SOFTWARE REQUIREMENTS SPECIFICATION

for

COE 1186 Project

Version 0.1 approved

Prepared by:
Alec Rosenbaum
Aric Hudson
Isaac Goss
Mitch Moran
Parth Dadhania

Training Montage

December 14, 2017

Contents

Revision History

Name	Date	Reason For Changes	Version
et all	September 24, 2017	Module Requirements	0.2.0
Alec Rosenbaum	September 14, 2017	init	0.1

1 Introduction

1.1 Purpose

The product whose software requirements are specified in this document is the Training Montage Rail Simulator (TMRS). This SRS covers the functional and non-functional requirements, design constraints, and functional limitations that will be observed during the development of this project. This SRS shall encompass the entire system, and will include specific requirements, constraints, and limitations for each sub-system.

1.2 Document Conventions

The following acronyms will be used throughout the paper:

TMRS Training Montage Rail Simulator

Failure A subsystem has failed when it allows two locomotives to share a block.

Unsafe A situation or subsystem is unsafe if failure is made possible by said situation or subsystem.

Vital A vital subsystem is one which shall not allow unsafe output.

PLC Programmable Logic Controller, a hardware platform which operates solely on binary values, controlled by code given by a user. The term PLC will be used interchangeably to refer to the device and the code which runs on it.

CTC Centralized Traffic Control, the module that represents the main dispatcher office.

Block The track is partitioned into blocks, what is considered an indivisible unit of track. A block has one speed limit, one grade, one altitude, no switches, and at most one train occupying it.

1.3 Overview

This document is intended as a part of the contractual agreement between Training Montage and their client for the TMRS software package. It will be used as reference by developers, project managers, testers, and the client.

It is suggested that the user read over the project scope and perspective, then visit the UI design prototype included in the Appendix. The UI design prototype will give a good sense of how users will interact with the software, and of what capabilities are provided by the system. After reviewing the UI design prototype, proceed to read the rest of the document in the order that it's presented.

1.4 Project Scope

This project shall have many deliverables, organized into three Work Packages (WPs).

WP 1. This includes project planning and requirement gathering, due Thursday, September 28, 2017. It shall contain the following deliverables:

- **WP 1.**1. The Software Requirements Specification, this very document.
- **WP 1.**2. User Interface prototype.
- **WP 1.**3. User's Manual detailing the functionality of the aforementioned UI.
- **WP 1.**4. Risk Assessment.
- **WP 1.**5. Coding Standard.
- **WP 1.**6. Defect Tracking Policy.
- **WP 1.**7. Schedule detailing our expected progress.

WP 2. This is the phase in which the system shall be designed, due Thursday, November 2, 2017.

- **WP 2.**1. System Architecture and Design.
- **WP 2.**2. Test Plan.
- **WP 2.**3. Updated Schedule with resource assignments.

WP 3. This is the implementation phase. The requirements and design shall be implemented to create a working system, due Thursday, September 7, 2017.

- **WP 3.**1. Code Base of the implemented application.
- **WP 3.**2. Installation Guide.
- **WP 3.**3. Configuration Management.
- **WP 3.**4. Executed Test Plan.
- **WP 3.**5. Defect Reports.
- **WP 3.**6. Project Presentation.

1.5 References

- We used the following pdf as a reference for how real life systems work. It is based on the English train system but still gave a good idea about the actual system.

  http://etheses.whiterose.ac.uk/14543/1/531116.pdf

- This document is based on the following SRS template.
  `https://github.com/jpeisenbarth/SRS-Tex`

2 Overall Description

2.1 Product Perspective

TMRS is a simulation and prototype bid for the North Shore Extension presented by the City of Pittsburgh, PA. In this capacity, the software is intended to allow users view of how a completed North Shore Extension control system would operate, and serve to inform the development of the system to be deployed, although no code will be shared between this prototype and the completed system.

There are five major modules which make up this system, for which the individual requirements will be specified in chapter 4.

Centralized Traffic Control (CTC) The CTC office module is the part of the software that sends trains from the train yard. A human dispatcher can view information of the entire system from the CTC office. The module can operate in automatic mode, where it dispatches and routes trains according to a set schedule, or manual mode where trains are dispatched by the dispatcher.

Wayside / Track Controller This module is a vital computing platform. The Wayside Controller module is a simulation of the many wayside controllers which sit next to track, and operate the switches and lights on the track, and provide the train traveling with within its region safe speed and authority.

Track Model The system shall have a model of the transit system track layout. This module is **not** vital.

Train Model The Train Model models the movement of the train using Newton's laws of motion assuming that the train is a rigid body point mass. This is a non-vital module.

Train Controller This module is responsible for safely dictating the train's speed by way of controlling its power, as well as controlling some smaller tasks like light and door control, and feedback about current and approaching stations. This module is VITAL.

2.2 Product Functions

The software shall:

1. Allow a dispatcher to perform tasks such as create, view, edit, and execute schedules, and also manually dispatch a train to a location.

2. Allow a wayside engineer to control the operation of switches and lights with PLC code.
3. Allow a locomotive engineer to observe automatic train operation, and also allow manual train control.
4. Move trains to their destination always in a SAFE manner.

2.3 User Classes and Characteristics

This produce is expected to be used by the following classes of users:

1. Dispatcher
2. Wayside Engineer
3. Train Conductor

It is expected none of these user classes to have technical expertise. As such, each user shall be provided with an graphical user interface that allows full functionality of each module.

It is, however, expected that Wayside Engineers are experts both on a section of track and in writing PLC code. All uploaded code is considered vital.

2.4 Operating Environment Constraints

The software shall operate on the Java Virtual Machine. The Java Virtual Machine can be installed on computers with a Pentium 2 266 MHz or faster processor, at least 128 MB of physical RAM, and 124MB of free disk space. Java supports Windows, Mac OS X, Linux, and Solaris. We will be writing software using Java Version 8. This software shall be executable on Windows 10 operating system.

2.5 User Documentation

The following documentation components shall be delivered along with the software:

1. UI Design
2. User's Manual

2.6 Assumptions and Dependencies

It is assumed that every user has a static map of the track detailing block information.

2.7 Apportioning of Requirements

All requirements are to be completed in the delivered version of the TMRS software.

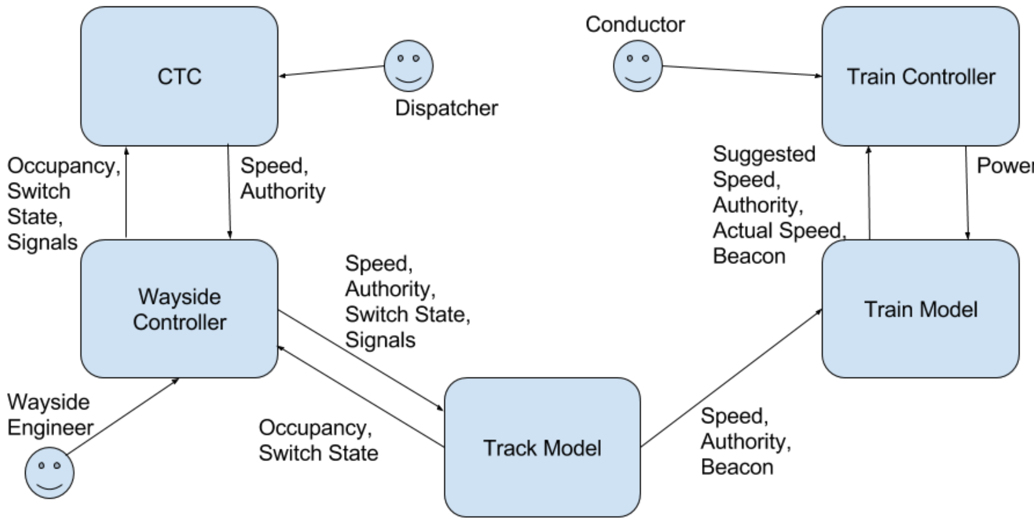
2.8 User Interfaces

There shall a be discrete user interface for each System Module. These user interfaces shall be capable of running separately, but may also be run together in order to show the full functionality of each module and how it interfaces behind-the-scenes with other modules.

UI Prototypes for each module are shown in the UI Design document. The User's Manual further elaborates on operational details and how users are expected to interface with each module.

3 System Modules

This application shall be made of 5 separate modules, each of which a team member shall own. These shall be the Centralized Traffic Controller, Wayside / Track Controller, Track Model, Train Model, and Train Controller. These modules shall have strict communication constraints and be organized in the following way.



3.1 Centralized Traffic Control (CTC) Office Module

3.1.1 Description

The CTC office module is the part of the software that sends trains from the train yard. A human dispatcher can view information of the entire system from the CTC office. The module can operate in automatic mode, where is dispatches and routes trains according to a set schedule, or manual mode where trains are dispatched by the dispatcher.

3.1.2 Interface

The CTC module interacts with two users, a human dispatcher and the wayside controller module.

1. The module shall display information on the current track layout and currently deployed trains to the dispatcher.
2. The dispatcher can manually schedule trains for the CTC to deploy.

1. The CTC gives suggested speed and authority for each train to the wayside controller module.
2. The wayside controller module shall send the CTC information on which pieces of track are occupied by trains.

There is also a communication link between the stations of the track model and the CTC to communicate information on passenger throughput.

3.1.3 Functional Requirements

1. The CTC module shall display the following track information. Some information shall be displayed in a dynamic map view and detailed information shall be displayed when selecting a block in the map view. Items indicated with an asterisk (*) are only displayed when relevant.
   - 1.1. Block ID
   - 1.2. Block Region
   - 1.3. Occupied State
   - 1.4. Speed Limit
   - 1.5. Length
   - 1.6. Grade
   - 1.7. Elevation
   - 1.8. Block Passable (Yes, Broken, or Maintenance)
   - 1.9. Heater (On, Disabled, N/A)
   - 1.10. Underground/Above Ground
   - 1.11. Light Information*
     - 1.11.1. Light State (Super Green, Green, Yellow, Red)
   - 1.12. Switch Information*
     - 1.12.1. Switch ID
     - 1.12.2. Switch State
   - 1.13. Station Information*

        1.13.1. Station Name
        1.13.2. Number of Passengers
    1.14. Railway Crossing*
        1.14.1. Activated

2. The CTC module shall let the dispatcher close and open blocks for maintenance.
3. The CTC module shall be able to operate in manual or automatic mode. In manual mode, all trains shall be manually dispatched by the human dispatcher. In automatic mode, trains shall be dispatched by the module according to a schedule.
4. The CTC module shall let the dispatcher upload a schedule to be followed in automatic mode.
5. The CTC module shall let the dispatcher view and edit the schedule in manual mode.
6. The CTC module shall display the following train information. Some information shall be displayed in a dynamic map view and detailed information shall be displayed when selecting a train in the map view.
    6.1. Train ID
    6.2. Current Block
    6.3. Suggested Speed
    6.4. Given Authority
    6.5. Origin
    6.6. Destination
7. The CTC module shall display the ambient temperature.
8. The CTC module shall display the current passenger throughput based on tickets sold at each station.

3.2 Wayside / Track Controller

3.2.1 Description

This module is a vital computing platform. The Wayside Controller module is a simulation of the many wayside controllers which sit next to track, and operate the switches and lights on the track, and provide the train traveling with within its region safe speed and authority.

3.2.2 Interface

1. The Wayside Controller shall accept suggested (non-vital) speed and authority (per train) from the CTC.
2. The Wayside Controller shall accept block occupancy and switch positions from the Track Model.
3. The Wayside Controller shall have vital outputs to the Track Model:
   3.1. Speed & Authority per train.
   3.2. Light colors per track block.
   3.3. Switch positions.
   3.4. Railway crossings.
4. The Wayside Controller shall report the following back to the CTC:
   4.1. Broken rail.
   4.2. Railway crossings.
   4.3. Block occupancy.
   4.4. Switch positions.
   4.5. Light colors.

3.2.3 Functional Requirements

1. The Wayside Controller shall be a vital computing platform.
2. The Wayside Controller shall receive suggested speed and authority from the CTC per train, and output safe speed and authority per train.
3. The Wayside Controller shall control the switching of the track.
4. The Wayside Controller shall detect broken blocks of rail.
5. The Wayside Controller shall report the state of the track, railway crossings, signals, and occupancy to the CTC.
6. This is a PLC, and shall allow a user, a Wayside Engineer, to upload a code to control it. It is assumed that this Wayside Engineer's code is vital. The implementation of this shall be diverse.

3.3 Track Model

3.3.1 Description

The system shall have a model of the transit system track layout. This module is not VITAL.

3.3.2 Interface

1. The Track Model shall accept speed and authority as inputs from the Wayside Controller.
2. The Track Model shall provide block occupancy and switch positions to the Wayside Controller.
3. The Track Model shall provide speed and authority to the Train Model.

3.3.3 Functional Requirements

1. The Track Model shall consider grade and elevation.
2. The Track Model shall be configurable.
3. The Track Model shall consider allowable directions of travel, branching, and speed limits.
4. The Track Model shall be able to export and import track layouts.
5. The Track Model shall consider block size.
   - 5.1. Blocks shall be shown and configurable.
6. The Track Model shall signals and switch machines.
7. The Track Model shall implement track circuits for presence detection.
8. The Track Model shall consider railway crossings.
9. The Track Model shall include stations.
   - 9.1. Passengers shall be loaded and unloaded at stations.
10. The Track Model shall implement the following failure modes:
    - 10.1. Broken rail
    - 10.2. Track Circuit failure
    - 10.3. Extra or no trains detected
    - 10.4. Power failure
    - 10.5. No communication going to train

3.4 Train Model

3.4.1 Description

The Train Model models the movement of the train using Newton's laws of motion assuming that the train is a rigid body point mass. This is a non-vital module.

3.4.2 Interface

The Train Model shall take in speed and authority from Track Model. It shall output the speed and authority to the Train Controller.

3.4.3 Functional Requirements

1. Train Specifications
   1.1. Length
   1.2. Height
   1.3. Width
   1.4. Mass
   1.5. Passenger Count
   1.6. Crew Count
2. Physics/Inputs
   2.1. Authority
   2.2. Speed Limit
   2.3. Acceleration/Deceleration Limit
   2.4. Door open/close
   2.5. Lights on/off
   2.6. Temperature Control
   2.7. Emergency Brake
   2.8. Brake Command
   2.9. Track Circuit
3. Failure Modes
   3.1. Brake Failure
   3.2. Signal Pickup Failure
   3.3. Train Engine Failure

3.5 Train Controller

3.5.1 Description

This module is responsible for safely dictating the train's speed by way of controlling its power, as well as controlling some smaller tasks like light and door control, and feedback about current and approaching stations. This module is VITAL.

This vital controller receives a suggested speed and compares it to its own calculated maximum safe speed and selects the lower of the two values to relay to the Train Model.

As the train's speed is controlled by its current power, the Train Controller converts the new speed into a power requirement (or a brake setting, if needed). The train controller also dictates when the lights and doors should be activated, and announces when the train is approaching a station or has stopped at a station.

3.5.2 Interface

The Train Controller receives a suggested speed from the Wayside Controller. This information shall be processed and relayed to the Train Model as a power setting. The controller sends signals to turn lights on or off, and open doors on either side of the train. This module receives feedback from the Train Model in the from of its current speed and/or power. The Train Controller also knows the train's current authority in the form of distance in miles.

3.5.3 Functional Requirements

1. The Train Controller shall have an algorithm capable of calculating a maximum safe speed for the train in question.
   - 1.1. The maximum safe speed shall be the maximum speed that can be fully arrested in the minimum safe breaking distance.
     - 1.1.1. The minimum safe braking distance shall take the train's authority into account during its calculation.
     - 1.1.2. The details of this calculation is TBD.
   - 1.2. The train controller shall consider the train's mass, current speed, and current authority when calculating this speed.
2. The Train Controller shall select a safe speed for the train.
   - 2.1. The Train Controller shall choose the suggested speed or its own calculated maximum safe speed.
   - 2.2. By default, the lower of the two speeds shall be chosen for the train.
   - 2.3. The Train Controller shall allow a manual mode for the driver of the train to manually input a speed. This speed shall not exceed the speed suggested by the Wayside Controller.
3. The module, once it has chosen a speed for the train, shall require an algorithm to convert that desired speed into either (1) a power setting for the train, or (2) an application of the brake for a certain amount of time.
   - 3.1. This algorithm shall use the train's current and desired kinetic energy combined with a time constraint to determine the power required to achieve the desired speed.
   - 3.2. This algorithm shall also take grade into account.
   - 3.3. The details of this algorithm are TBD.

4. The Train Controller shall dictate that lights should be turned on or off.
   - 4.1. The Train Controller shall control lights based on a daily schedule.
   - 4.2. The Train Controller may manually control these lights regardless of schedule, if desired.
   - 4.3. It is the responsibility of the Train Model to execute these instructions.
5. The Train Controller shall dictate that train doors should open.
   - 5.1. The Train Controller shall only open doors when the train is stopped.
   - 5.2. Doors can be opened on the left or the right, or both.
   - 5.3. The Train Controller shall read which doors should be open from beacons approaching each station.
   - 5.4. In the event of an emergency stop and train evacuation, the Train Controller shall send the command to open whichever doors are appropriate, given the situation at hand.
   - 5.5. It is the responsibility of the Train Model to execute these instructions.
6. The Train Controller shall display stops and stations.
   - 6.1. The upcoming stop shall be displayed no more than one block before the station or stop's location.
   - 6.2. The Train Controller shall display the information until the train has physically left that station.
   - 6.3. If there is no relevant station or stop to display, the Train Controller shall display nothing by default.

3.6 Nonfunctional Requirements

3.6.1 Performance Requirements

The simulation software shall be able to simulate at 10 times wall clock speed. This will allow quicker evaluation of a simulated circumstances. The simulation speed should be modifiable during simulation; i.e. it specified while the simulation is running, rather than before the software starts.

3.6.2 Safety Requirements

No train shall at any point collide with any other train. Trains shall not share blocks. No train shall exceed the speed limit or authority.

3.7 Design and Implementation Constraints

1. The Wayside Controller is a PLC, meaning that it can only operate on simple binary values. Any PLC code written for it shall observe this constraint.