Software Design Description

Thomas Bui, Justin Carter, Patrick Flaherty, Wesley Miller Philip Seitz

Contents

1 Introduction

1.1 Purpose

The purpose of this document is to outline the design views and decisions in the development of the Train Control Signaling System software, including all relevant diagrams and descriptions to describe the architecture, layout, and dependencies of the system and its modules.

1.2 Scope

The purpose of this project is to design and implement a new Centralized Traffic Control Center and Signaling System for Light Rail Transit system. The final system will be an executable that will demonstrate a fully operational control center, communications, train and track control system simulator for the transit system. This system would include a specification and implementation of a Centralized Traffic Control Center, a Track Controller, a Track Model, a Train Controller, and a Train Model. These modules would all be interface-able with a user. Above all, we hope to provide a comfortable user experience along with the best pricing available.

1.3 Definitions, Acronyms, Abbreviations

The following are a list of terms, each followed by synonyms and abbreviations. A definition of each of the terms then follows:

- Train Control Signaling System - TCSS
  - The Train Signaling System is the complete deliverable which this SRS shall specify.
- Centralized Traffic Control - CTC
  - The Central Office is one of the major modules of the TCSS. The CTC is able to track the locations of the trains in the railway, as well as communicate to these trains important and vital information.
- Operation Control Center - OCC
  - See *Centralized Traffic Control*
- Central Office
  - See *Centralized Traffic Control*
- Track Controller
  - This is a major module of the TCSS. The Track Controller is responsible for sending important information to both the CTC and the Train Controller (Through the Track and Train Models). It also

is responsible for controlling the state of the tracks, such as track switching and railway crossing. This runs a PLC and is configurable on a per-Track Controller basis.

- Train Controller
  - This is a vital *(See Vital)* component of the TCSS. It is responsible for regulating the speed of the train. It is also responsible for controlling various other features of the Train Model. It is directly interface-able by the train driver, and will be capable of being controlled in both automatic and manual modes.
- Track Model
  - This is a major component of the TCSS. It is a digital representation of the track with which the Train Model rides over, and the model which the Track Controller is responsible for. This is configurable via a formatted csv file that will allow the user to simulate and control over any track diagram.
- Train Model
  - This is a major component of the TCSS. It is a digital representation of the Train with which the Train Controller is responsible for. Furthermore it is a representation of the effect of Newtonian physics on the TCSS decision making.
- Port Authority of Allegheny County - PAAC
  - A public transit agency within Pennsylvania.
- Programmable Logic Controller - PLC
  - This is a logic controller that is specially programmable on a per-device basis. It is often adapted for the control of systems and devices that require a high reliability and ease of programming.
- Vital
  - Safety-Critical
- Java Runtime Environment - JRE
  - The JRE is a virtual machine that enables a host computer to execute programs written in Java.
- Office Open XML Workbook - XLSX
  - Office Open XML is a zipped, XML-based file format developed by Microsoft for representing spreadsheets, charts, presentations and word processing documents.

- Graphic User Interface - GUI
  - An electronic user interface displayed through a screen so that a user may interact with software.

1.4 References

- Blackpool Flexity Tram Specifications, Bombardier Inc., 2009. Contained in professor-supplied Project Information directory.

1.5 Overview

The rest of the document outlines the basic design of the TCSS. In section 2, the system is outlined as a whole, dealing mostly with how outside actors, the users, will interact with the system. Section 3 looks at each module individually, relaying how they interact with each other as well as users.

2 System Architecture

2.1 System Architectural Design

Diagram.png

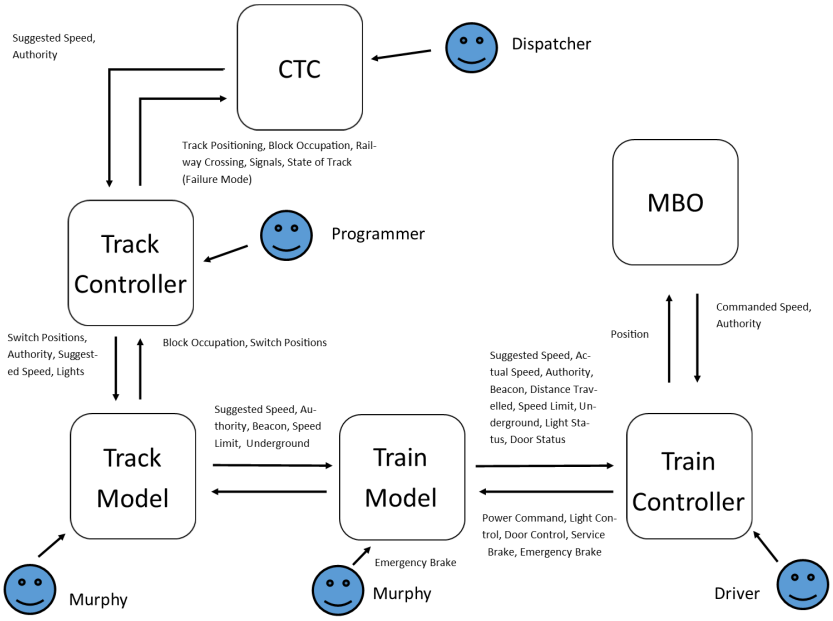

Figure 1: System Architecture

2.2 System Design Description

2.2.1 System Use Cases Diagram

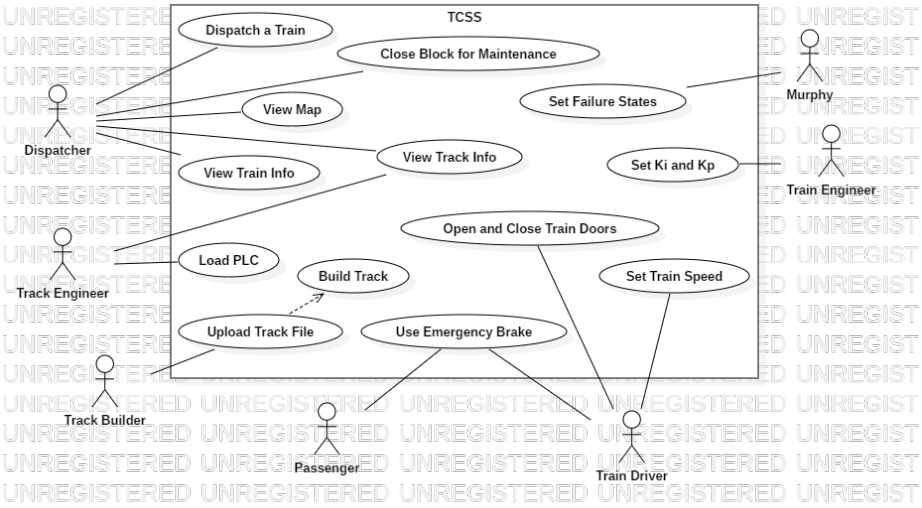

Figure 2: System Use Case Diagram

2.2.2 System Use Case Descriptions

System	**System**
Use Case	Dispatcher dispatches a new train
Actors	CTC, Track Controller, Track Model, Train Model, Train Controller
Description	• CTC receives new dispatch request • CTC computes and sends new suggested speed and authority to the Track Controller at the calculated time • Track Controller passes suggested speed and authority to designated block (start block) • Track Model creates a train and adds it to the start block • Train model sends speed and authority to Train Controller
Data	Input: New dispatch request Output: Train
Stimulus	Dispatcher wants to dispatch a new train

Response	Train is added to system at designated time and given suggested speed and authority

System	**System**
Use Case	Dispatcher closes a new block for maintenance
Actors	CTC, Track Controller, Track Model
Description	• CTC receives close block request • CTC sends suggested speed and authority to Track Controller at designated time • Track Controller sends suggested speed and authority to designated block • Track Model designates block as closed • CTC sends suggested speed and authority after designated time • Track Controller sends suggested speed and authority to designated block • Track Model designates block as open
Data	Input: New maintenance request Output: Display block changes
Stimulus	Track block needs maintenance
Response	Dispatcher enters maintenance request to the system

System	**System**
Use Case	Train Driver sets Train Speed
Actors	Train Driver, Train Controller, Track Model, Train Model

Description	• Train driver assigns a setpoint speed to the train controller in manual mode • The suggested speed and authority of the train are transmitted to the train controller through the track circuit • The train controller determines a safe operating speed by comparing the suggested speed, and setpoint speed, and considers both authority and system faults • The train controller transmits a power command to the train model • the speed of the train is adjusted according to the power command or brake command • The displays of all the appropriate displays are updated to reflect the new operating conditions
Data	Input: Set-point speed Output: Train Speed Changes
Stimulus	Driver or Murphy enter a new setpoint speed
Response	The train model adjusts in speed and relevant displays update to present this change

System	**Track Controller**
Use Case	Upload PLC
Actors	Track Engineer
Description	• Track Engineer shall upload PLC to Track Controller and set new values. • Track Controller shall pass values to the Track Model to update.
Data	Input: Actor Input Output: Switch, lights, and railroad crossing
Stimulus	Track Engineer input
Response	Track Controller values are updated as necessary

2.2.3 System Class Diagram

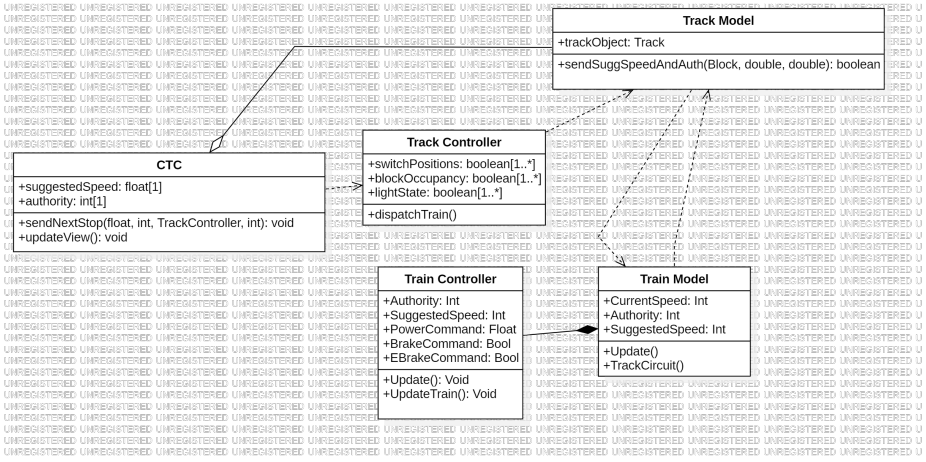

Figure 3: System Class Diagram

2.2.4 System Sequence Diagram

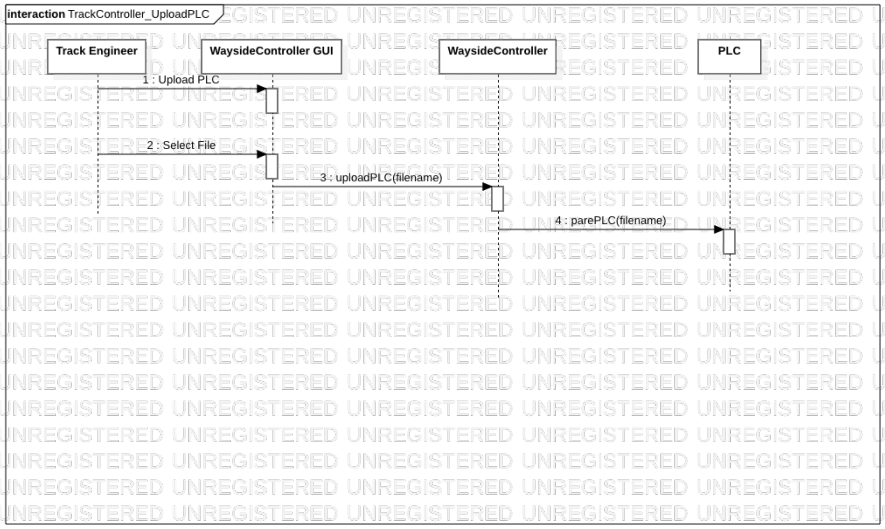

Figure 4: Upload PLC System Sequence

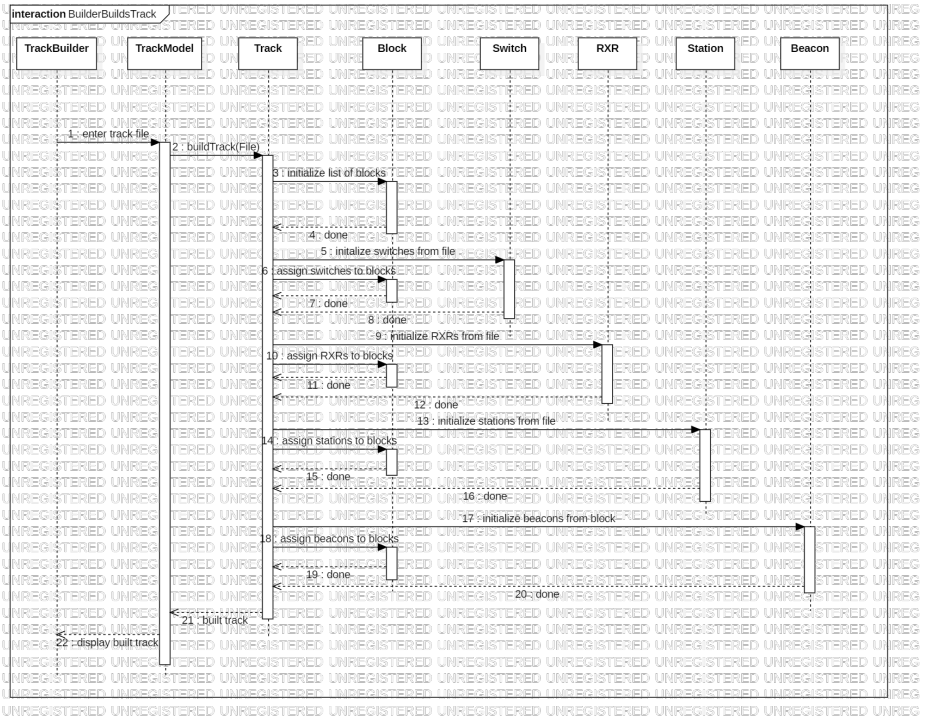

Figure 5: Use Case: Upload Track File System Sequence

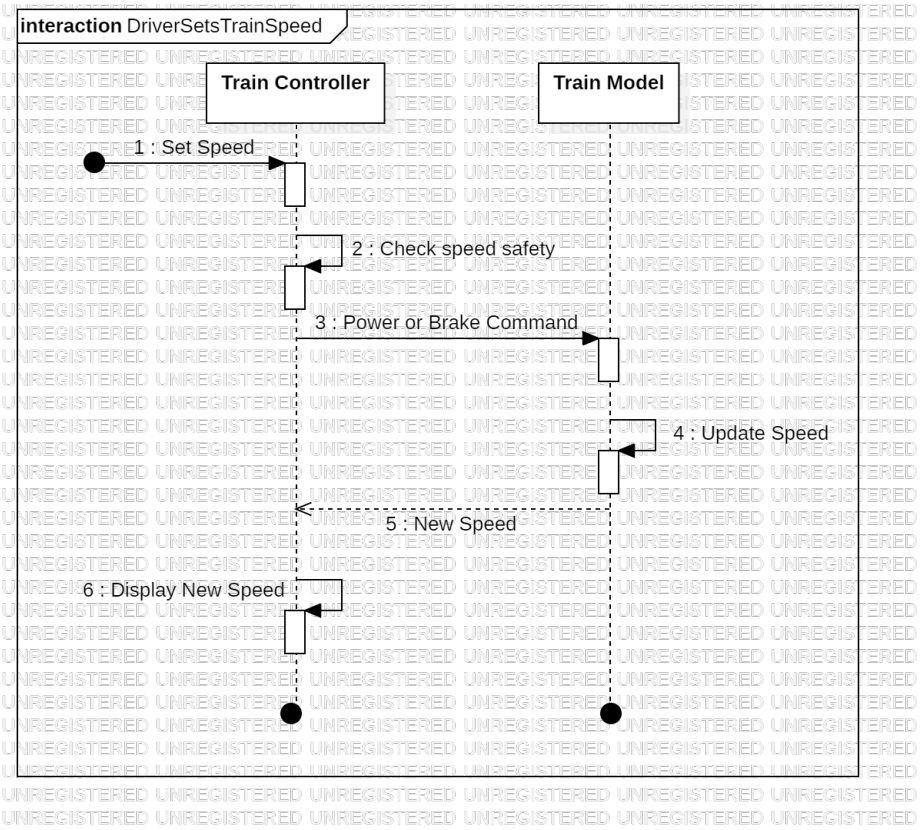

Figure 6: Driver Sets Train Speed

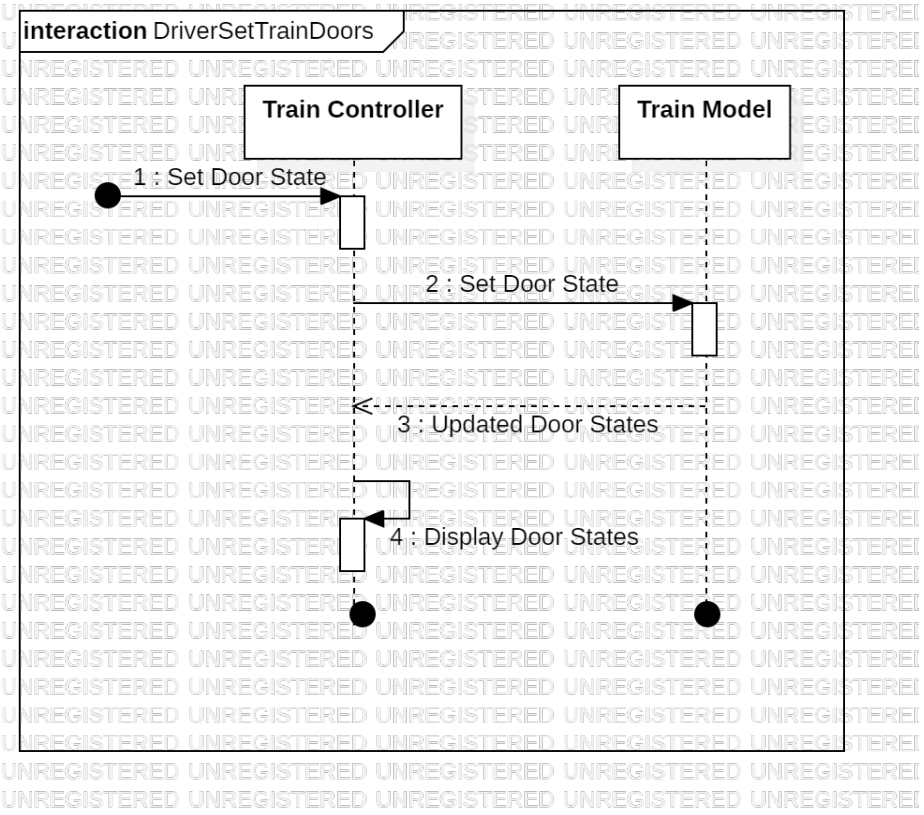

Figure 7: Driver Sets Doors

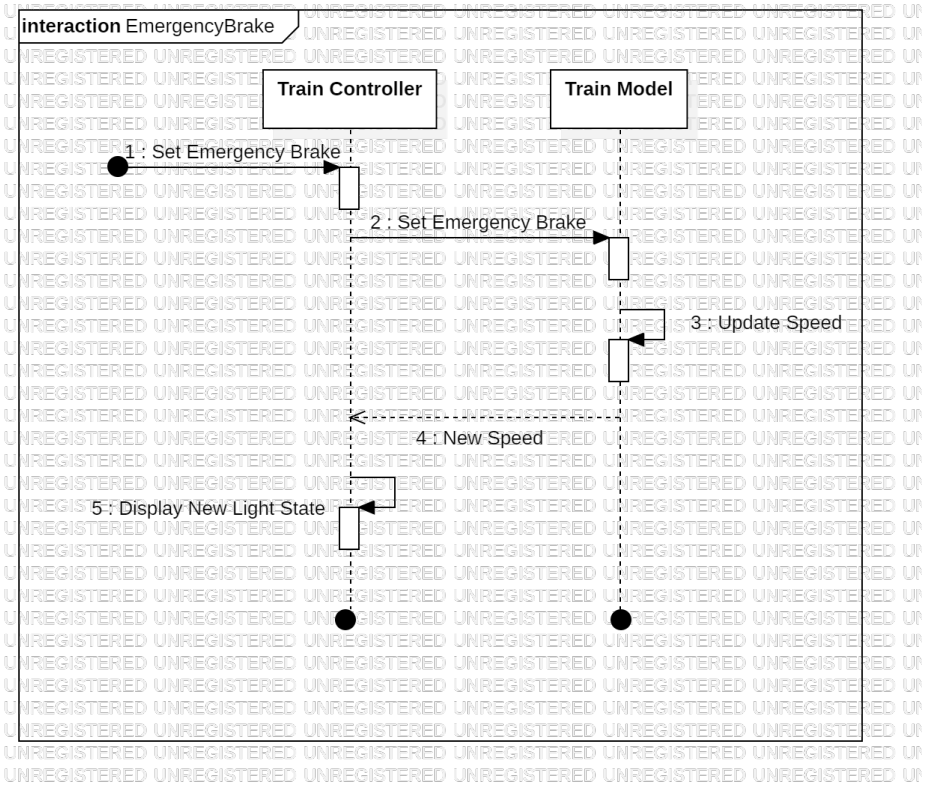

Figure 8: User Sets Emergency Brake

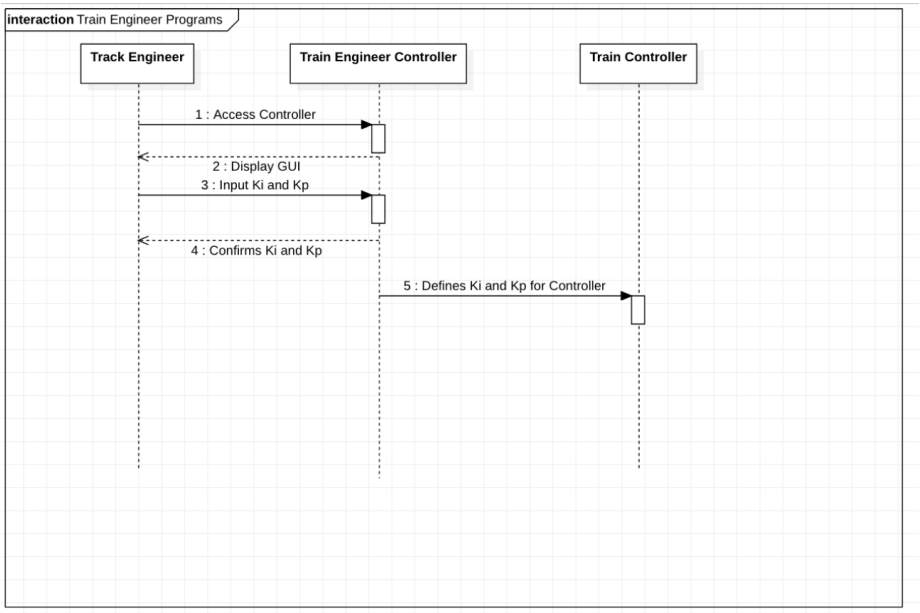

Figure 9: Train Engineer Programs Ki and Kp

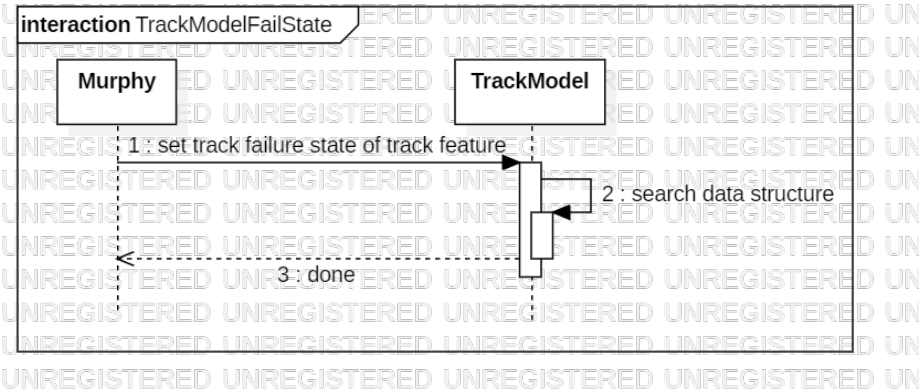

Figure 10: Murphy Causes Track Failure

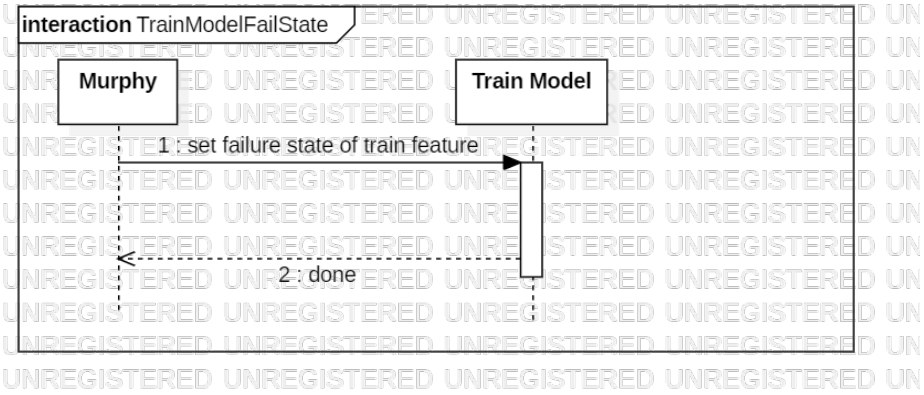

Figure 11: Murphy Causes Train Failure

2.2.5 System Deployment Diagram

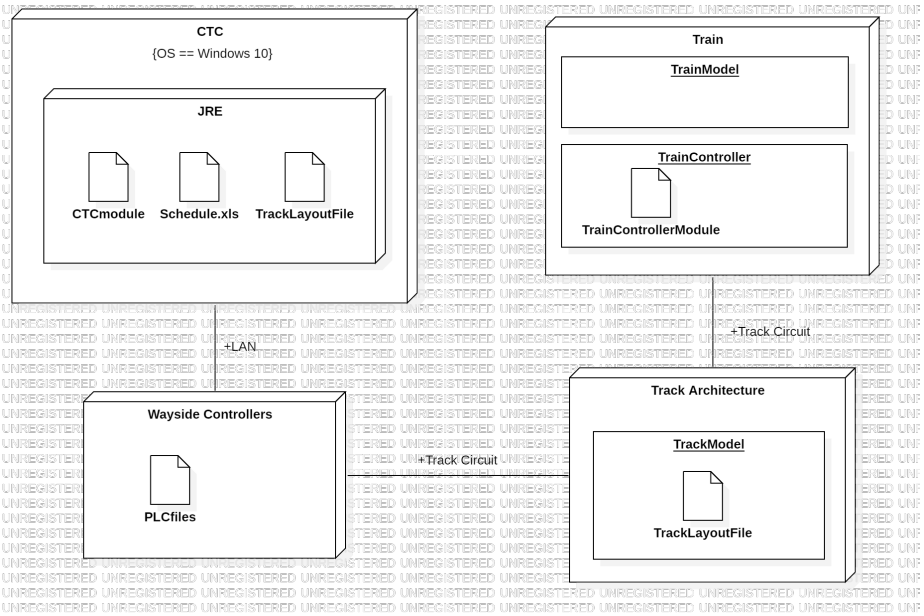

Figure 12: System Deployment Diagram

3 Module Design

3.1 Centralized Traffic Control

3.1.1 Use Case Diagram

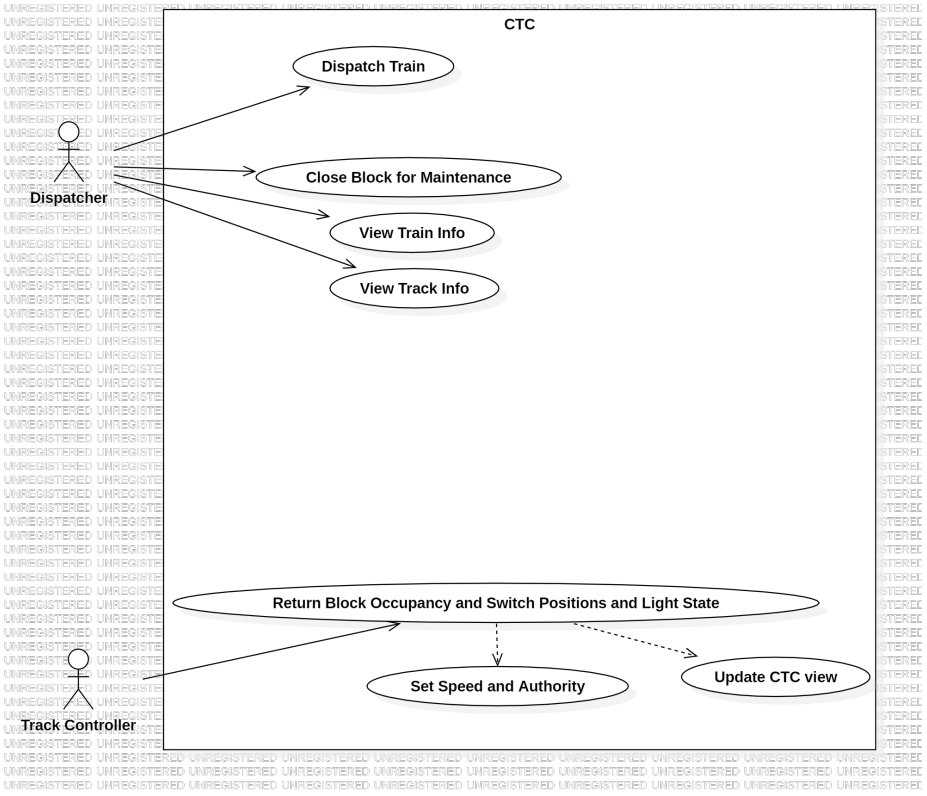

Figure 13: CTC Use Cases

3.1.2 Use Case Descriptions

System	**CTC**
Use Case	Dispatcher closes a block for maintenance
Actors	Dispatcher

Description	• Dispatcher selects a track section from the CTC view • Track Controller shall pass values to the Track Model to update. • Dispatcher selects a specific block from the section list • Dispatcher sets start time and duration for maintenance and creates request • Block is closed at time request is created • Maintenance period starts at request start time • Block is reopened after maintenance request is completed
Data	Input: Actor Input Output: Switch, lights, and railroad crossing
Stimulus	Dispatcher wants to close a block for maintenance
Response	The selected block is closed and re-opened when determined by the dispatcher

System	**CTC**
Use Case	Dispatch a train
Actors	Dispatcher
Description	• Dispatcher clicks on create new dispatch button • Dispatcher selects dispatch mode • Dispatcher selects schedule dependent on mode selected • Dispatcher confirms dispatch • New suggested speed and authority sent to the starting block at calculated start time of the dispatch
Data	Input: Schedule, Dispatch Output: Suggested Speed, Authority
Stimulus	New train needs to be dispatched in the system
Response	Dispatcher creates dispatch request and submits it to the system

System	**CTC**

Use Case	View train information
Actors	Dispatcher
Description	• Dispatcher clicks on an active dispatch from the CTC view • Dispatcher clicks on train name from the dispatch selected • Train information appears in the bottom of the screen
Data	Input: Train name Output: Formatted train information
Stimulus	Dispatcher wants to access current train information
Response	Formatted train information is displayed on the screen

System	**CTC**
Use Case	View Track Information
Actors	Dispatcher
Description	• Dispatcher selects a track section from the CTC view • Dispatcher selects a specific block from the section list • Block information is displayed onto the CTC view
Data	Input: blockID Output: Formatted block information
Stimulus	Dispatcher wants to access current status of a block
Response	Formatted block information is displayed on the screen

System	**CTC**
Use Case	Set Speed and Authority
Actors	Dispatcher, Track Controller

Description	• Track controller returns information to CTC (Block occupancy, switch positions, and light state) • CTC updates track model representation inside module to reflect current track conditions • CTC checks current train conditions for a need to send an updated speed and authority to a new block • CTC sends a new speed and authority to the correct block for the desired train
Data	Input: Block occupancy, switch positions, light state Output: suggested speed, authority
Stimulus	Global update to send information back to CTC to update information
Response	Dispatch list checked and new suggested speed and authorities sent as needed

System	**CTC**
Use Case	Update CTC view
Actors	Dispatcher, Track Controller
Description	• Track controller returns information to CTC (Block occupancy, switch positions, and light state) • CTC updates track model representation inside module to reflect current track conditions • CTC map view is updated accordingly
Data	Input: Block occupancy, switch positions, and light state Output: formatted track model view in GUI
Stimulus	Global update to send information back to CTC to update information
Response	Track model view in GUI is updated on the screen

3.1.3 Class Diagrams

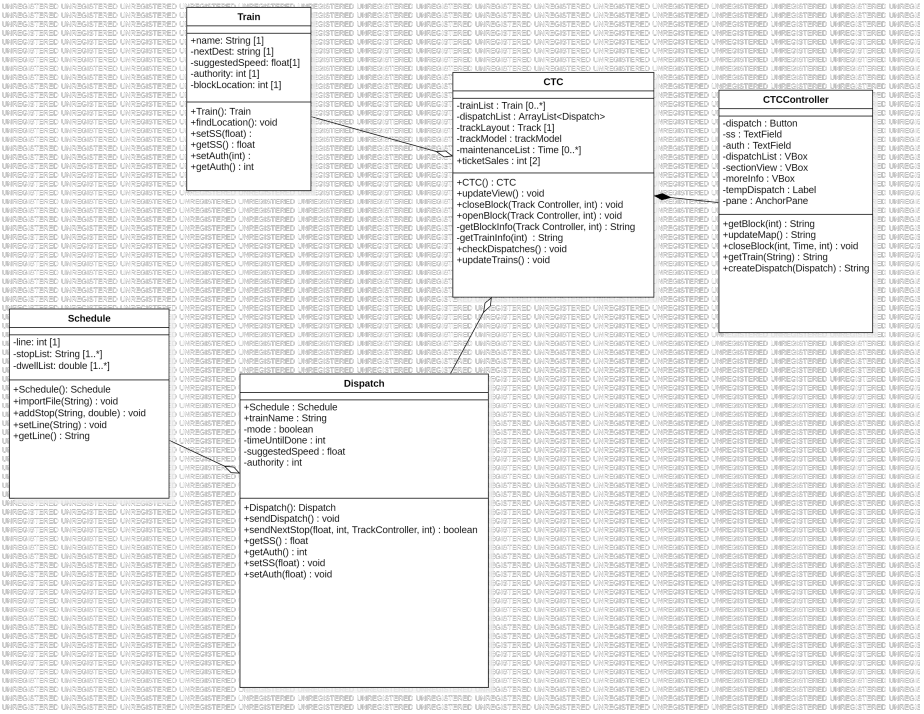

Figure 14: CTC Class Diagram

3.1.4 Sequence Diagrams

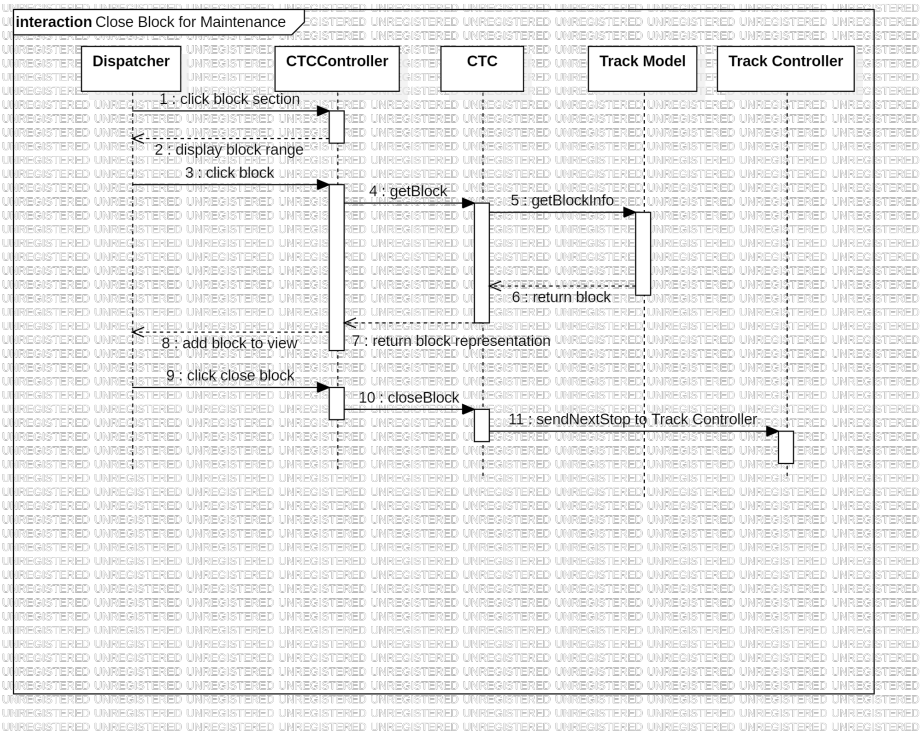

Figure 15: CTC Close Block for Maintenance

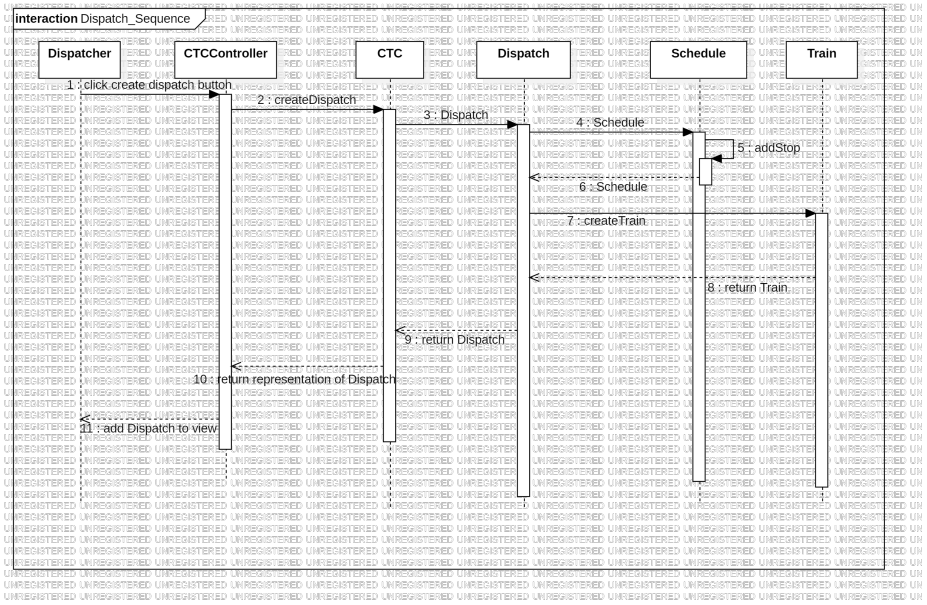

Figure 16: Dispatcher Dispatches a new train

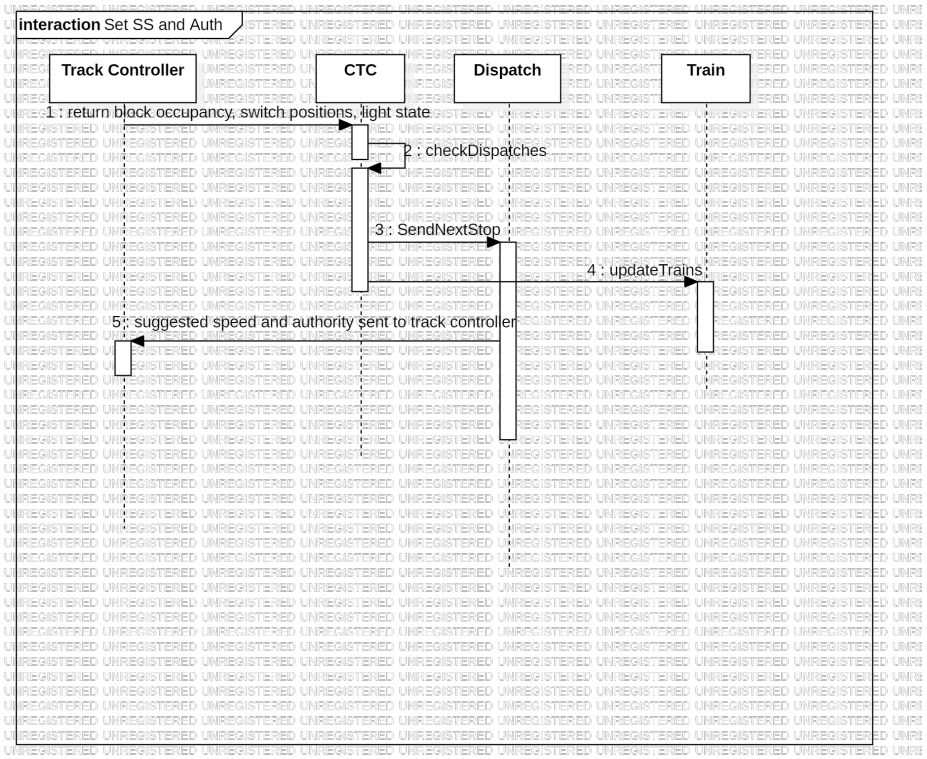

Figure 17: CTC Sets a New Suggested Speed and Authority

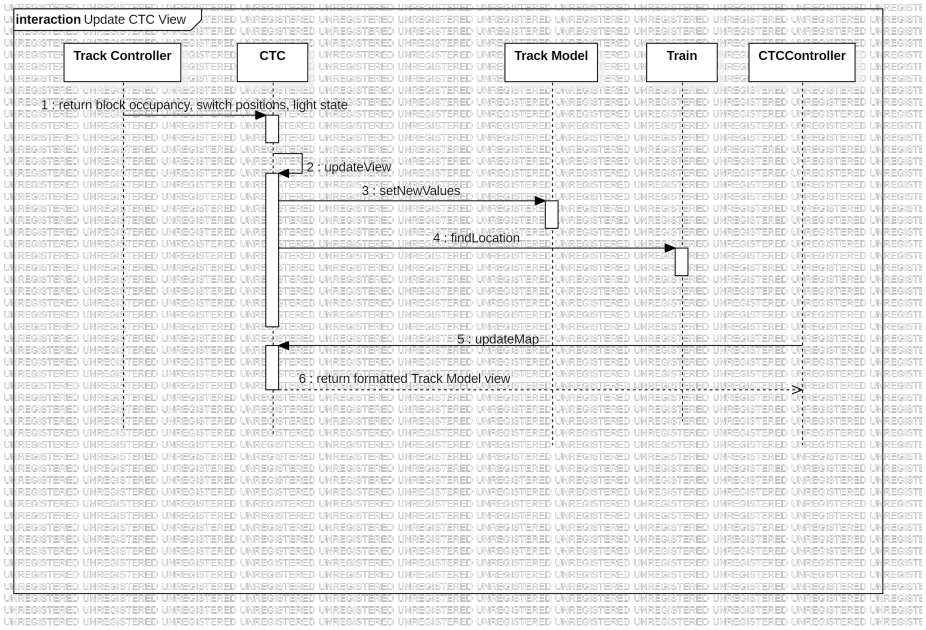

Figure 18: CTC view is updated

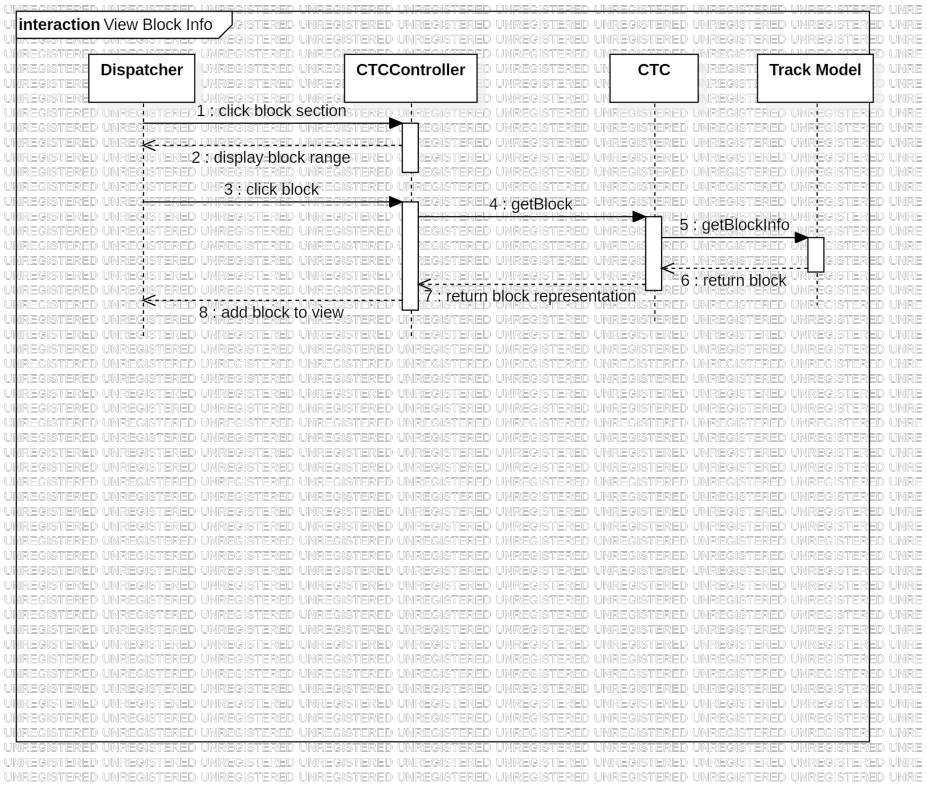

Figure 19: Dispatcher Views Information for a Block

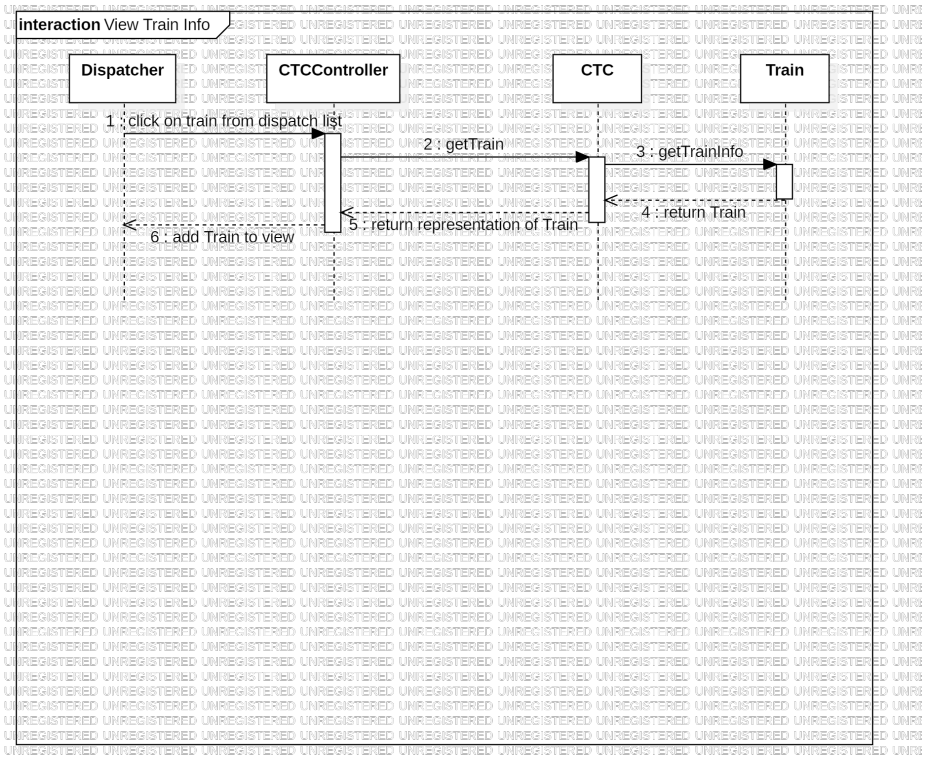

Figure 20: Dispatcher Views Information for a Train

3.2 Track Controller

3.2.1 Use Case Diagram

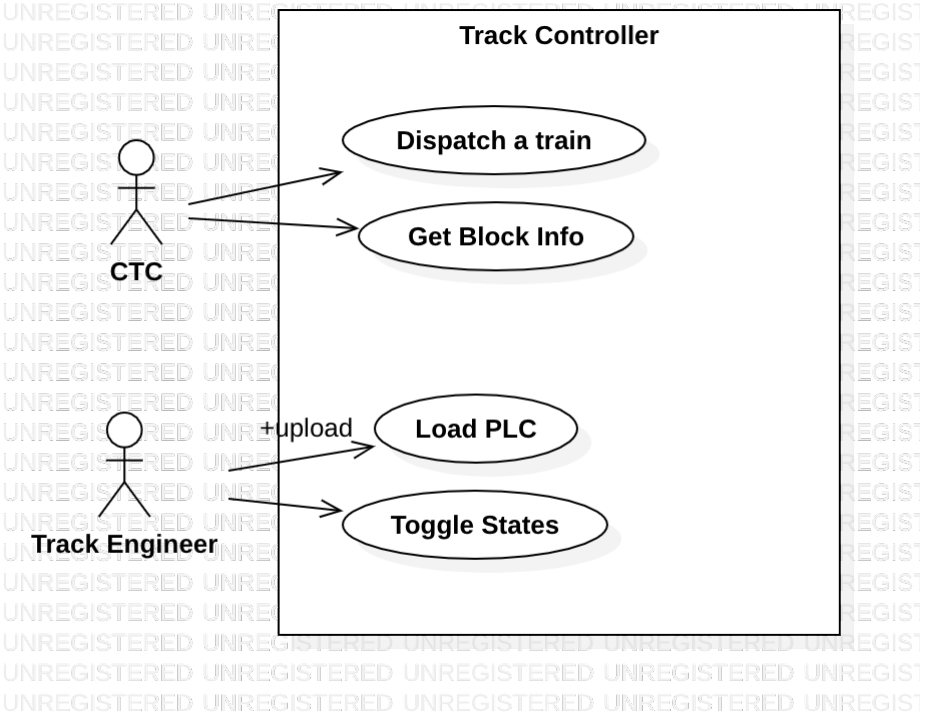

Figure 21: Track Controller Use Cases

3.2.2 Use Case Descriptions

System	**Track Controller**
Use Case	Upload PLC
Actors	Track Engineer
Description	• Track Engineer shall upload PLC to Track Controller and set new values. • Track Controller shall pass values to the Track Model to update.
Data	Input: Actor Input Output: Switch, lights, and railroad crossing
Stimulus	Track Engineer input

Response	Track Controller values are updated as necessary

System	**Track Controller**
Use Case	Toggle States
Actors	Track Engineer
Description	• Track Engineer toggles a switch, railroad crossing, or lights state when in maintenance mode. • Track Controller shall pass values to the Track Model to update.
Data	Input: Actor Input Output: Switch, lights, and railroad crossing
Stimulus	Track Engineer input
Response	Track Controller values are updated as necessary

System	**Track Controller**
Use Case	Select Block
Actors	Track Engineer
Description	• Track Engineer selects a Track Controller to view. • Track Controller selects a block to view information
Data	Input: Track Engineer Input Output: Block Info
Stimulus	Track Engineer input
Response	Track Controller block information will be shown

System	**Track Controller**
Use Case	Get suggested speed and authority
Actors	Track Controller, CTC
Description	• CTC sends an authority and suggested speed to the Track Controller • Track Controller shall pass this information to the track model to update

Data	Input: CTC Input Output: Suggested Speed, Authority
Stimulus	CTC input
Response	New suggested speed and authority shall be updated

System	**Track Controller**
Use Case	Update Maintenance Mode
Actors	Track Controller, CTC
Description	• CTC sends an authority of -1 to a block in the Track Controller • Track Controller shall pass this information to the track model to update • Track Controller shall close the block as necessary
Data	Input: Authority Output: Boolean for success or failure to close block
Stimulus	CTC input
Response	Block is either closed or open

3.2.3 Class Diagrams

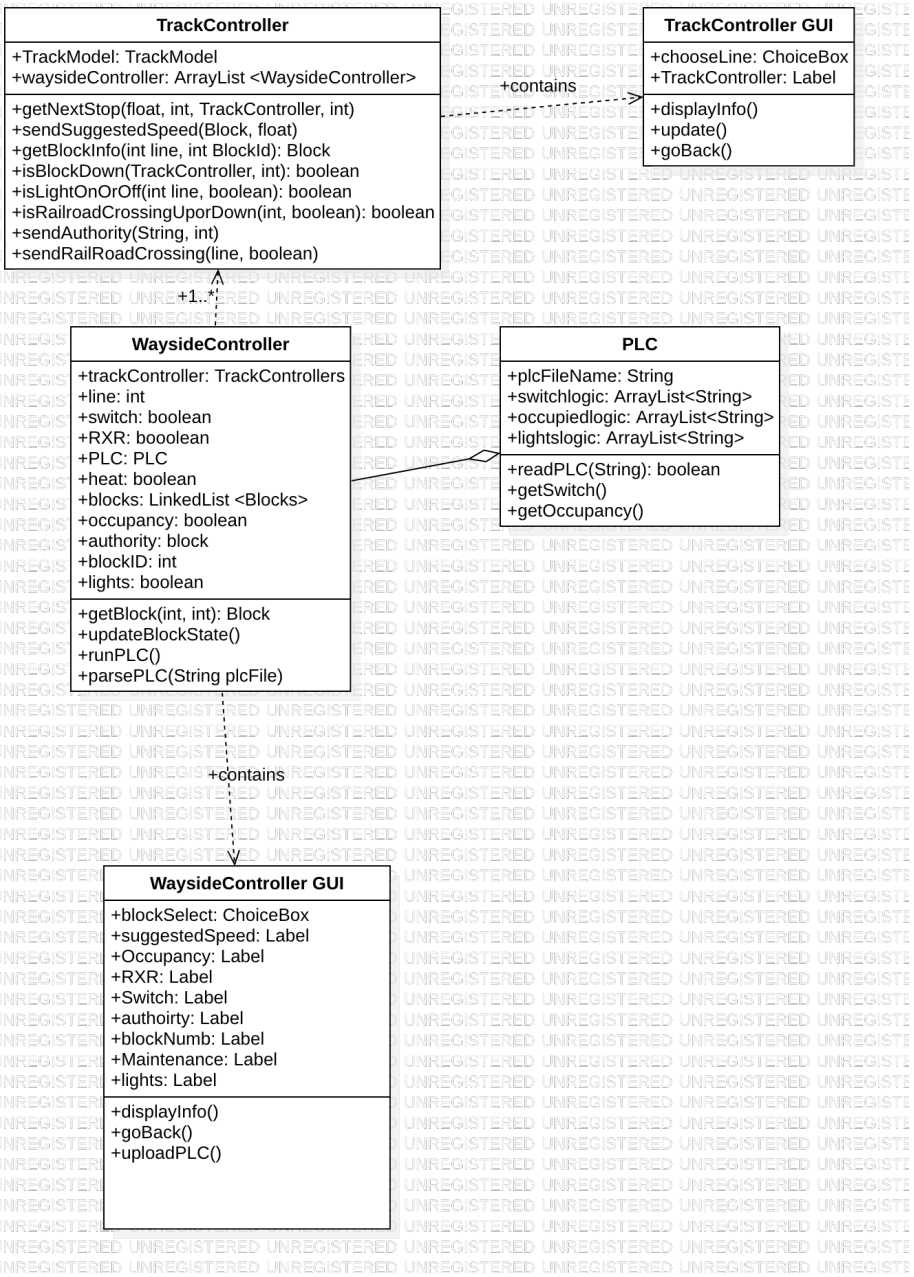

Figure 22: Use Case: Track Controller Class Diagram

3.2.4 Sequence Diagrams

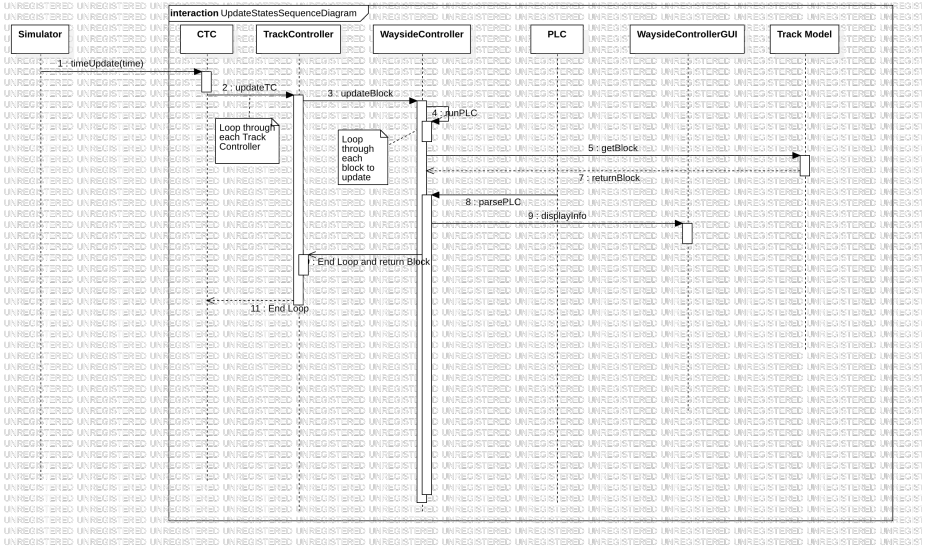

Figure 23: Use Case: Track Controller Update States

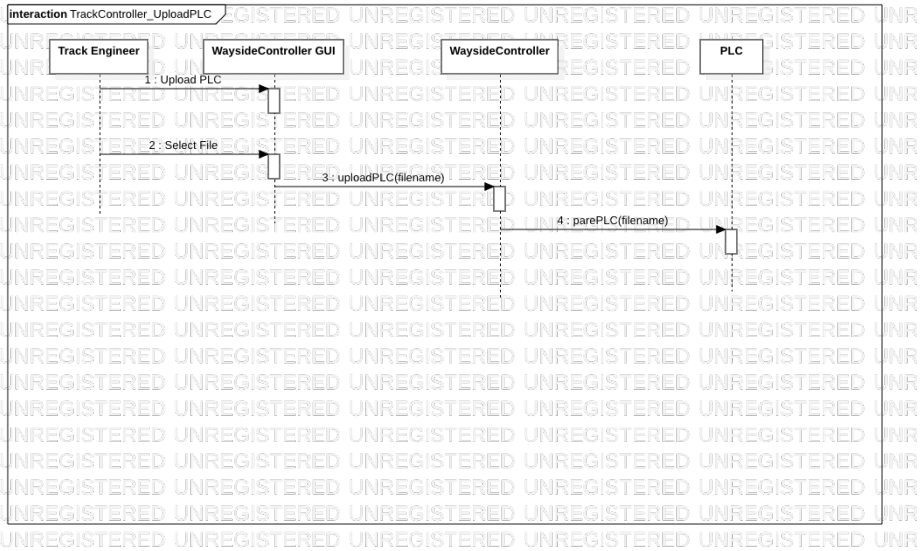

Figure 24: Use Case: Track Controller Update PLC

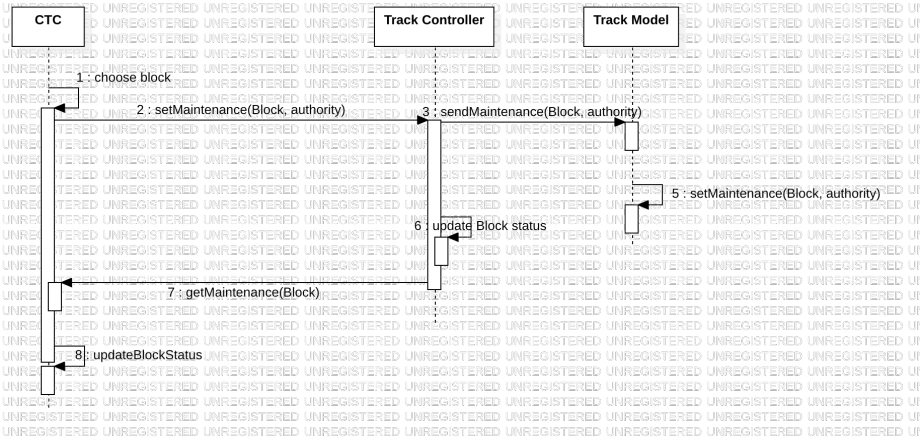

Figure 25: Use Case: Track Controller Update Maintenance

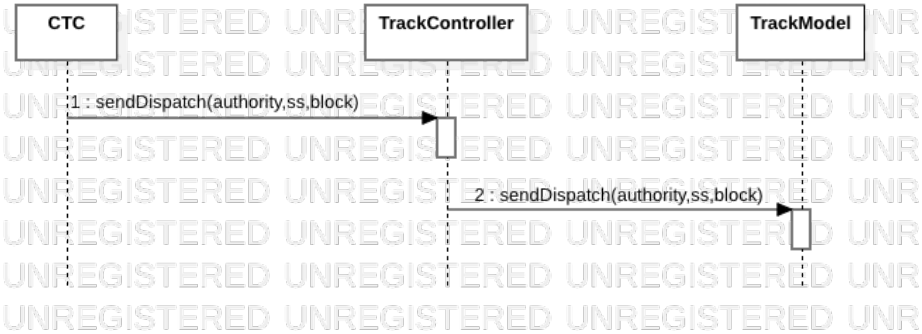

Figure 26: Use Case: Track Controller Set Authority and Suggested Speed

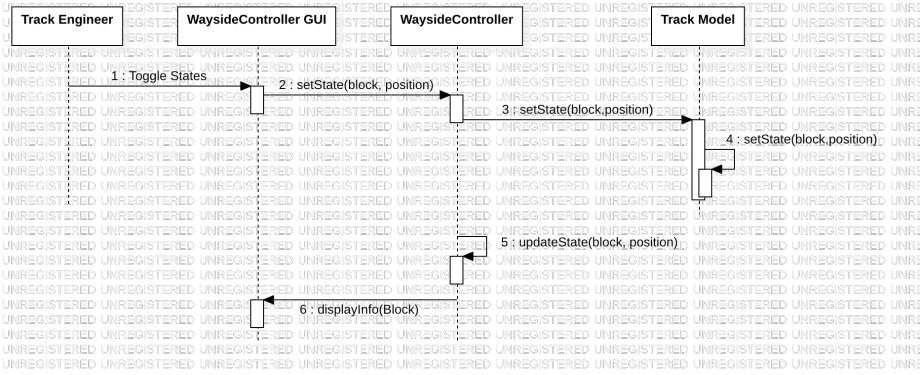

Figure 27: Use Case: Track Controller Toggle State

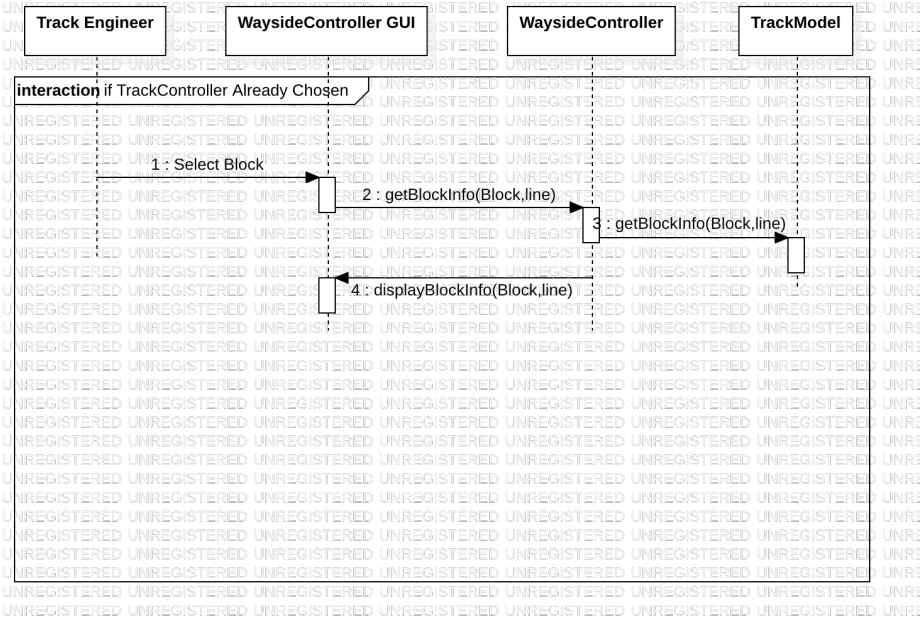

Figure 28: Track Controller Select Block

3.3 Track Model

3.3.1 Use Case Diagram

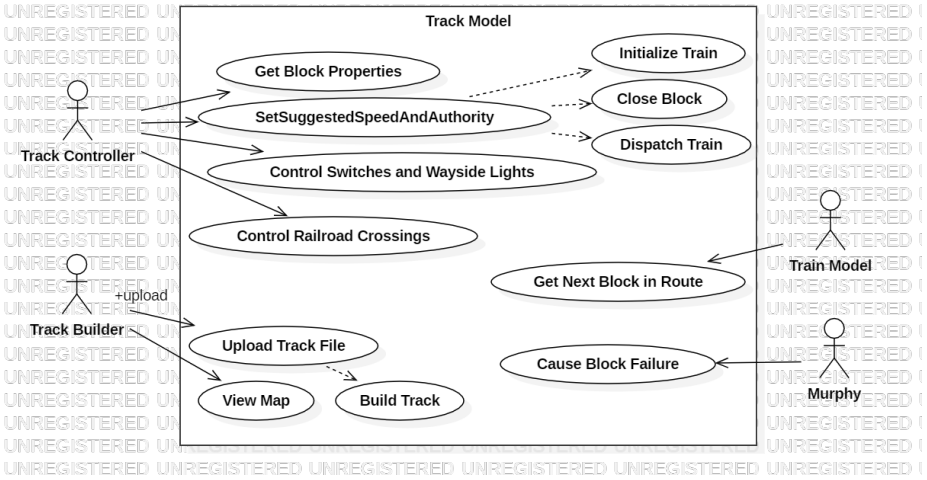

Figure 29: Track Model Use Cases

3.3.2 Use Case Descriptions

System	**Track Model**
Use Case	Build track from track file
Actors	Track Builder
Description	• Track Builder uploads track file • Track Model builds track by parsing file • Track Model performs verification on the built track • Track Builder confirms built track looks correct
Data	Input: Formatted track file Output: Track object
Stimulus	Module initialization
Response	Track object is constructed and ready to be used

System	**Track Model**
Use Case	Track Controller closes block for maintenance

Actors	Track Controller
Description	• Track Controller gives command to close block • Track Model searches for reference to desired block • Track Controller sends block non-functional suggested speed and authority (e.g -1) as an encoding to close block • Block state is set to closed
Data	Input: Suggested speed, authority, boolean value signifying opening/closing block Output: Boolean for success or failure to close block
Stimulus	Method call from Track Controller
Response	Block is either opened or closed for maintenance

System	**Track Model**
Use Case	Cause block failure
Actors	Train Controller
Description	• Murphy issues command to cause block failure • Track Model searches for reference to desired block • setFailure(int) called on block to create failure
Data	Input: Block number, failure type Output: Boolean signifying success
Stimulus	Method call from Murphy
Response	Block status is updated to show failure

System	**Track Model**
Use Case	Control RXR
Actors	Train Controller
Description	• Track Controller issues command to change state of railroad crossing • Track Model searches for reference to desired block • setDown(boolean) called on RXR of block to change state

Data	Input: Block number, commanded RXR state Output: n/a
Stimulus	Method call from Track Controller
Response	RXR status is updated

System	**Track Model**
Use Case	Control Switches and Wayside Lights
Actors	Train Controller
Description	• Track Controller issues command to change state of switch and corresponding wayside lights • Track Model searches for reference to desired block • setOrientation(boolean) and setLights(boolean) called on switch and lights of block to change state
Data	Input: Block number, commanded switch orientation, command light state state Output: n/a
Stimulus	Method call from Track Controller
Response	Switch and light statuses are updated

System	**Track Model**
Use Case	Train Model Requests Next Block in Path
Actors	Train Model
Description	• Train Model requests reference to next block on it's route • Track model returns the reference to the next block
Data	Input: Train's current block Output: Train's next block on path
Stimulus	Train detects polarity switch in track circuit (leaves current block)
Response	Train now has reference to new current block and can use its properties in it's force calculations

System	**Track Model**
Use Case	Track Controller Initializes Train
Actors	Train Controller

Description	• Track Controller gives command to initialize new train • Track Model searches for reference to desired block (should always be the start block for this command) • Track Controller sends block non-functional suggested speed and authority (e.g -1) as an encoding to initialize a new train
Data	Input: Suggested speed, authority to encode command to initialize train Output: Boolean signifying success
Stimulus	Method call from Track Controller
Response	Train Model instance is created and ready to be dispatched

System	**Track Model**
Use Case	Track Controller Dispatches Train
Actors	Train Controller
Description	• Track Controller gives command to dispatch train • Track Model searches for reference to desired block • Track Controller sends block a functional (non-negative) suggested speed and authority to be passed into the train occupying the block
Data	Input: Block number, suggested speed, authority Output: Boolean signifying success
Stimulus	Method call from Track Controller
Response	Train Model occupying block updated with new suggested speed and authority

3.3.3 Class Diagram

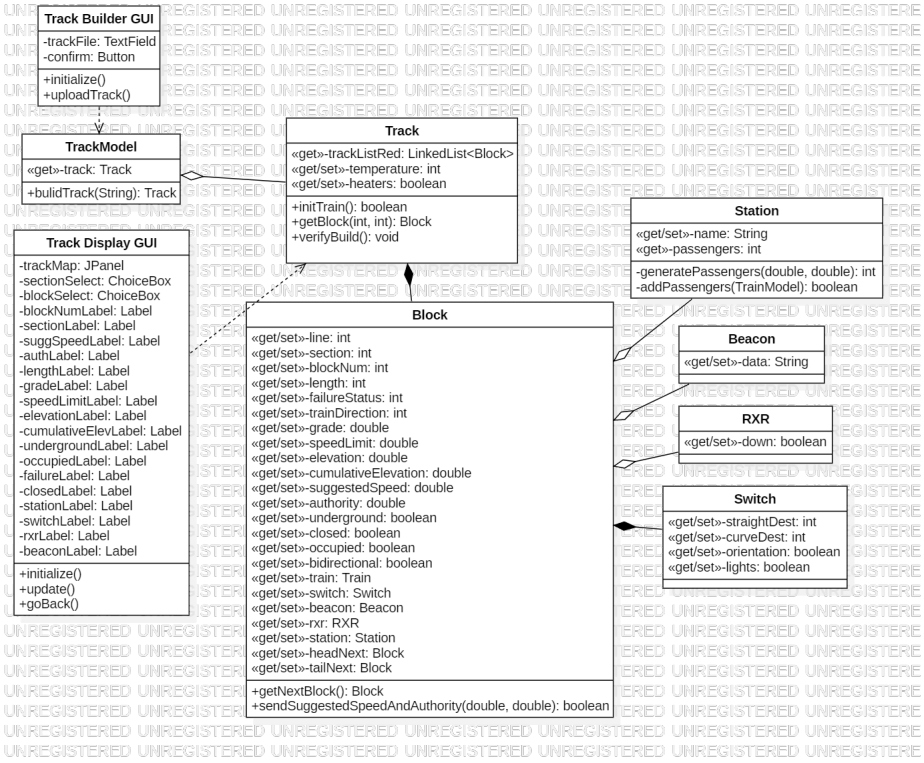

Figure 30: Track Model Class Diagram

3.3.4 Sequence Diagrams

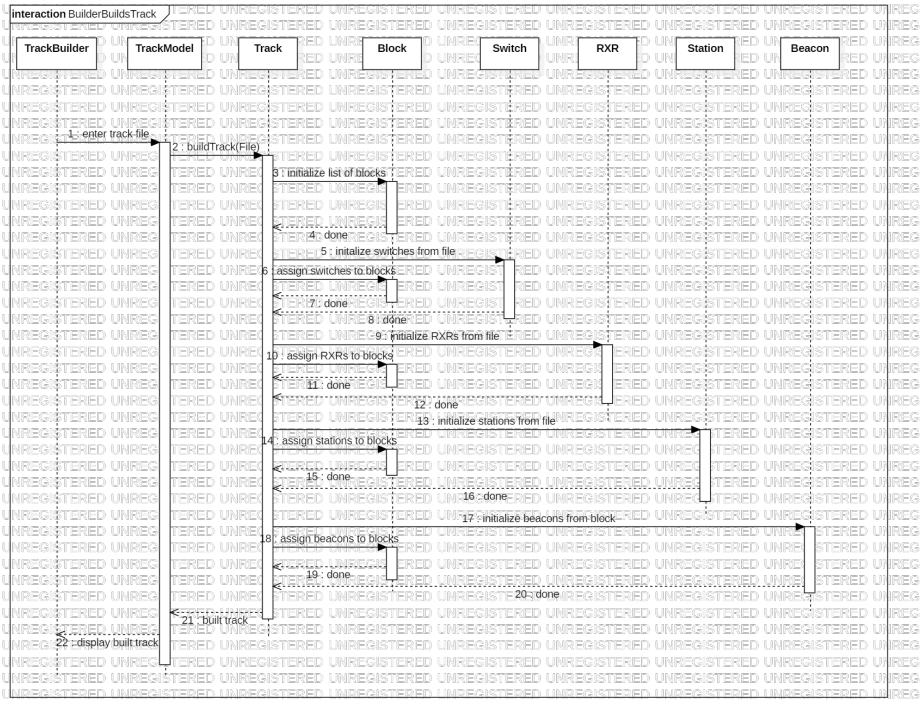

Figure 31: Use Case: Track Builder builds track using track layout file

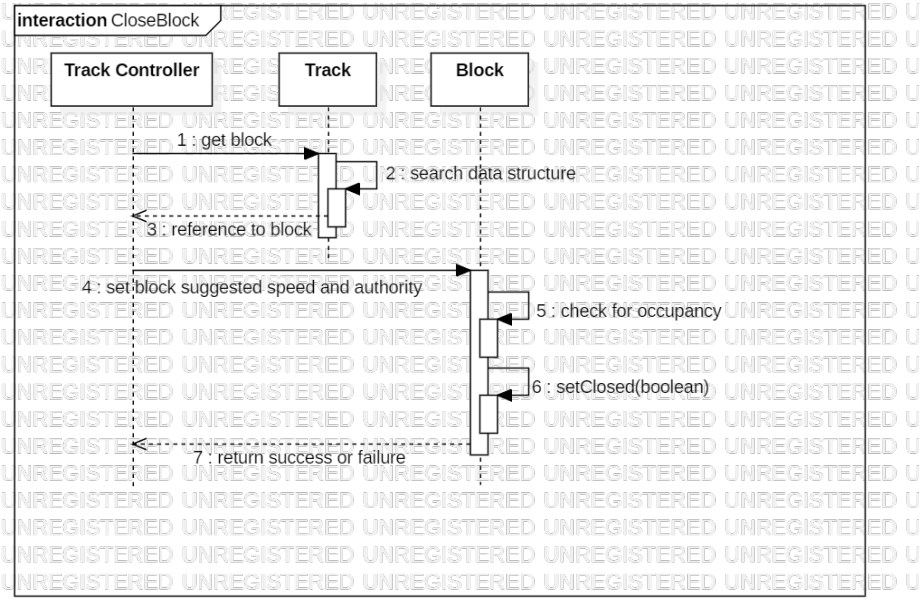

Figure 32: Use Case: Track Controller closes block for maintenance

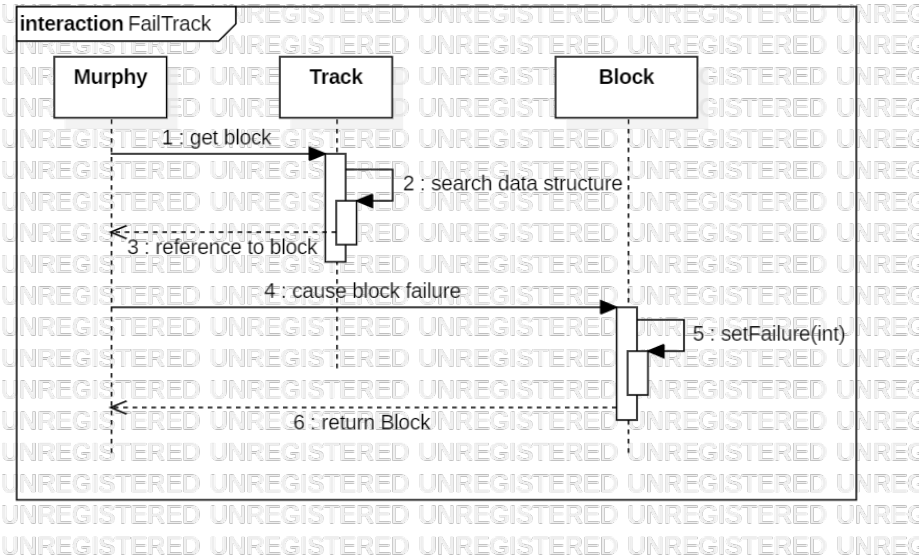

Figure 33: Use Case: Murphy Causes Block to Fail

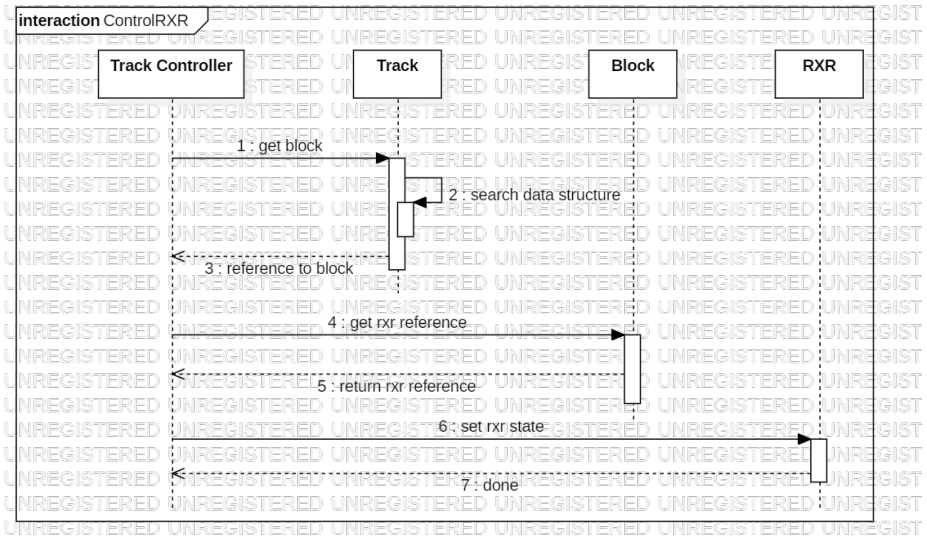

Figure 34: Use Case: Track Controller Controls Railroad Crossing State

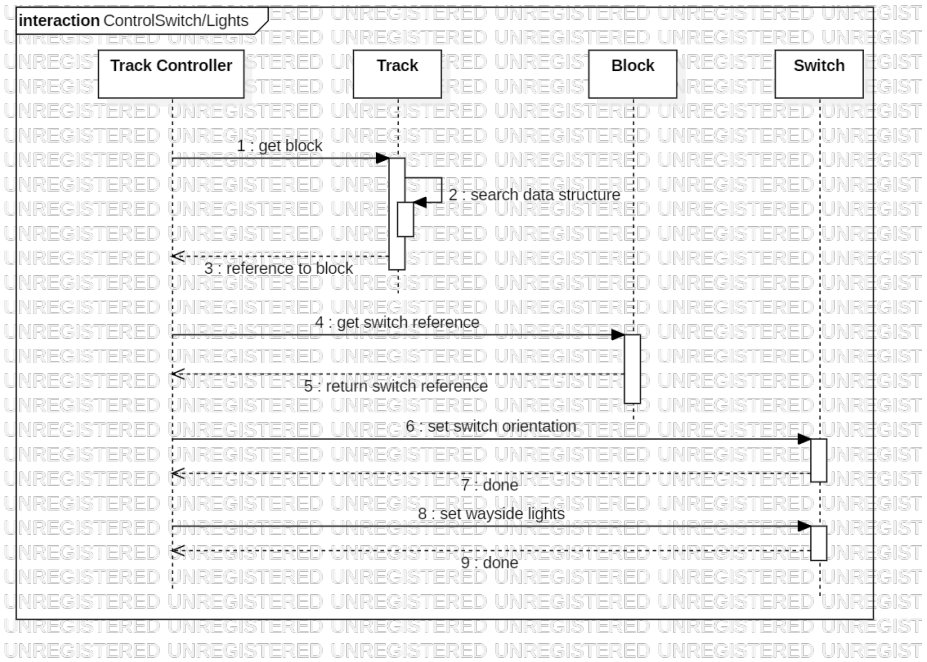

Figure 35: Use Case: Track Controller Controls State of Switch and Wayside Lights

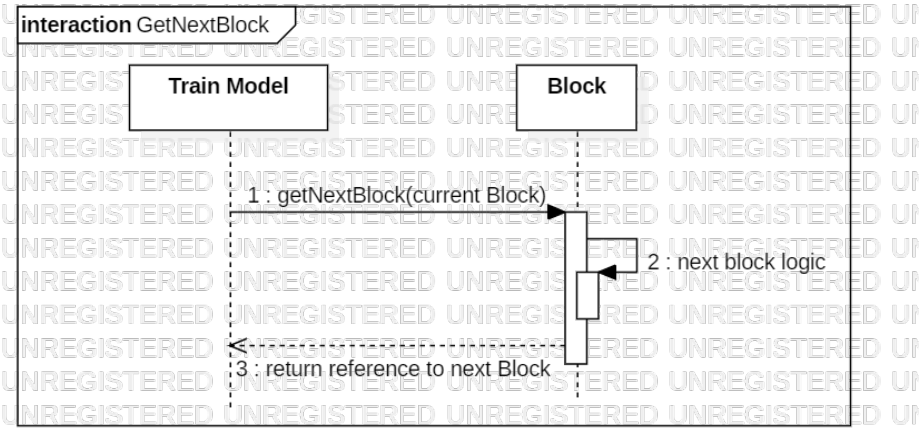

Figure 36: Use Case: Train Model Requests Reference to Next Block in Path

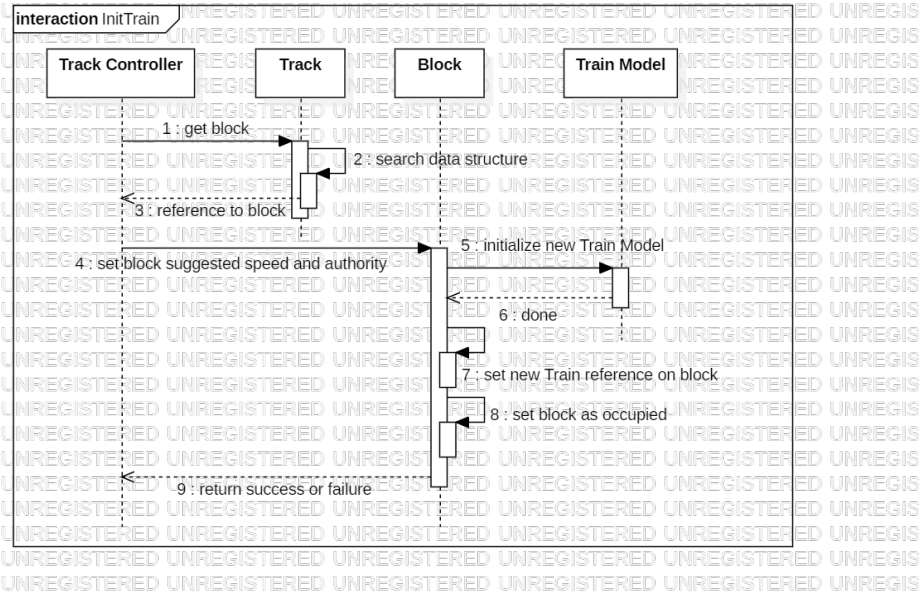

Figure 37: Use Case: Track Controller Initializes Train

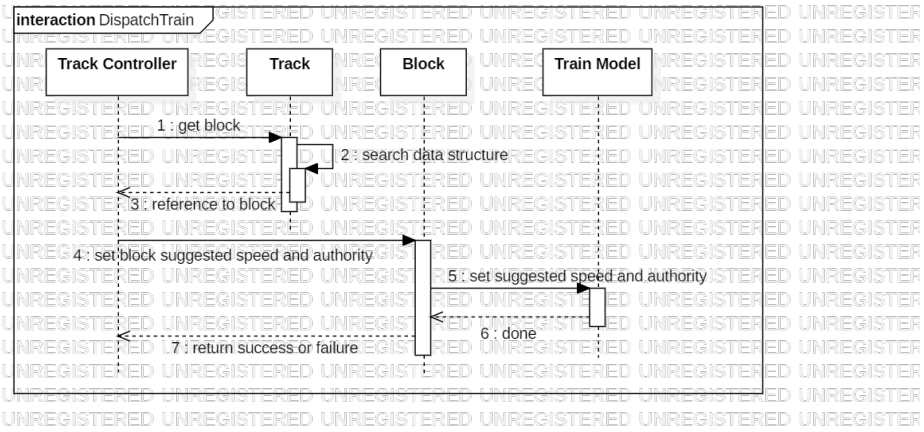

Figure 38: Use Case: Track Controller Dispatches Train

3.4 Train Model

3.4.1 Use Case Diagram

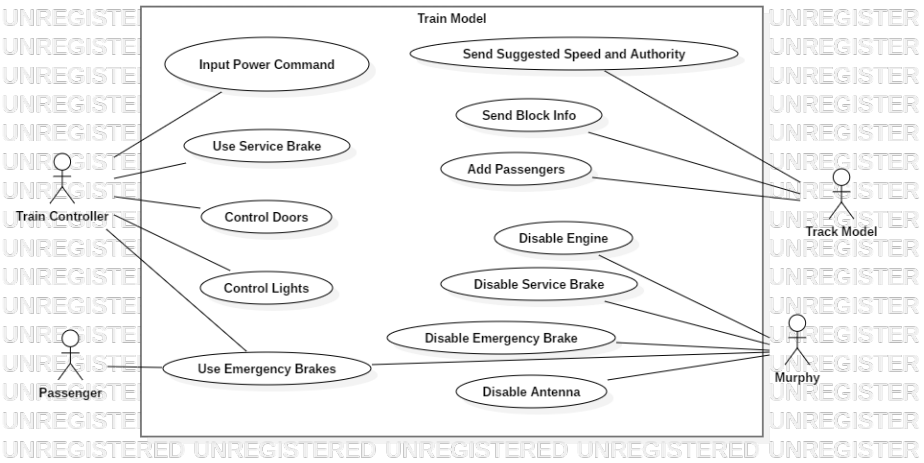

Figure 39: Train Model Use Cases

3.4.2 Use Case Descriptions

System	**Train Model**
Use Case	Input a power command
Actors	Train Controller

Description	• Train Controller inputs a power command • The train calculates the current forces on it such as gravity • The train calculates its new acceleration based on the power and forces • The train calculates its new speed during the next system-wide update
Data	Input: Power command Output: forces, acceleration
Stimulus	Method call from Train Controller
Response	Force and acceleration values are updated

System	**Train Model**
Use Case	Train Controller activates service brake
Actors	Train Controller
Description	• Train Controller activates the service brake • The train decelerates at the max safe rate (see Blackpool specifications sheet) • If brake remains on, train comes to a stop
Data	Input: Service brake command Output: acceleration
Stimulus	Method call from Train Controller
Response	Acceleration is changed, affecting speed on the next train update

System	**Train Model**
Use Case	Train Controller activates emergency brake
Actors	Train Controller
Description	• Train Controller activates the emergency brake • The train decelerates at the max safe rate (see Blackpool specifications sheet) • If brake remains on, train comes to a stop

Data	Input: Emergency brake command Output: acceleration
Stimulus	Method call from Train Controller
Response	Acceleration is changed, affecting speed on the next train update

System	**Train Model**
Use Case	Train Controller sets the lights
Actors	Train Controller
Description	• Train Controller calls a method to turn the lights on or off • The train sets its lights field accordingly
Data	Input: Boolean light state Output: light state
Stimulus	Method call from Train Controller
Response	Train Model lights field set to true or false

System	**Train Model**
Use Case	Train Controller opens or closes a door
Actors	Train Controller
Description	• Train Controller inputs desired state for a certain door • Train sets specified door to that state
Data	Input: door and state Output: state of that door updated
Stimulus	Method call from Train Controller
Response	Door is updated

System	**Train Model**
Use Case	Track Model sends Suggested Speed and Authority
Actors	Track Model
Description	• Track Model inputs suggested speed and authority • Train Model updates its local fields to send to the Train Controller

Data	Input: suggested speed, authority Output: suggested speed and authority fields updated and passed to Train Controller
Stimulus	Method call from Track Model
Response	Train Model holds and passes on new suggested speed and authority

System	**Train Model**
Use Case	Train Model gets block info from Track Model
Actors	Track Model
Description	• Train Model requests block info from the Track Model • Track Model returns appropriate info
Data	Input: Info request Output: speed limit, grade, block length
Stimulus	Method call from Train Model
Response	Updated info on current block

System	**Train Model**
Use Case	Train Model requests the next block
Actors	Track Model
Description	• Train Model calculates that it has traveled over an entire block • Train Model tells the Track Model it has entered the next block • Track Model internally assigns train to the next block • Track Model tells train it has been reassigned • Train Model requests info on the new block (see previous use case)
Data	Input: New block Output: placed in new block and given information for it
Stimulus	Method call from Train Model
Response	Both Track Model and Train Model are appropriately updated

System	**Train Model**
Use Case	Track Model tells a train to update its passengers at a station
Actors	Track Model
Description	• Train comes to a stop at a station • Track Model tells train to remove some passengers (if any present) • Train removes some passengers and tells the track how much free space it has • Track adds a random number of passengers such that the train is not overfilled • Train updates its mass field based on number of passengers
Data	Input: train stop at station Output: train passenger number, train mass
Stimulus	Method call from Track Model
Response	Train has updated passenger and mass fields

System	**Train Model**
Use Case	Murphy causes a failure
Actors	Murphy
Description	• Murphy chooses a failure state to set (engine failure, service/emergency brake failure, antenna failure) • Train Model updates the selected failure state and modifies its behavior accordingly
Data	Input: failure state, Boolean Output: updated fields in Train Model
Stimulus	Method call from Murphy
Response	Train sets specified failure state, potentially altering its behavior

3.4.3 Class Diagrams

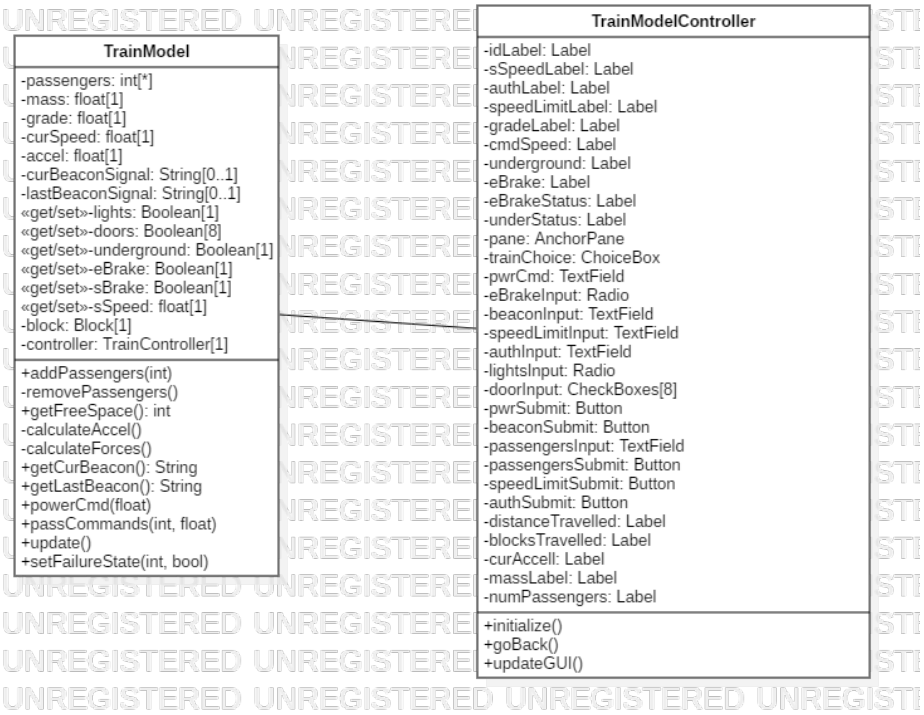

Figure 40: Train Model Class Diagram

3.4.4 Sequence Diagrams

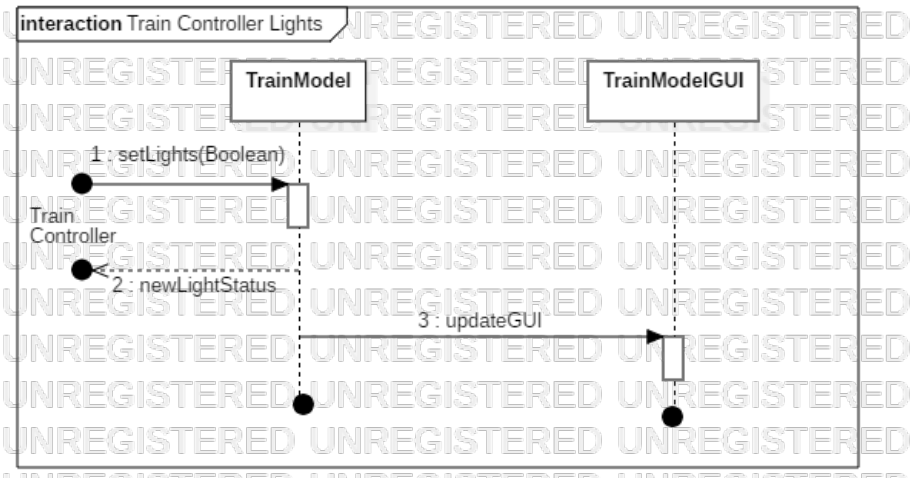

Figure 41: Use Case: Train Controller sets Lights

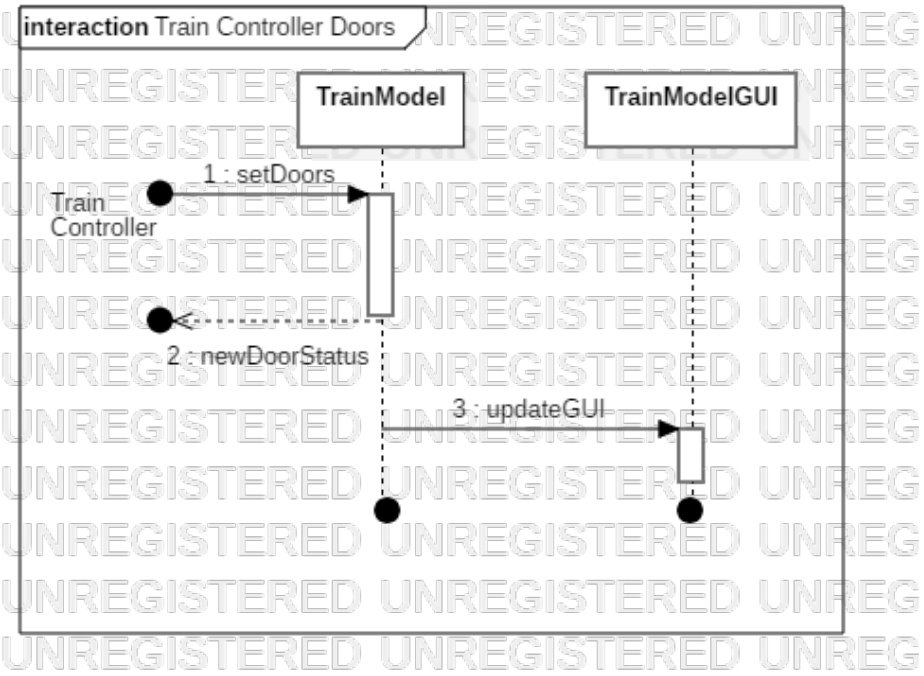

Figure 42: Use Case: Train Controller sets Doors

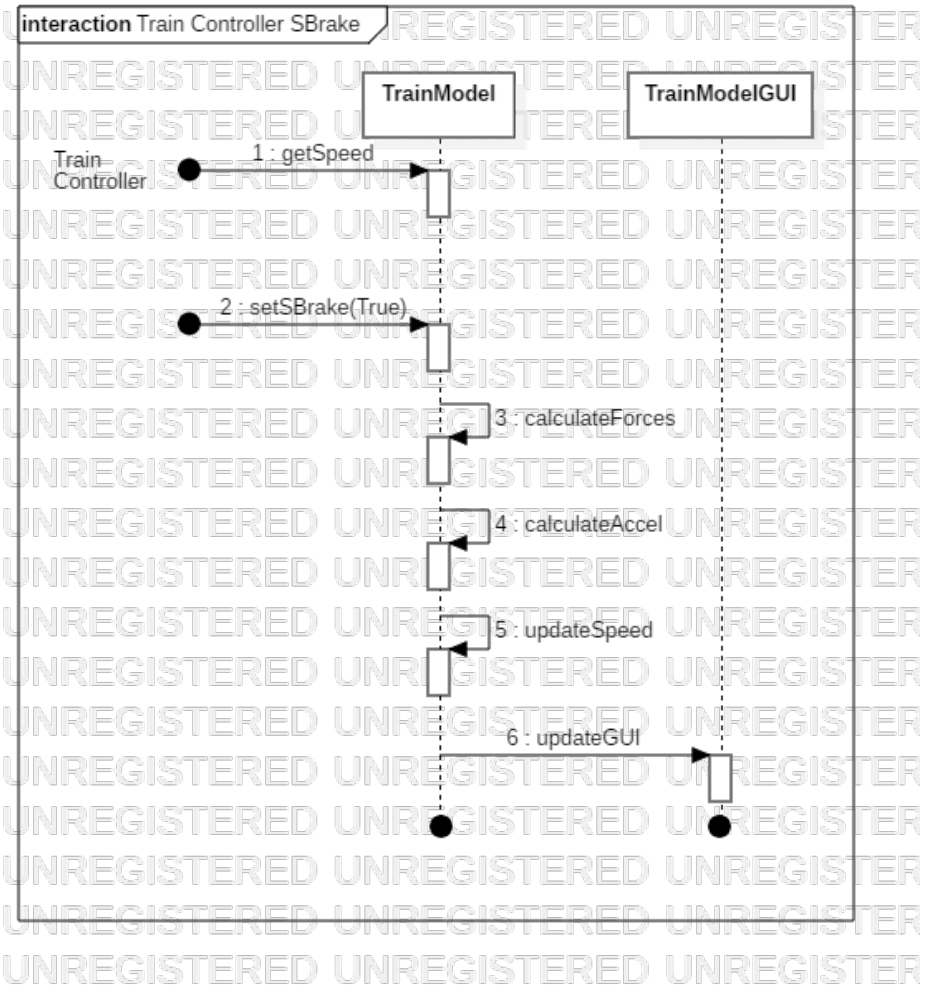

Figure 43: Use Case: Train Controller sets Service Brake

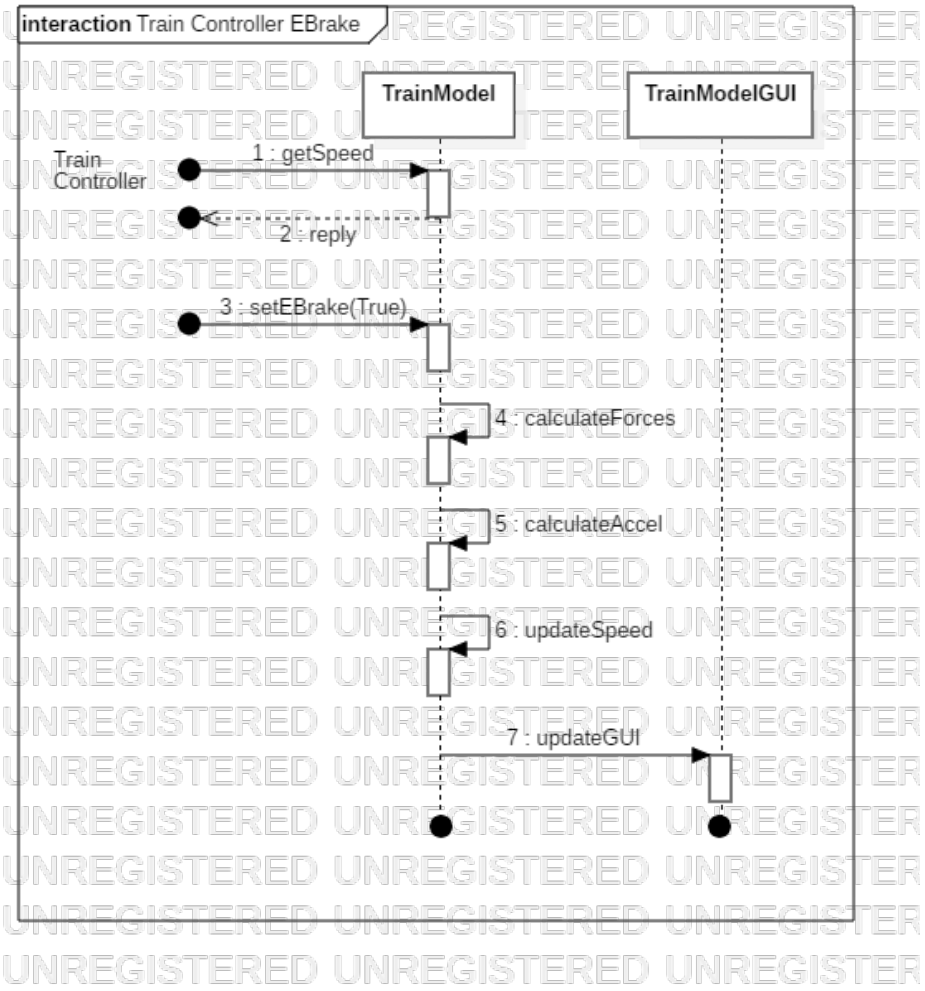

Figure 44: Use Case: Train Controller sets Emergency Brake

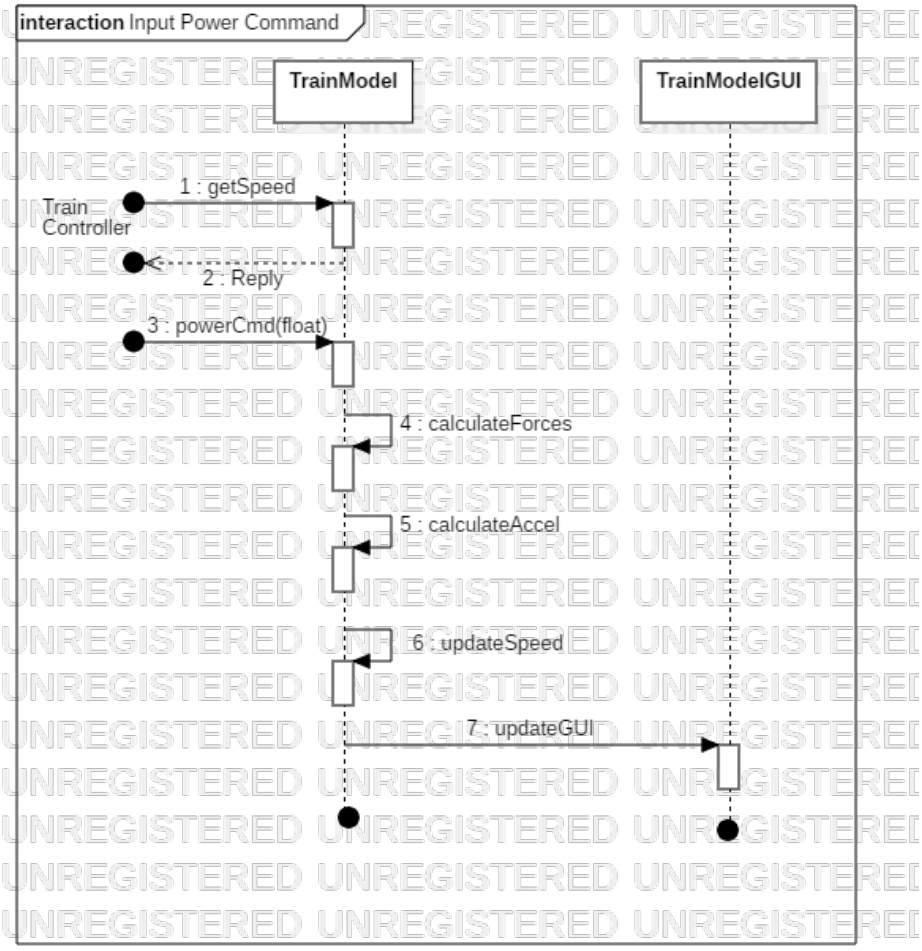

Figure 45: Use Case: Train Controller Gives Power Command

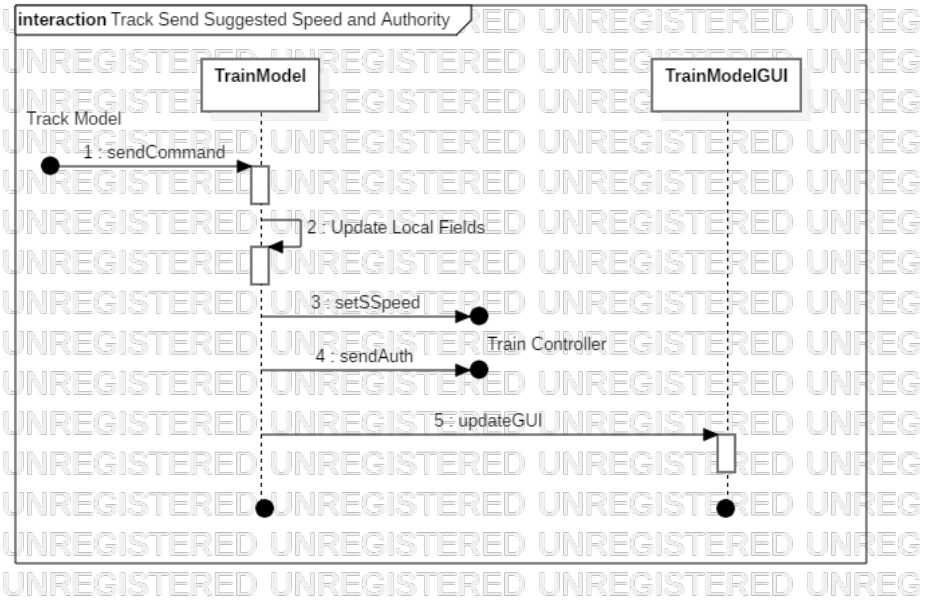

Figure 46: Use Case: Track Model Sends Suggested Speed and Authority

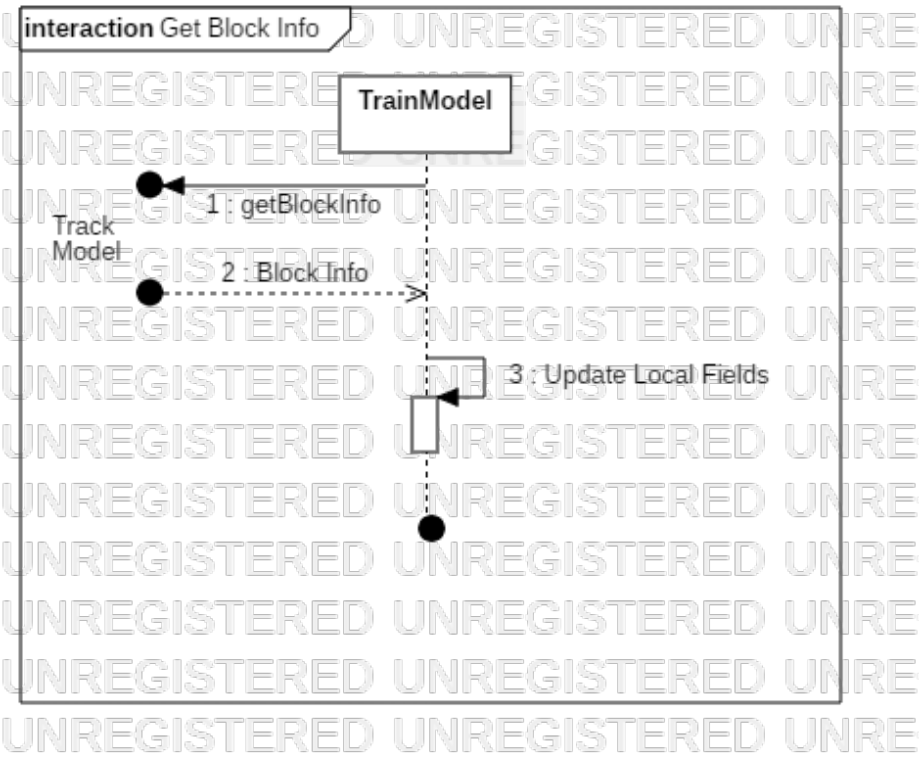

Figure 47: Use Case: Train Model gets Block Info from Track Model

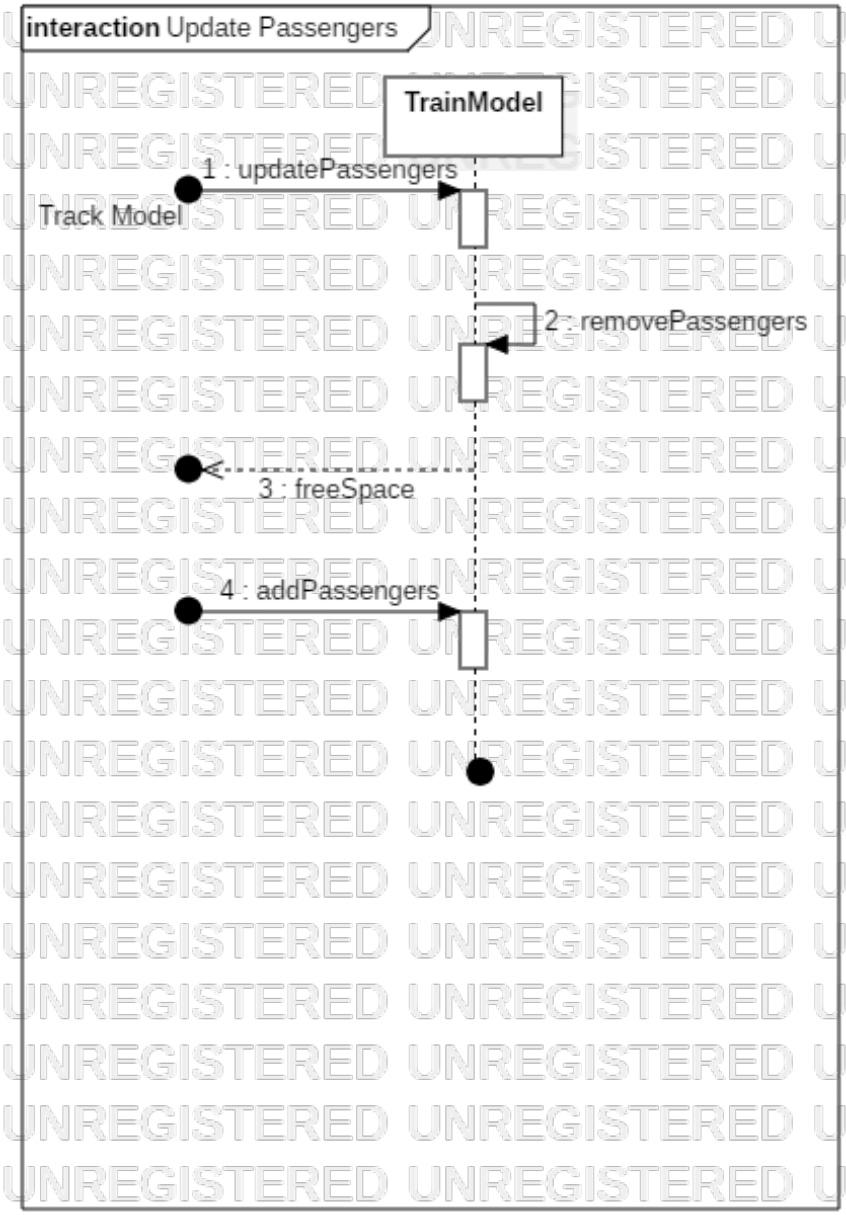

Figure 48: Use Case: Track Model tells Train Model to Update Passengers

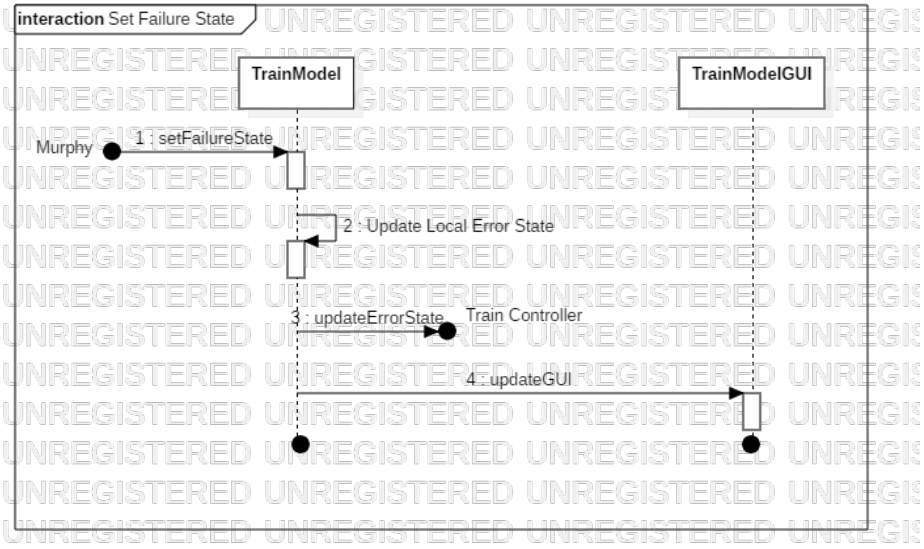

Figure 49: Use Case: Murphy sets a Failure State for the Train Model

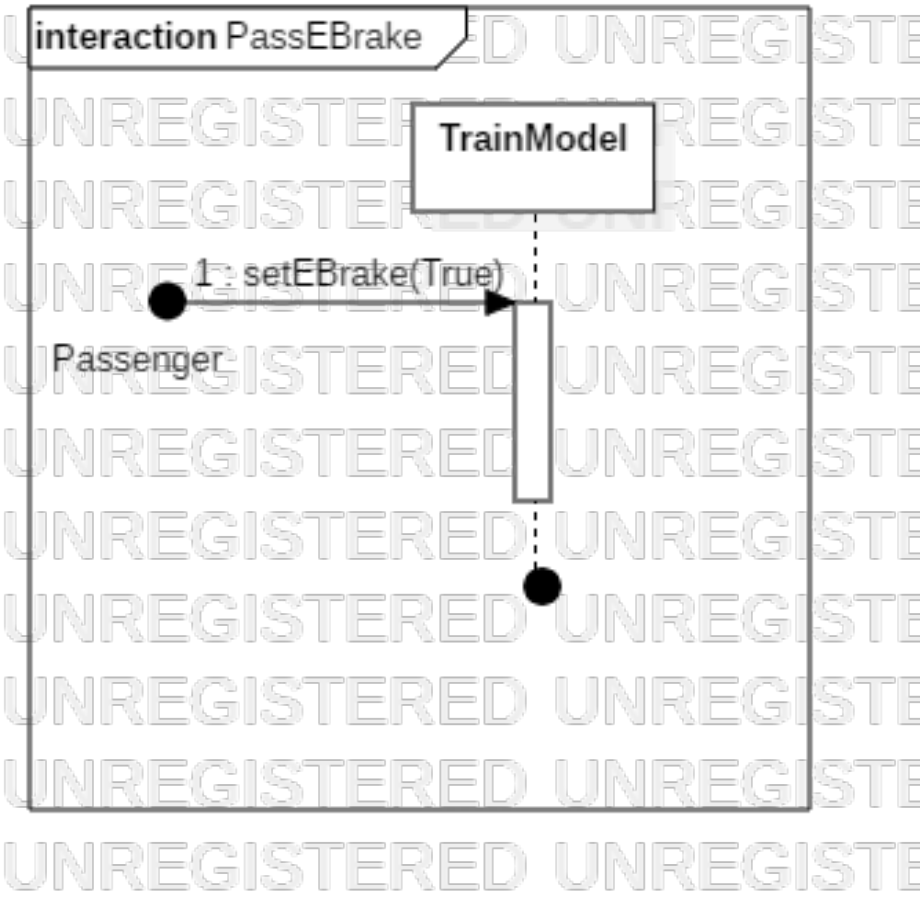

Figure 50: Use Case: Passengers Activate Emergency Brake

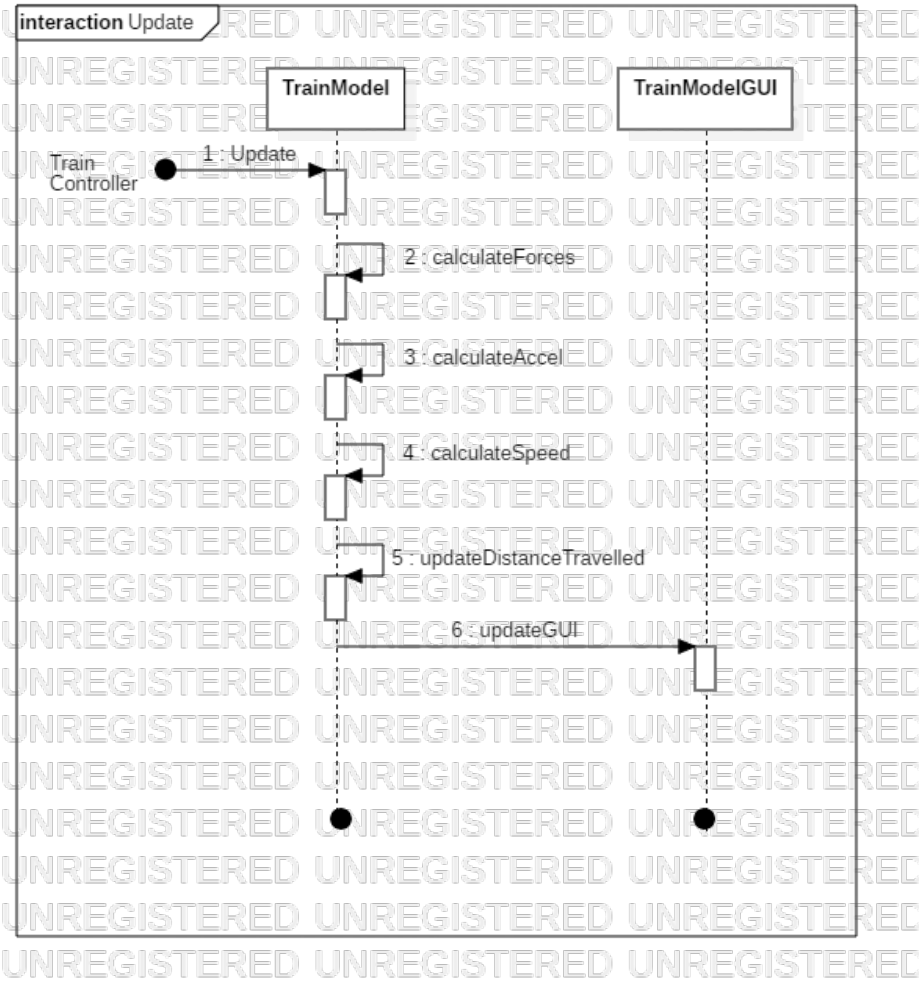

Figure 51: Use Case: Train Model Updates Itself

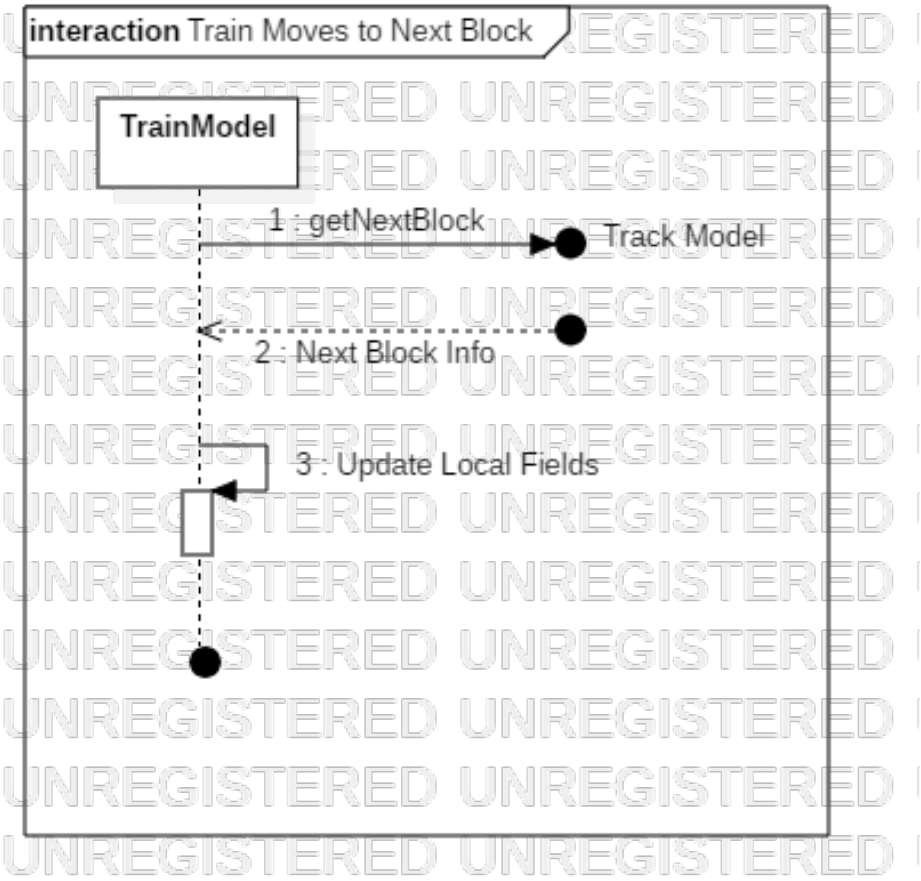

Figure 52: Use Case: Train Model gets Next Block from Track Model

3.5 Train Controller

3.5.1 Use Case Diagram

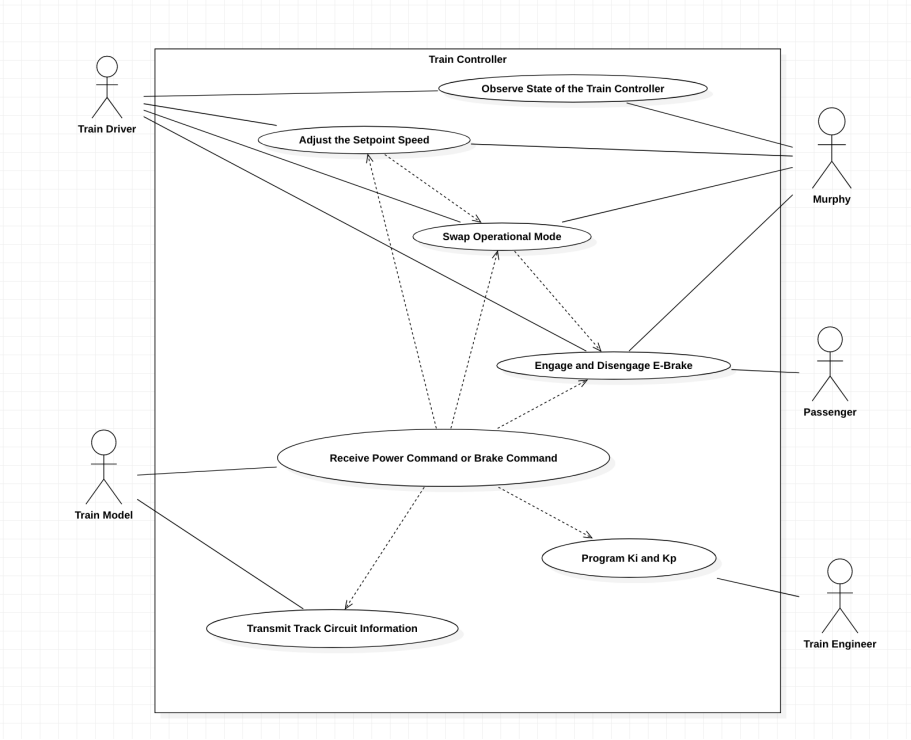

Figure 53: Train Controller Use Cases

3.5.2 Use Case Descriptions

System	**Train Controller**
Use Case	Observe State of the Train Controller
Actors	Train Driver/Murphy
Description	• Train Controller collects current operation data • Train Controller GUI accepts this data • Train Controller GUI updates its view with the new information
Data	Input: None Output:

Stimulus	Display of Train Controller Graphic User Interface
Response	Current operating conditions presented to use

System	**Train Controller**
Use Case	Adjust Set-point Speed
Actors	Train Driver/Murphy
Description	• Actor inputs a new set-point speed • If E-Brake is activated, E-Brake signal is transmitted • Sub-case: Train is operating in automatic mode – Set-point speed is ignored in the determination of the power command and brake command • Sub-Case: Train is operating in manual mode – Set-point speed is compared to suggested speed, lower speed is used to determine power and brake command • If determined speed is lower than operating speed, S-Brake signal is transmitted • If determined speed is higher than operating speed, Power command is transmitted
Data	Input: Setpoint Speed Output: Power/Brake Command to Train Model
Stimulus	Actor enters a new setpoint speed
Response	Power or Brake command is transmitted to Train Model

System	**Train Controller**
Use Case	Swap operational Mode
Actors	Train Driver/Murphy

Description	• Actor selects a new operational mode • Sub-case: Actor selects Manual – Train controller enters manual mode – The Set-point speed of the train defaults to the suggested speed – The train controller user interface allows the user to input a set point speed – The train controller considers the set-point speed in deciding commands to the train model • Sub-Case: Actor selects Automatic – Train Controller enters automatic mode – The train controller user interface restricts the adjustment of the setpoint speed – The train controller does not consider the setpoint speed in deciding commands to the train model
Data	Input: None Output: None
Stimulus	Actor selects an operational mode from the Train Controller user interface
Response	Operational decisions of the train controller are adjusted appropriately

System	**Train Controller**
Use Case	Engage and Disengage E-Break
Actors	Train Driver/Murphy/Passenger

Description	• Actor manipulates the E-Brake Control from either automatic or manual mode • Sub-Case: E-Brake is activated – The train controller transmits an E-Brake command to the train model – The train controller sets the mode of operation to manual – The manipulation of the operational mode is restricted while the E-Brake is activated • Sub-Case: E-Brake is deactivated – The Train controller stops transmitting the E-Brake command to the train model – The train controller begins normal operation in the operational mode that was present before the E-Brake signal was activate. – Manipulation of the E-Brake is enabled
Data	Input: Setpoint Speed Output: Power/Brake Command to Train Model
Stimulus	Actor enters a new setpoint speed
Response	Power or Brake command is transmitted to Train Model

System	**Train Controller**
Use Case	Program Ki and Kp
Actors	Train Engineer
Description	• Train Engineer enters the programming view from the main display • The Train engineer enters the new Ki and Kp into the display • The new constants are utilized through all the train controllers of the system
Data	Input: Ki (Double) Kp (Double) Output: None
Stimulus	Train Engineer enters programming view from the main display view

Response	Train Controller is programmed with new PID constants

System	**Train Controller**
Use Case	Transmit Track Circuit Information
Actors	Train Model
Description	• Train Model receives a suggested speed, authority, and beacon information from the track circuit • Train model sends that information to the train controller
Data	Input: Setpoint Speed Output: Power/Brake Command to Train Model
Stimulus	Train model receives new suggested speed, authority, and beacon, from the track circuit
Response	Train controller commands to train model correctly respond to transmitted information

System	**Train Controller**
Use Case	Receive Power Command From Train
Actors	Train Model
Description	• Use case "Program Ki and Kp" must have already been done • Do use case "Transmit Track Circuit Information" • The Train Controller will determine a proper t
Data	Input: None Output: Power Command
Stimulus	Function Call to Train Controller
Response	Power or Brake command is transmitted to Train Model

3.5.3 Class Diagrams

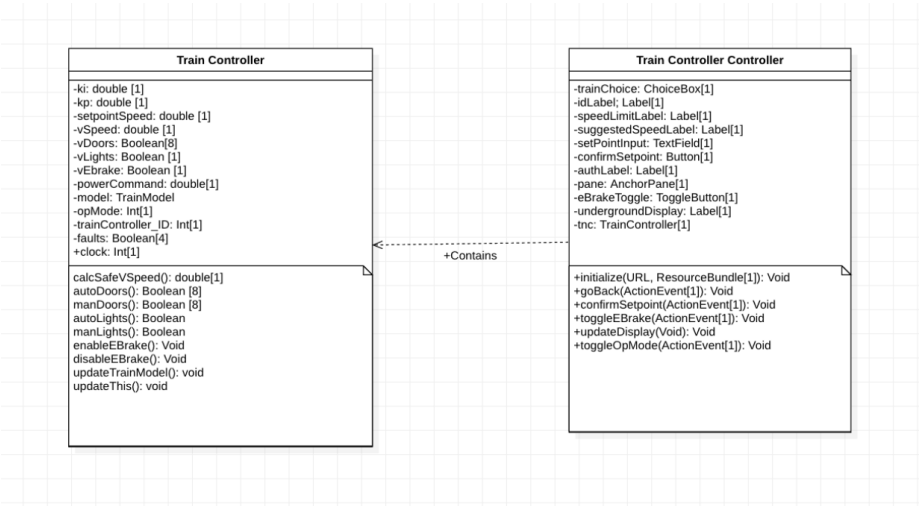

Figure 54: Train Controller Class Diagram

3.5.4 Sequence Diagrams

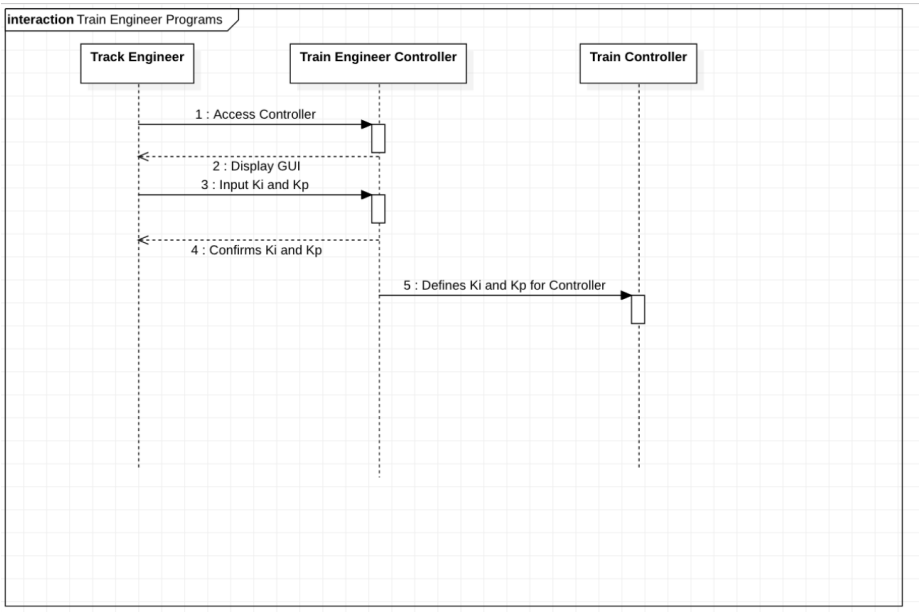

Figure 55: Train Engineer Programs Ki and Kp

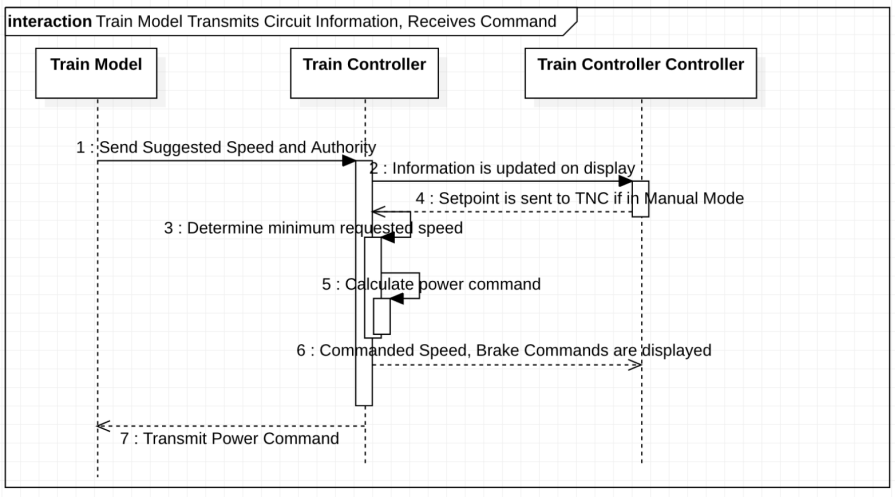

Figure 56: Train Model Transmits Track Circuit Information

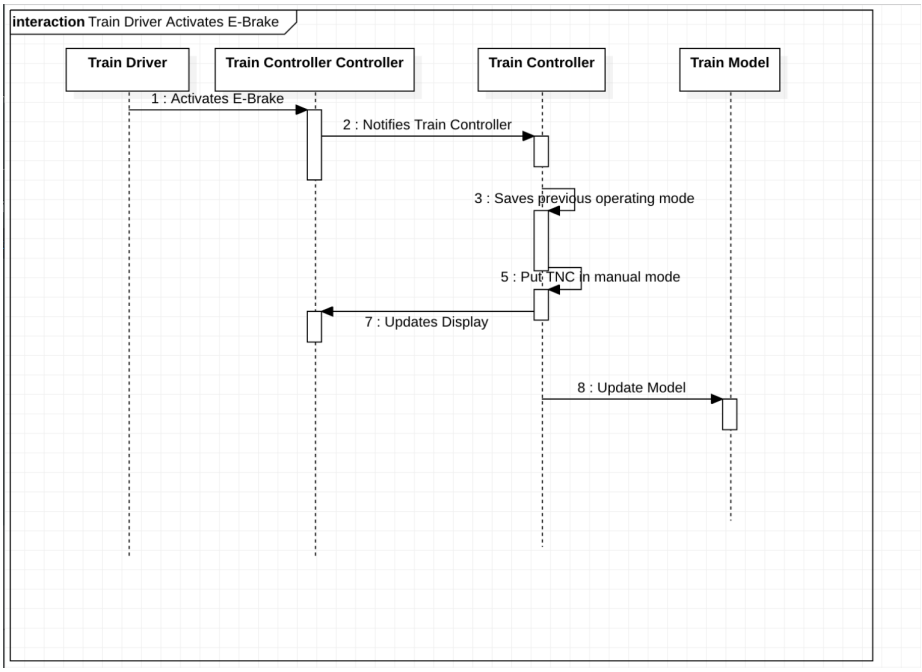

Figure 57: Train Driver or Murphy activates E-Brake

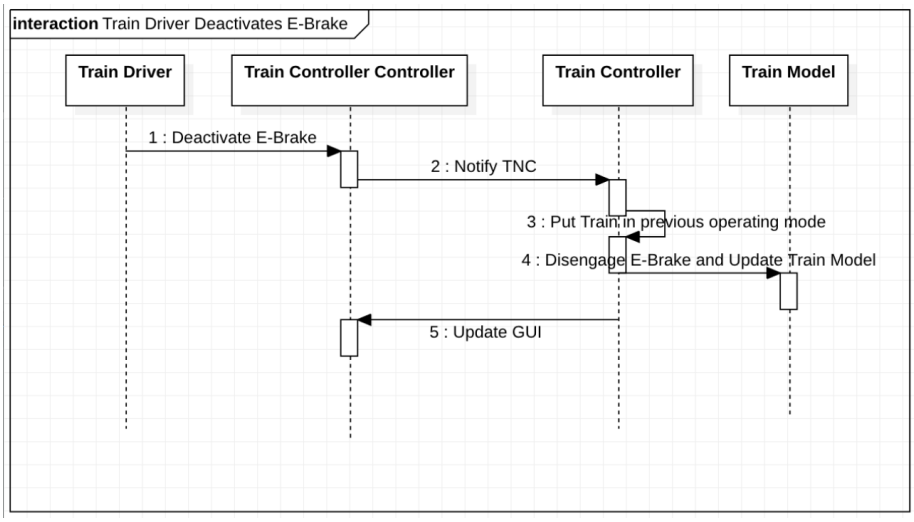

Figure 58: Train Driver or Murphy deactivates E-Brake

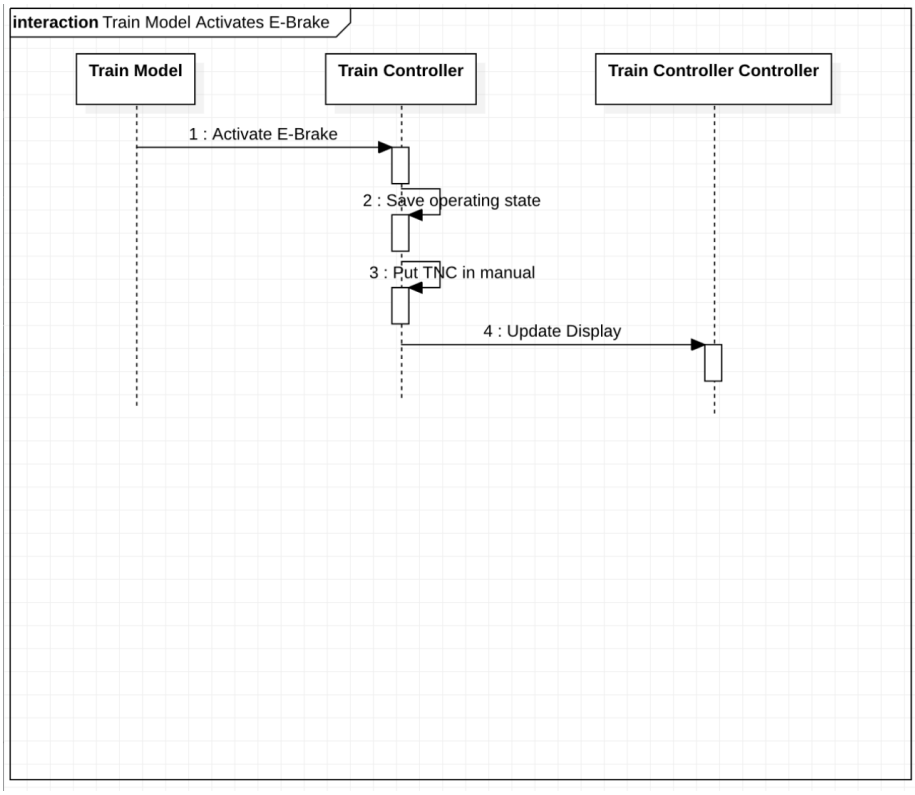

Figure 59: Train Model activates E-Brake

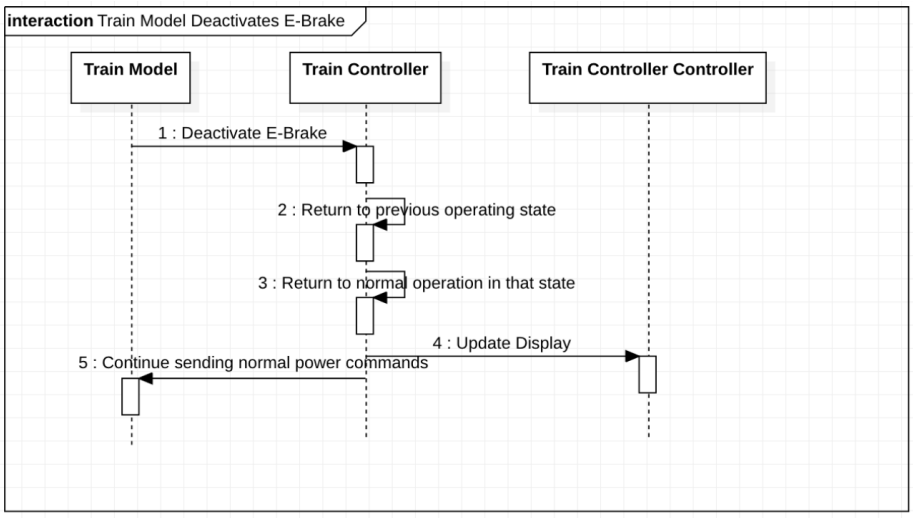

Figure 60: Train Model deactivates E-Brake

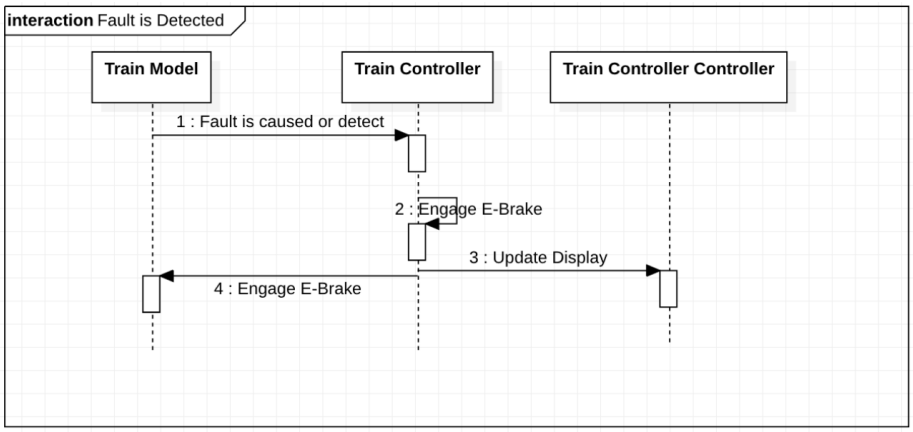

Figure 61: Train Controller is notified of a fault