities were found for any of the Java Drivers to the the database.

Ease of use: 3. In order to use it from java, multiple java drivers have to be evaluated because of the spectrum of supported version advertised. Otherwise, a JNI interface would need to be developed to ensure that we have the support we need.

Outcome:

MySql was selected because it has the best chance of being supported long term and the fewest steps necessary to implement a working solution.

# **Pseudo code for components**

To be added as developed

# **Appendices**

To be added as necessary

Content guide for section 3.X (Table to be removed on document completion)

|  |  |
| --- | --- |
| Identification | The unique name for the component and the location of the  component in the system. |
| Type | A module, a subprogram, a data file, a control procedure, a class, etc |
| Purpose | Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements. |
| Function | What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified. |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | How the component's function and performance relate to other  components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components. |
| Interfaces | Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here. |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| Data | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary. |