**SHWOZ**

****

Railway Train System Simulation

System Design

Hongyao Shi

March 27th 2014

**Table of Contents**

1 Introduction…………………………………………………………………………………. 2

2 User Case Diagram………………………………………………………………………… 2

3 System Architecture Patter…………………………………………………………………... 3

4 Dependency Discription ……………………………………………………………………….. 3

5 Detailed Design………………………………………………………………………………… 4

6 Sequence Diagram …………………………………………………………………………….. 5

1. **Introduction**

***1.1 Purpose and Scope***

The purpose of this document is to describe and write in detail for the system design for the Train Controller of the Centralized Train Control Suite.

This document contains the system requirements, system and subsystem architecture, detailed design, processing logic and external interfaces for the Track Controller.

***1.2 Project References***

1.2.1 IEEE-1016 Software Design Description.

***1.3 Abbreviation***

Authority – the distance the train is allowed to travel

GUI – Graphical User Interface

MVC – Model View Control Architecture Pattern

1. **Use Case Diagram**

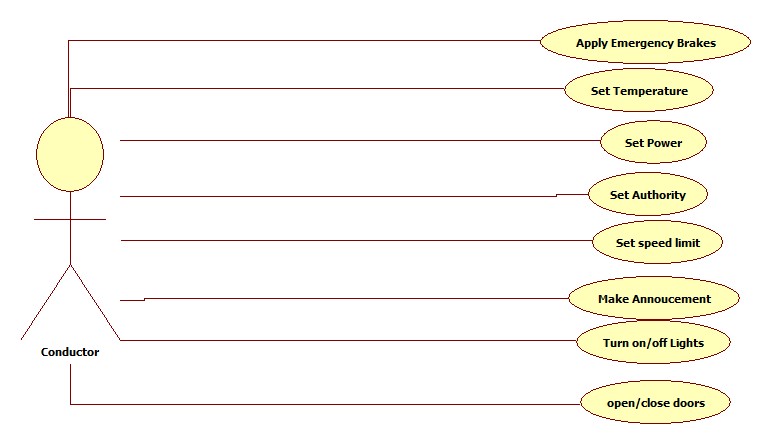


Figure 1: User Case Diagram for the Train Controller Subsystem

The use case diagram is shown in Figure 1, there is only 1 user for the train controller subsystem which is the conductor. The conductor shall be able to set temperature, apply emergency brake and normal brake. Toggle both light and door states. The conductor shall also be make announcement, set authority for the train, and most importantly, conductor shall be able to set speed and output as power to the train model.

1. **System Architecture Pattern**

MVC is chosen as the system architecture pattern, in which Train Model served as a Model section, Train controller functioned as a Control section and the Train Controller GUI acted as a View section. The architecture pattern is shown in Figure 2.

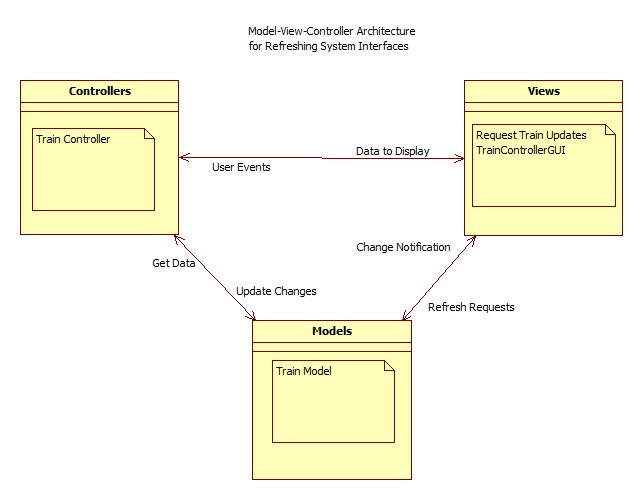


Figure 2: Train Controller System Architecture MVC

System users shall be able to communicate with the system through the TrainController Module. User issues a command by clicking the button or input values on the TrainControllerGUI. The TrainControllerGUI modules then notifies the TrainController module. TrainController module decodes the command and then update the state of the TrainModel module. Finally, TrainModel module notifies the TrainController to update the GUI.

1. **Dependency Description**

The relationships between each module inside the system are shown in figure 3. TrainControl Module communicates with the TrainModel, it sends output power and other information to TrainModel and receive responded parameters. The TrainController has a SafeControl sub-class to make sure that all the input parameters are valid.

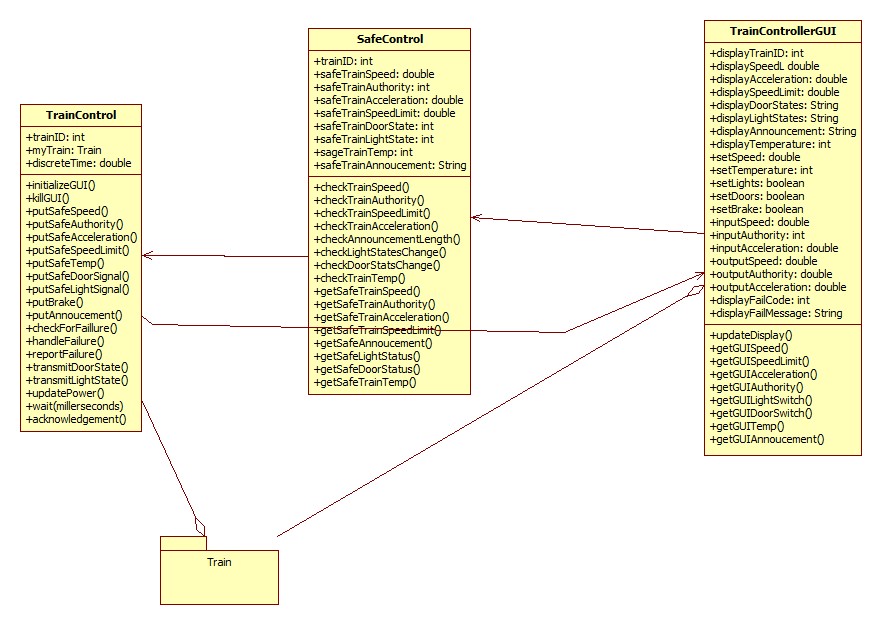


Figure 3 Class Diagram for the Train Controller

1. **Detailed Design**

**5.1 TrainControllerGUI**

TrainControllerGUi is responsible for showing the current attributes for the train model such as current speed, current acceleration, light/door status etc. Also, it is the input interface for the user to input signals or attributes to manually change the state for the train model. The attributes will be updated by using the updateDisplay() method.

**5.2 SafeControl**

This class is a subclass of the TrainControl Class, this will go through all the possible parameters passed to the TrainControl class, and it will check for the validity of the parameters. For example, if the set speed is over the physical speed limit. It will process the parameter and change the parameter to a valid value if they are not. It will also calculate the power for the train model using the acceleration and velocity.

**5.3 TrainControl**

This class will store the safe data from the SafeControl Class and output the attrbutes to the train model, it will also update the display in the TrainControllerGUI display.

1. **Sequence Diagram**

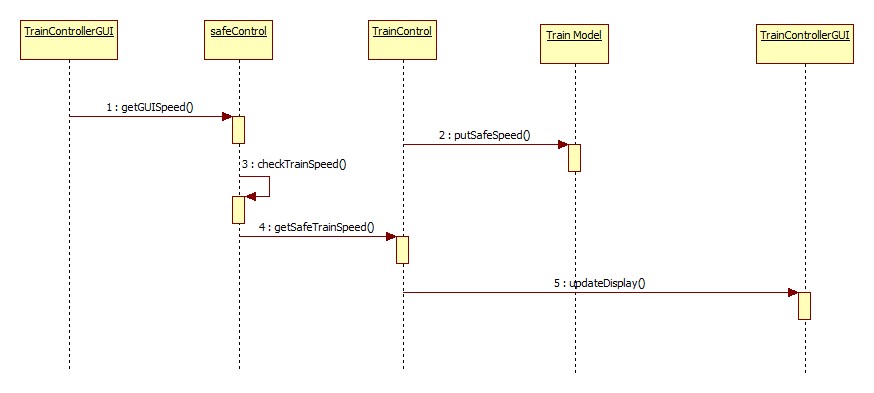


Figure 4: Sequence diagram to change speed

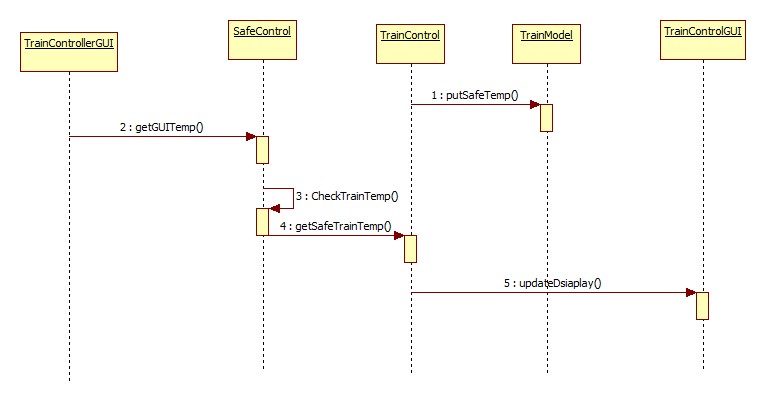


Figure 5: Sequence diagram to change temperature.