



Bilkent University

Department of Computer Engineering

CS 319 - Object Oriented Programming Term Project

RSim: Railway Simulator

Analysis Report

Semahat Elif Dayı, Zahit Saygın Dođu, Mevlüt Geredeli, Gizem Uzuner

Instructor: Ertuđrul Kartal Tabak

Analysis/Final Report

March21, 2015

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object Oriented Programming course CS319.

Contents

Introduction	1
Current System	2
Proposed System	3
1.1 Overview	3
1.2 Functional Requirements	3
1.3 Non-functional Requirements	3
1.4 Pseudo Requirements	4
1.5 System Models	4
1.1.1 Scenarios	4
1.1.2 Use-Case Model	5
1.1.3 Object and Class Model	8
1.1.4 Dynamic Models	10
1.6 User Interface	13
References	15

Analysis Report

RSim: Railway Simulator

Introduction

Simulation is a key application of estimating the future states of the systems that are working on a structured basis. Computer aided simulators helps people to analyze complicated systems and react on the real systems using the confidence from the simulation, knowing that the system will work properly after its construction is complete or it won't collapse if some properties of the system is changed.

Railway systems can be very complex and it can be hard to estimate the effects of the design decisions or changes in the system. If there would be a convenient railway simulation software that could ease the work of the designers and maintainers of such systems. Therefore we decided to design and implement a railway simulator software called RSim, which provides a tool for designing railway systems and simulating the given system to observe how the system works. This simulator allows user to create new railway route and edit existing routes. User can add new stations, trains and edit their properties. Eventually, railway system is simulated and user can observe system in action.

Our aim while creating RSim is to show how a railway system works and enable users to build efficient railway systems or create more efficient timetables or plans by simulating and observing the effects of the changes. User can create different variation of routes and observe what will happen at the desired time. Hence, once simulated with satisfying results, the system can be constructed or changes in an existing system can be implemented.

This report contains the overview of the simulator, specifies the simulator's requirements and analysis of the system. This report includes the architectural patterns which will be used in our simulator software and current system that is familiar with our project. Moreover, report gives information about functional requirements, non-functional requirements, pseudo requirements, use-case models that include scenarios, use-case diagrams, object and class model, dynamic model, which are a part of the analysis stage. At the end of the report, there are mockups of user interface of the RSim software.

Current System

We have made an online search for existing railway simulator software and we have found a satisfying software called OpenTrack. Here is the explanation of the software website:

OpenTrack[2] began in the mid-1990s as a research project at the Swiss Federal Institute of Technology. The aim of the project *Object-Oriented Modeling in Railways* was to develop a catalyst for practical economic solutions to complex railway technology problems.

Today, the railway simulation tool OpenTrack is a well-established railway planning software and it is used by railways, the railway supply industry, consultancies and universities in different countries.

OpenTrack allows modeling, simulating and analyzing the following types of rail systems:

- High speed rail
- Heavy rail / Intercity rail
- Commuter rail systems
- Heavy haul freight
- Mining railway systems
- Metro / Subway / Underground systems
- Light rail (LRT)
- Tram / Streetcar systems
- People mover systems
- Rack railways / Mountain railways
- Maglev (magnetic levitation) systems (e.g. Transrapid)

OpenTrack supports the following kinds of tasks:

- Determining the requirements for a railway network's infrastructure
- Analyzing the capacity of lines and stations
- Rolling stock studies (for example, future requirements)
- Running time calculation
- Timetable construction; analyzing the robustness of timetables (single or multiple simulation runs, Monte-Carlo simulation)
- Evaluating and designing various signaling systems, such as *discrete block systems, short blocks, moving blocks, LZB, CBTC (communication-based train control), ATP, ATO, ETCS Level 1, ETCS Level 2, ETCS Level 3* (see also: **ERTMS**)
- Analyzing the effects of system failures (such as infrastructure or train failures) and delays
- Calculation of power and energy consumption of train services
- Simulation of railway power supply systems (using **OpenPowerNet**)

Proposed System

While the existing software OpenTrack provides a vast amount of functionalities, the complexity of the system makes it hard to learn and start using it. Our software RSim won't be as complicated as OpenTrack and provide considerably less functionalities. However, the main goal of the RSim is to learn the software development stages for the team so we will implement the project.

1.1 Overview

The purpose of this project is to simulate a railway system. For simplicity there is only one line with no cross sections. User can add a new station, specify the distance between the stations, edit and remove the stations if necessary. User can also specify the capacity of the trains, the frequency of the trains and number of the trains. User should also specify the average quantity of people coming to a specific station, waiting and times for the trains for that station. After everything is specified user can advance time to see how full the trains and stations are and how the system works. Then user can modify the parameters and observe the system so that the user can achieve optimal operation strategy for the system.

We are planning to realize our design by using Java programming language. We are planning to build a simple GUI for adjusting parameters and showing the state of the system. However, since it will exceed the scope of the course adding any animations or graphical representations for the line, stations and trains will be considered to be added at the end of the implementation.

We decided to pick this topic for the project because we see high potential for expanding. If the project won't satisfy the requirements for the course, we can always add new features.

1.2 Functional Requirements

- User should add new stations to the system.
- User should define the distances between the stations.
- User should define train specifications (such as speed, capacity...)
- User can change the station properties at edit time.
- User is not allowed to modify the properties of trains and stations during simulation time.
- User can save the layout of the stations to a file.
- User can see the system state after some defined time by the user.
- User can view the simulation as time passes.
- Controller won't let trains to be overlapped. This means, in between two adjacent stations there can be only one train at a time instant.
- User can log in to system.
- User can open and simulate a saved system by any user.
- User cannot edit a system saved by other users.

1.3 Non-functional Requirements

- All user inputs should be acknowledged within 1 second.
- A system crash should not result in long term data loss.
- Any simulation calculation taking more than 1 minute will terminate and result in warning.

1.4 Pseudo Requirements

- The System will be implemented on Java. Because Java is one of the best object oriented programming languages and it provides nice UI API's to make platform independent applications.

1.5 System Models

1.1.1 Scenarios

Ertuğrul has a childhood dream of creating a virtual train system and simulation its behavior. Ertuğrul finds out there is software called Railway Simulator for this. Ertuğrul gets the program, starts it. Ertuğrul sees the user interface. Ertuğrul first creates an account for the program. After that he logs in to the system and starts editing. He first creates a track by using create track button. He creates a track initially with three stations. He decides he wants to add some more stations and adds them by using add station button. After that he creates a train dispatcher at the beginning of the track. After this he creates 3 trains using add train button. He specifies the destination and the departure hour vector of the trains. After that he is satisfied with the design of the railway system so he decides to simulate it. He pushes the simulate button and watches the simulation for hours. After a while he gets bored and decides to close the program. He stops the simulation and saves state of the railway design. After that he closes the program. After closing the program he gets curious and opens the program with another account to see what happens. He locates the design file to open it. He is able to see the design, simulate it but he notices that he can't change the design. It is because he is not logged in as his original profile.

1.1.2 Use-Case Model

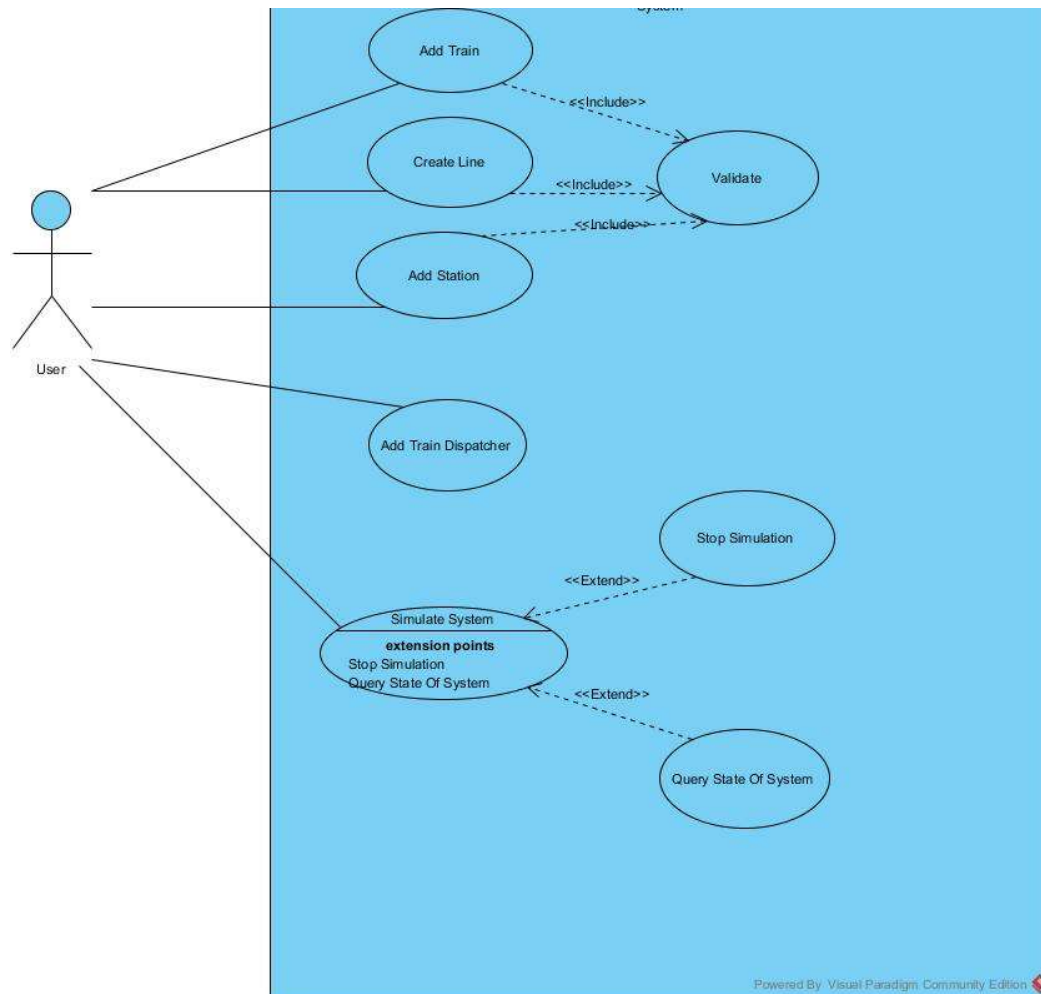


Figure 1: Use Case Diagram

In the Figure 1 use case diagram of RSim is given. The corresponding textual descriptions of the use cases are given below.

Use Cases

Adding Station

Use Case Name: Adding Station

Participating actors: User (Ertuğrul)

Entry Conditions:

- Program is running.
- There is no simulation in progress.
- User is logged in.
- There is a track existing.

Flow of Events:

- Ertuğrul presses the button for adding new station.,
- Ertuğrul specifies the properties of the station.
 - Ertuğrul can specify the name of the station.
 - Ertuğrul can specify the position of the station.
 - Ertuğrul can specify the maximum wagon capacity of the station.
- If the station is overlapping with another station, or it is conflicting with the line specifications, Ertuğrul will see a warning.
- If there is no warning Ertuğrul can confirm the addition of the station.
- The station will be added to the line system.

Exit Condition:

- The station is added, or the station addition process is aborted.

Creating a New Track

Use Case Name: Creating a new track

Participating actors: User (Ertuğrul)

Entry Conditions:

- Program is running.
- There is no simulation in progress.
- User is logged in.
- There is no track existing.

Flow of Events:

- Ertuğrul presses the button for adding new line.
- Ertuğrul specifies the properties of the line.
 - Ertuğrul can specify the minimum distance between stations.
 - Ertuğrul can specify the maximum length of the track.
 - Ertuğrul can specify the maximum station count in the line.
 - Ertuğrul can specify the name of the line.
 - Ertuğrul can specify the name of the first station.
 - Ertuğrul can specify names and the positions of the following stations.
- If any station is overlapping with another station, or it is conflicting with the line specifications, Ertuğrul will see a warning.
- If there is no warning Ertuğrul can confirm the creation of the line.
- The track will be added to the line.

Exit Condition:

- The new track is added or the track addition process is aborted.

Adding a Train

Use Case Name: Adding a train

Participating actors: User (Ertuğrul)

Entry Conditions:

- Program is running.
- There is no simulation in progress.
- User is logged in.
- There is a track existing with at least 2 stations.

Flow of Events:

- Ertuğrul presses the button for adding new train.
- Ertuğrul specifies the properties of the train.
 - Ertuğrul can specify which train dispatcher the train will start it's service.
 - Ertuğrul can specify the capacity of the train.
 - Ertuğrul can specify the count of the wagon.
 - Ertuğrul can specify the maximum capacity of each wagon.
 - Ertuğrul can specify the departure time table for the train.
 - Ertuğrul can specify the directions which the train will go at each departure time.
- If the wagon count is more than the stations can handle, the program will give a warning.
- If there is no warning Ertuğtul can confirm the addition of the train.
- The train will be added to the line system.

Exit Condition:

- The train is added to the dispatching queue of the dispatcher.

Add a Train Dispatcher

Use Case Name: Add Train Dispatcher

Participating actors: User (Ertuğrul)

Entry Conditions:

- Program is running.
- There is no simulation in progress.
- User is logged in.
- There is a track existing with at least 2 stations.

Flow of Events:

- Ertuğrul presses the button for adding new traindispatcher.
- If there is a dispatcher on the station, Ertuğrul will see a warning message.
- Ertuğrul specifies the properties of the train dispatcher.
 - Ertuğrul must specify which station the train dispatcher will work at.
 - Ertuğrul can specify which way the train will move initially.

Exit Condition:

- The train dispatcher is added to the station, or the existing dispatcher is overridden.

Simulate System

Use Case Name: Simulate the System

Participating actors: User (Ertuğrul)

Entry Conditions:

- Program is running.
- There is no simulation in progress.
- User is logged in.
- There is a track existing with at least 2 stations.
- There is at least one train dispatcher on one of the stations.

Flow of Events:

- Ertuğrul presses the button for starting the simulation.
- Ertuğrul will see the simulation results in the screen.
 - Ertuğrul can adjust the speed of the simulation.
 - Ertuğrul can stop the simulation.
- After stopping Ertuğrul can save the simulation process.

Exit Condition:

- The simulation process is either saved or nothing happens.

1.1.3 Object and Class Model

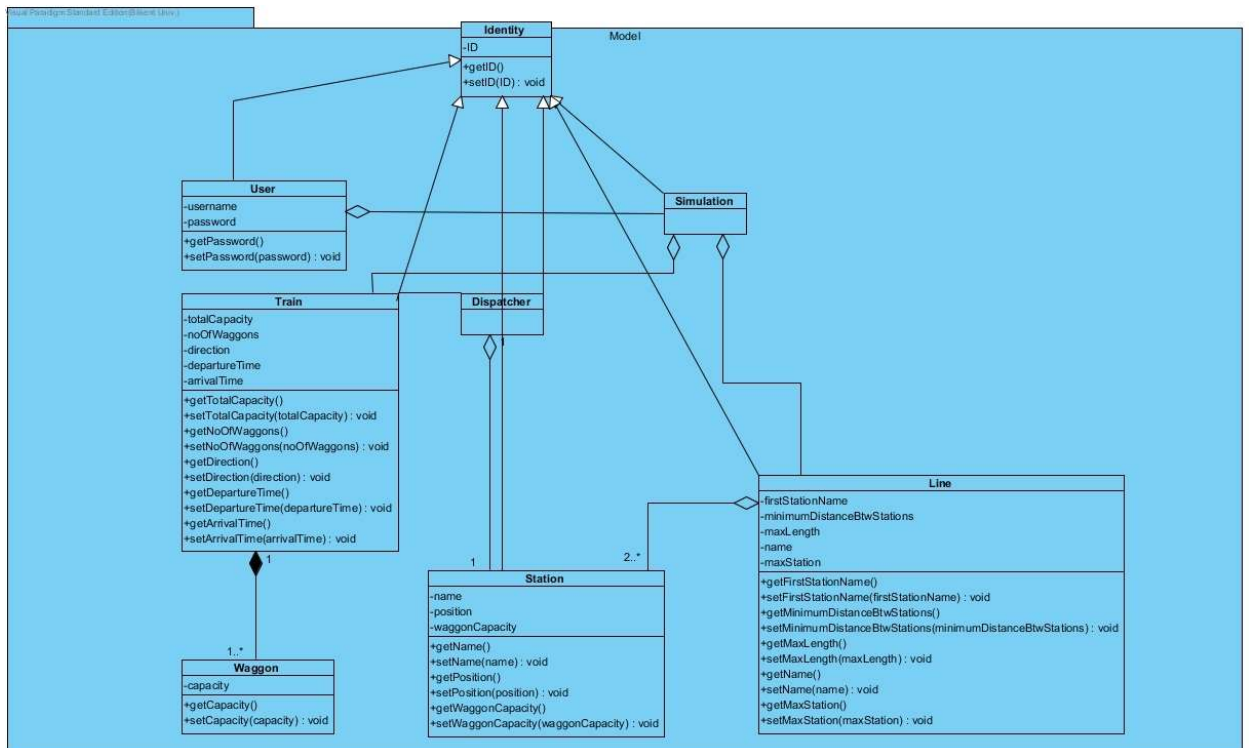


Figure 2: Model Package Class Diagram

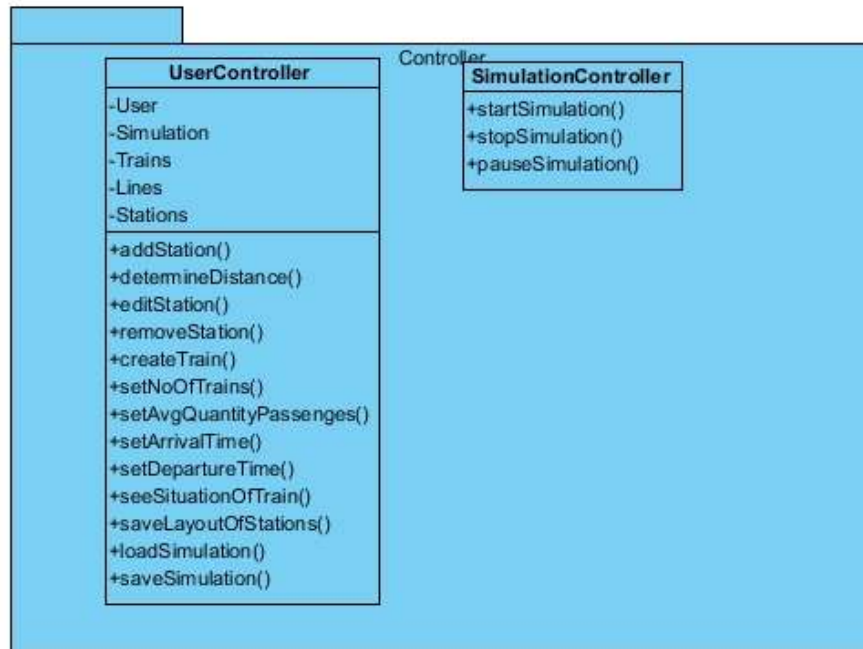


Figure 3: Controller Package Class Diagram

1.1.4 Dynamic Models

*
□

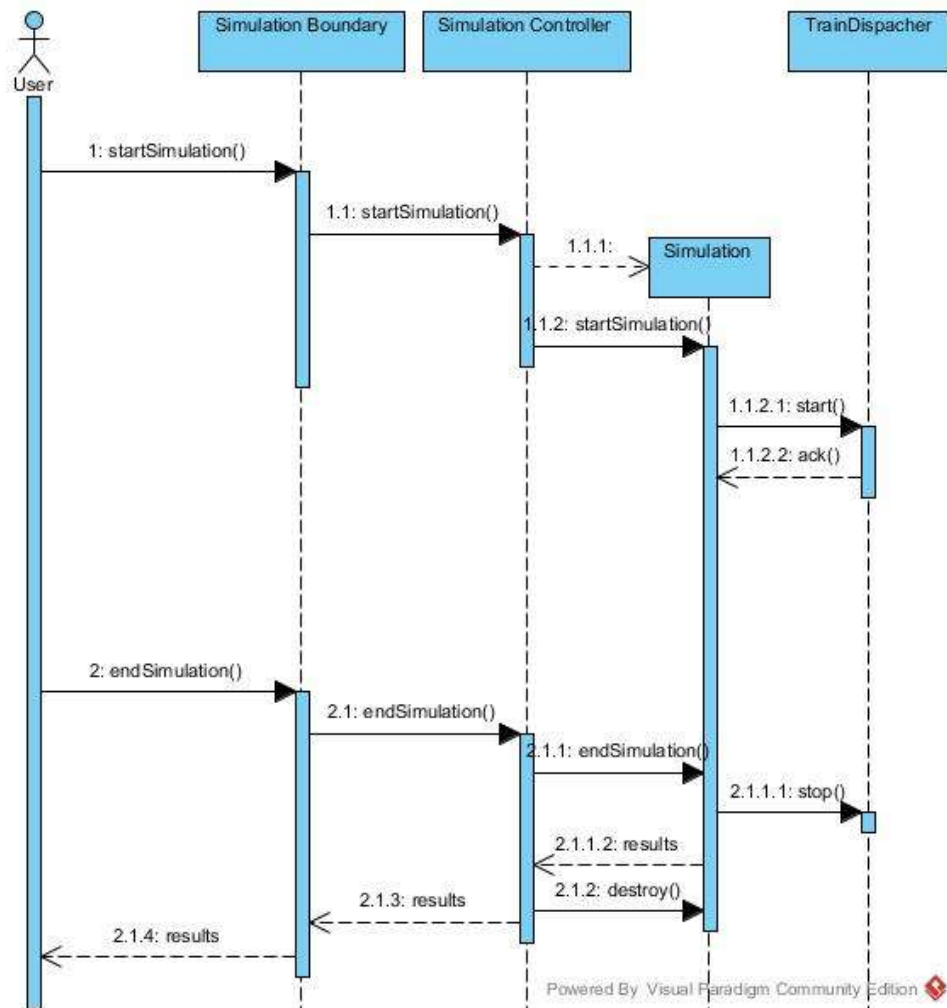


Figure 4: Sequence Diagram for Simulating the System

In Figure 4, the sequence diagram of simulating a system is given. The user pushes the button to start simulation, which is on the SimulationBoundary object. SimulationBoundary object is the View object associated with a simulation. When the button is pressed, SimulationBoundary notifies SimulationController which then creates a Simulation instance. After that the calculations are made and shown to the user until the user presses the stop button.

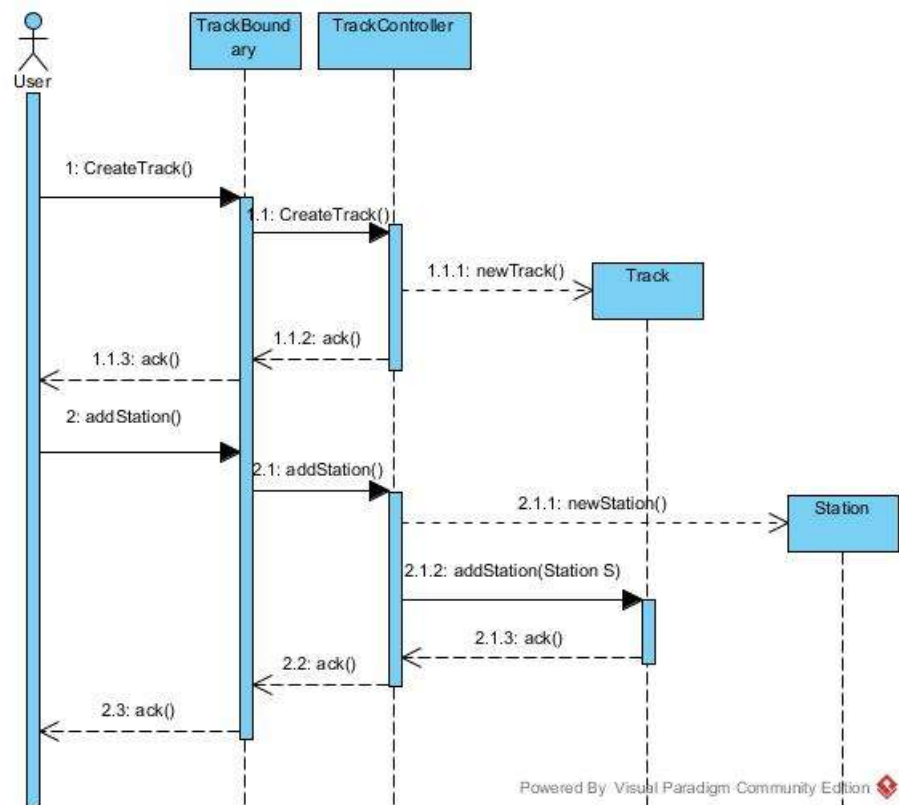


Figure 5: Sequence Diagram for Creating a Track

In Figure 5, the sequence diagram of creating and managing a track is given. The user pushes the add new track button which is on TrackBoundary which is a View object associated with the Track. After this, the TrackController is notified and it will create a new Track instance. After this the user clicks the button for adding a station to a track, then the TrackController object then it creates a new Station instance and links it to the associated Track object. The changes in the Track model will be shown by the TrackBoundary object to the user.

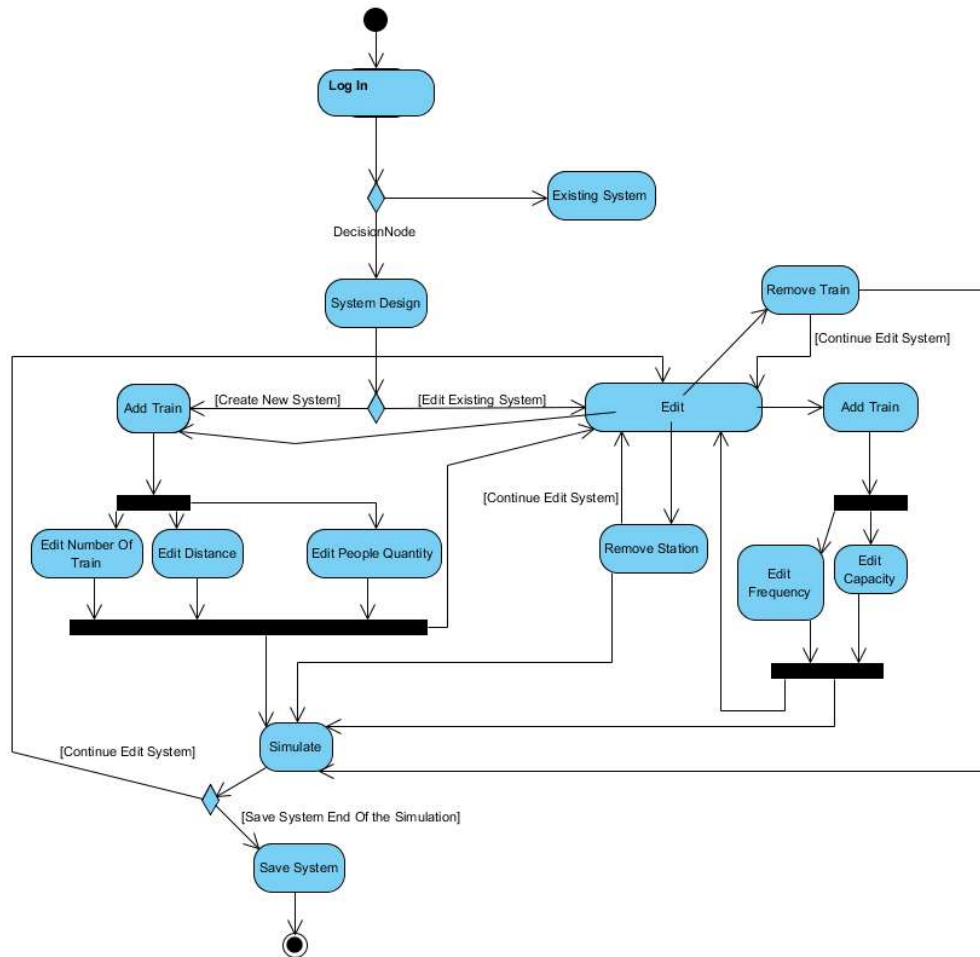
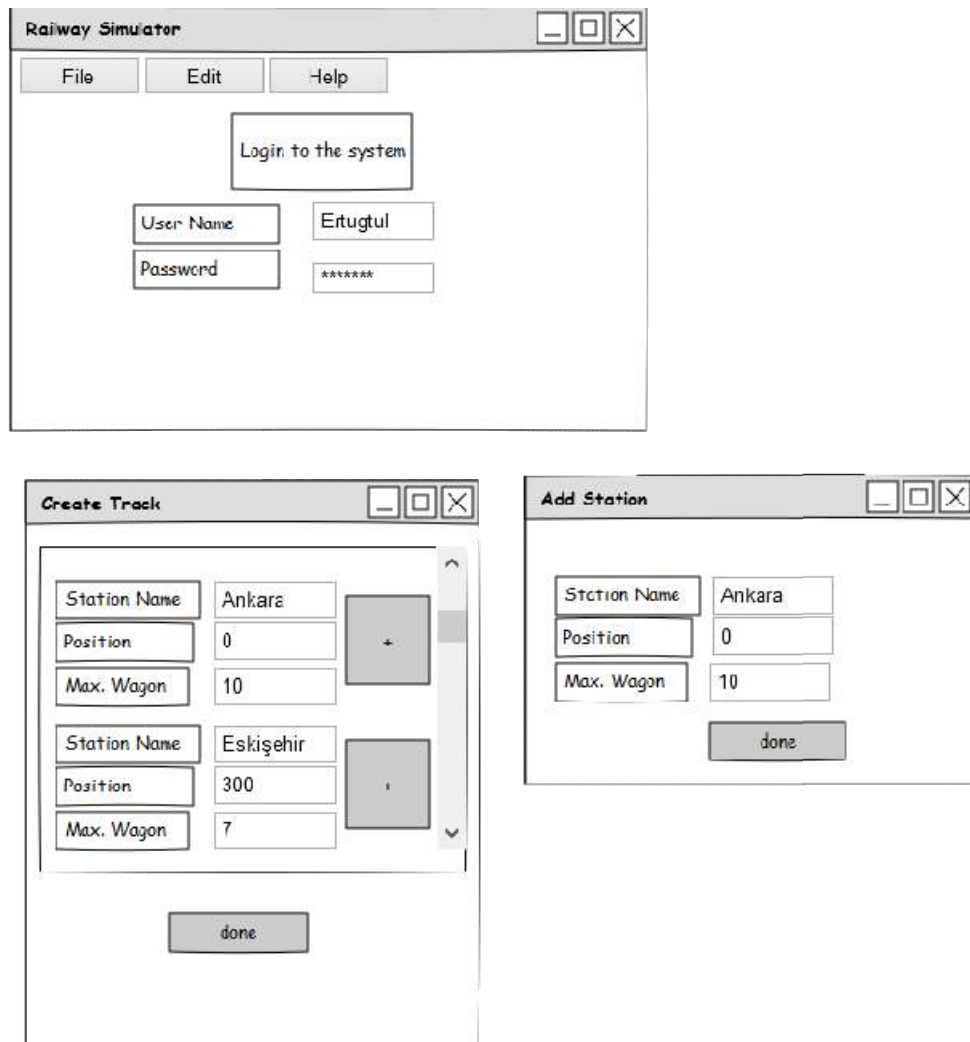


Figure 6: Activity Diagram

1.6 User Interface



The figure displays three mockups for a railway simulator's user interface. The first mockup, titled "Railway Simulator", shows a login window with a menu bar (File, Edit, Help), a "Login to the system" button, and input fields for "User Name" (containing "Ertugrul") and "Password" (containing "*****"). The second mockup, titled "Create Track", shows a list of stations with input fields for "Station Name", "Position", and "Max. Wagon". The first station is "Ankara" at position 0 with 10 wagons, and the second is "Eskişehir" at position 300 with 7 wagons. A "done" button is at the bottom. The third mockup, titled "Add Station", shows a single station entry with input fields for "Station Name" (Ankara), "Position" (0), and "Max. Wagon" (10), with a "done" button below.

Railway Simulator

File Edit Help

Login to the system

User Name Ertugrul

Password *****

Create Track

Station Name Ankara

Position 0

Max. Wagon 10

Station Name Eskişehir

Position 300

Max. Wagon 7

done

Add Station

Station Name Ankara

Position 0

Max. Wagon 10

done

Figure 7: Mockup for Log in, Add Station and Create Track Windows

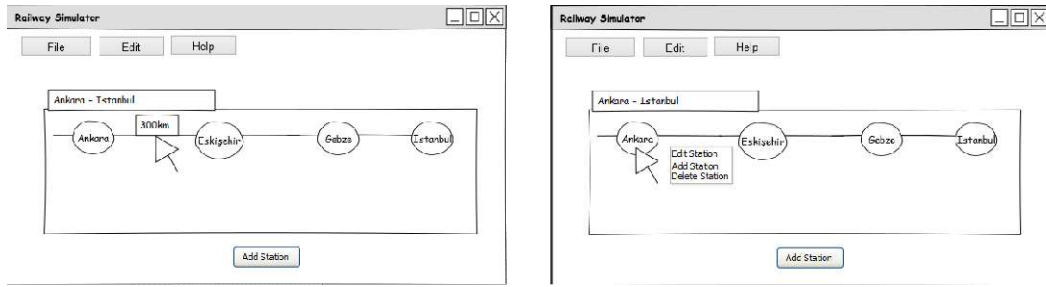


Figure 8: Mocukup for General View of the System

References

- [1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.
- [2] "OpenTrack Railway Technology." - Railway Simulation. Web. 23 Mar. 2015.
<http://www.opentrack.ch/opentrack/opentrack_e/opentrack_e.html#Events>.