**Design Description**

for

**Train Controller (NSECS-TNC)**

**Prepared by**

BreathlessBovine: Ben Tomasulo

**Organization**

ECE/COE 1186: Software Engineering

**Date**

3/28/2013

**Table of Contents**

1. Introduction

1.1. Purpose

1.2. Scope

2. System Overview

2.1. System Architecture

3. Design Description

4. UML Diagrams

4.1. Use case Diagram

4.2. Class Diagram

4.3. Sequence Diagram

5. Human Interface Design

5.1 Overview of GUI

5.2 Screenshots of GUI

**1. Introduction**

The Train Controller (TNC) system is responsible for simulating the PLCs that control the systems of each train by taking the information provided to it from the Train Model (TNM) and using that information to keep the train running in a safe and efficient manner.

**1.1. Purpose**

This design document shall explain the system architecture for the Train Controller sub system within the North Shore Extension Control System (NSECS). This document’s intended readership is for all stakeholders and it shall detail how the train controller will interface with other modules within the NSECS.

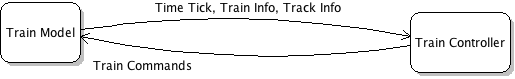
**1.2. Scope**

The produced software product shall be the Train Controller Module within the NSECS. The software shall interface with the TNM.

**2. System Overview**

The TNC uses the information given to it from the TNM to know when to activate and deactivate the various systems onboard each train.

**2.1. System Architecture**

****

**3. Design Description**

The TNC consists of two classes, the TrainControllerUI class and the TrainPLC.

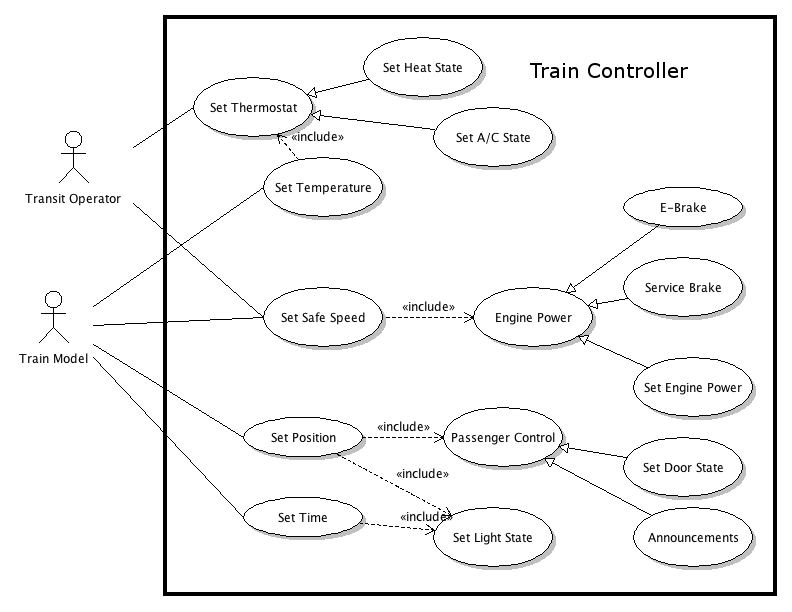
The TrainControllerUI interfaces with the TNM, contains the UI screen, and passes override information to the TrainPLC. There is only one TrainControllerUI, and as many TrainPLCs as there are trains. The override inputs and toggles can be used either to override information from the TNM, or provide substitute information in the event the TNC cannot communicate with the TNM. For testing purposes in the event the TNC cannot communicate with the TNM, the Tick Override Toggle should be engaged and only one TrainPLC will be created. If the TNC and TNM are interfacing correctly, the operator should only need to worry about entering the speed setpoint. They can also change the desired temperature (which defaults to 22°C on PLC startup), and activate the E-Brake in case of emergency. It will also contain 3 fake track blocks for testing purposes in case the TNM fails to interface with the TNC, one above ground with a 19m/s speed limit, one below ground with a speed limit of 17m/s before a station, and one above ground station block with a speed limit of 15m/s.

The TrainPLC uses the information given to it from the TNM to calculate how much engine power to provide, when to apply the service brake, when to apply the E-Brake, when to activate and deactivate the lights, A/C, and heating, when to open and close the doors, and when to announce what stop is coming up. This information is passed back to the TNM. One TrainPLC object is created every time the TNM adds another train, and one removed every time the TNM removes a train.

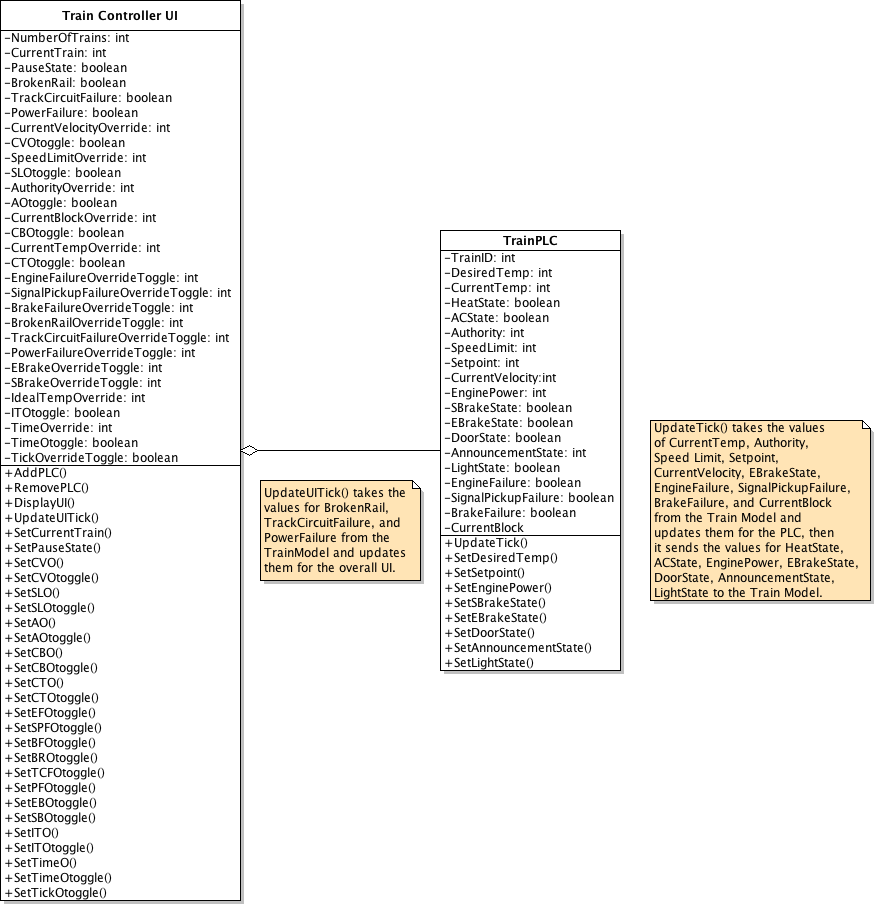
**4. UML Diagrams**

The following diagrams shall detail the structure, functionality, and behavior of the TNC.

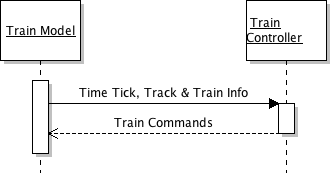
**4.1. Use Case Diagram**

****

**4.2. Class Diagram**



**4.3. Sequence Diagram**

****

**5. Human Interface Design**

**5.1. Overview of GUI**

The UI for the TNC is primarily used for testing and maintenance purposes. In real time use, interface should not be utilized. In the event an operator needs to use the TNC GUI, the proper train should be selected, and the “Pause/Resume” toggle and all controls labeled with “Override” (with the exception of the “E-Brake Override Toggle” in case of emergencies) should be ignored.

**5.2. Mockup of GUI**

