Bilkent University

Department of Computer Engineering

CS 319 - Object Oriented Programming Term Project

RSim: Railway Simulator

Design Report

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

Instructor: Ertuğrul 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.

Table of Contents

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, analysis and design 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.

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.

Functional Requirements

There are main functionalities of the RSim. One of them is to design and model a railway system to be able to observe simulations on it. After the modelling is done, the user can simulate the behaviour of the system. The functional requirements for RSim can be organized in three main sections the requirements related to the sections. The list of all functional requirements are given below:

- Authentication
    - 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.
- Railway System Design
    - 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 can save the layout of the stations to a file.
- Simulation

- User is not allowed to modify the properties of trains and stations during simulation time.
- 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.

Non-functional Requirements

- All user inputs should be acknowledged within 1 second.

  The responsiveness of a system is important. Therefore we decided to include this non-functional requirement for user convenience.

- A system crash should not result in long term data loss.

  The progress of the activity of the user is important. We will introduce a saving machanism to achieve this goal.

- Any simulation calculation taking more than 1 minute will terminate and result in warning.

  Since the simulations can be quite complicated and the time interval that the user states can be too large, there will be a requirement that at maximum the time it takes to calculate the simulation process won't exceed 1 minute. Otherwise the simulation will terminate and the user will be warned about the situation.

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.

System Models

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.

Use-Case Model

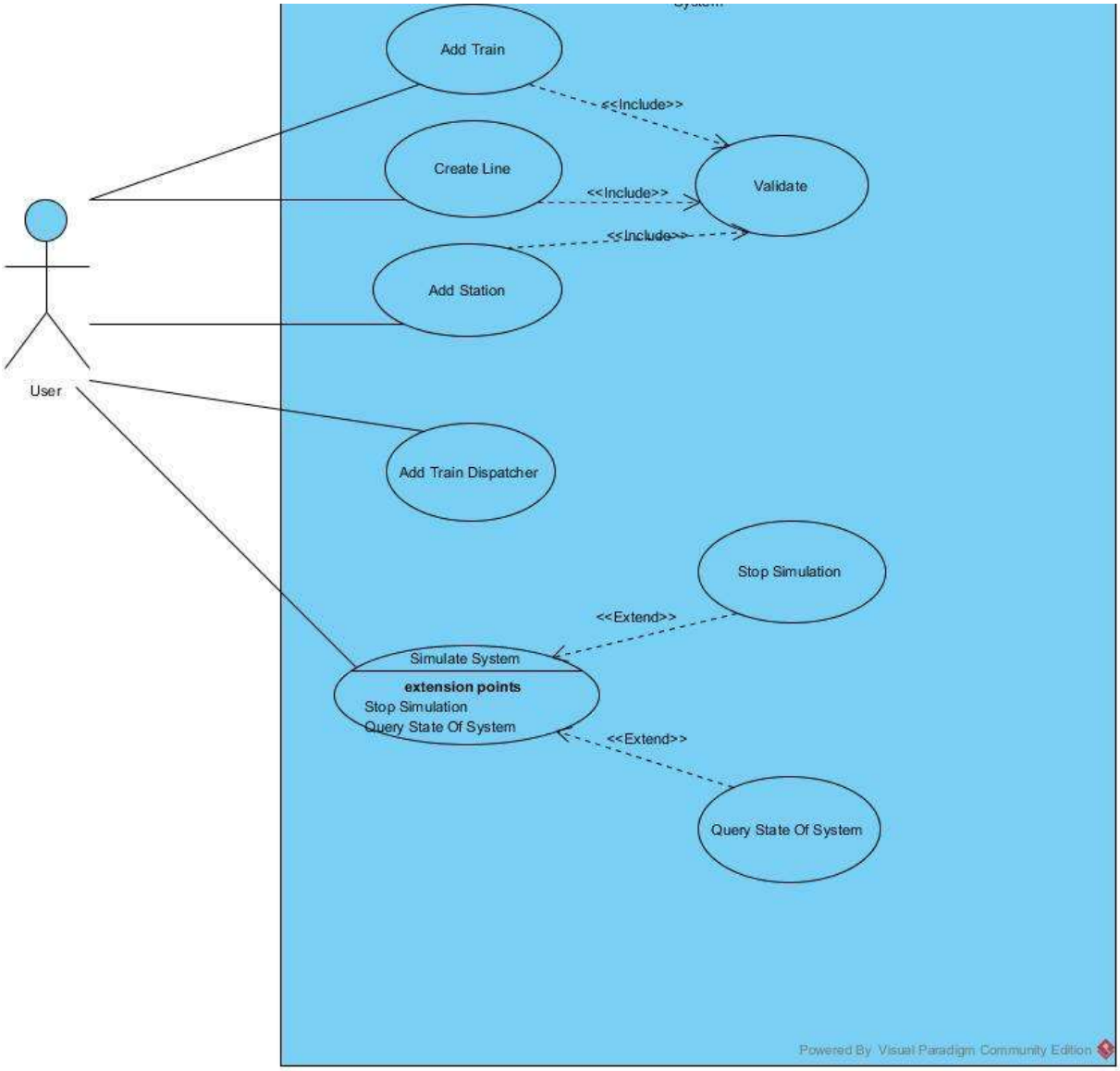

Figure 1: Use Case Diagram

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.

Object and Class Model

Class Model

The initial class diagram of the system is given below. The classes are organized in a way that they will be addressing the requirements for the system. Most of the classes in this stage are the the problem domain objects that are mostly the entity objects. There are no implementation specific objects in this stage. In the following stages of the projects implementation specific objects and classes might be added to the system.

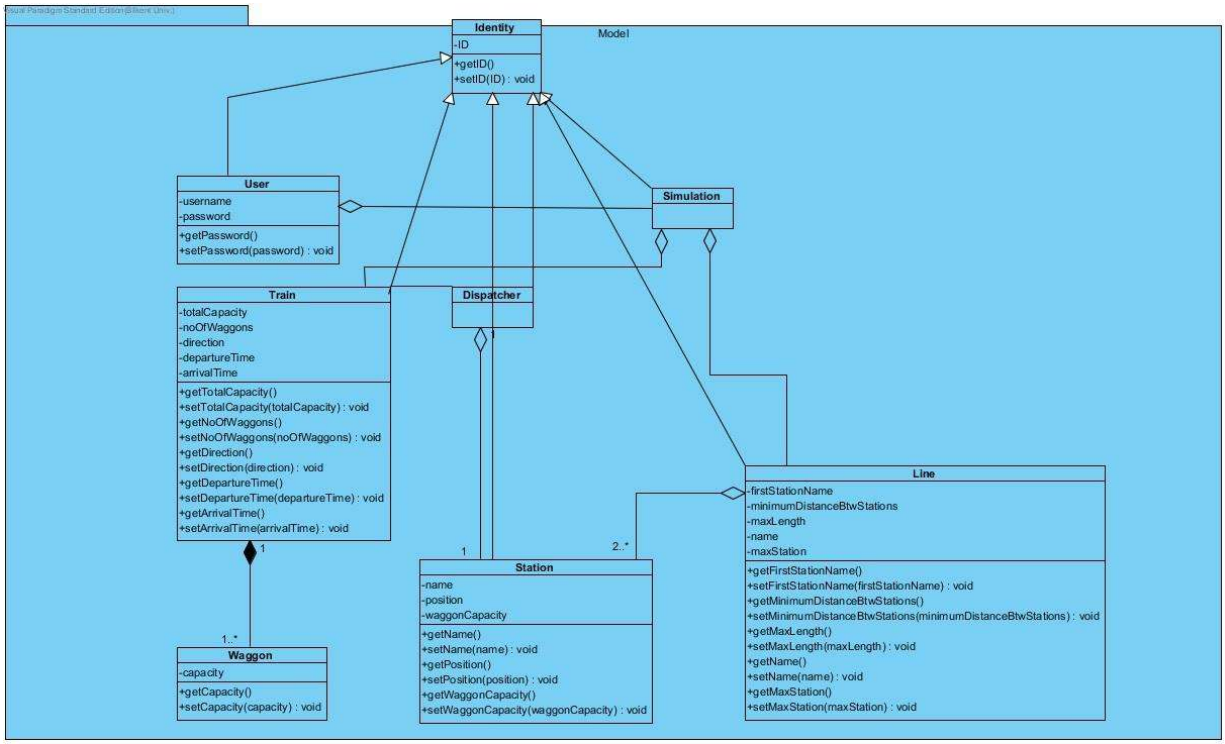

Figure 1 Class Diagram of Model Classes

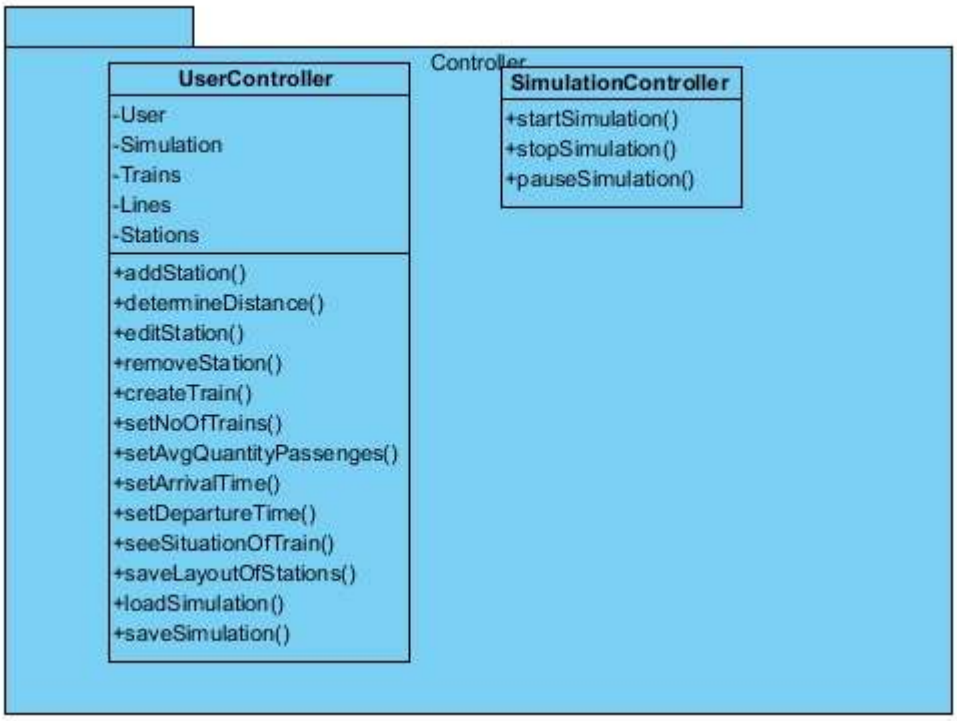

Figure 2 Class Diagram of Controller Classes

State Diagrams

New File Creation Diagram

Figure 3 represent how user can creates a new file. Simulator saves railway systems as a file. Firstly, user creates new railway system and s/he must add new station. Station is a major requirement for railway system. After adding new station, user can modify properties of station. If user wants to add more station, s/he can continue adding. However, if user is done with system design and s/he wants to simulate, s/he can run simulator and user can get results. After simulation, user have both choice which are user is allowed to edit system or save system.

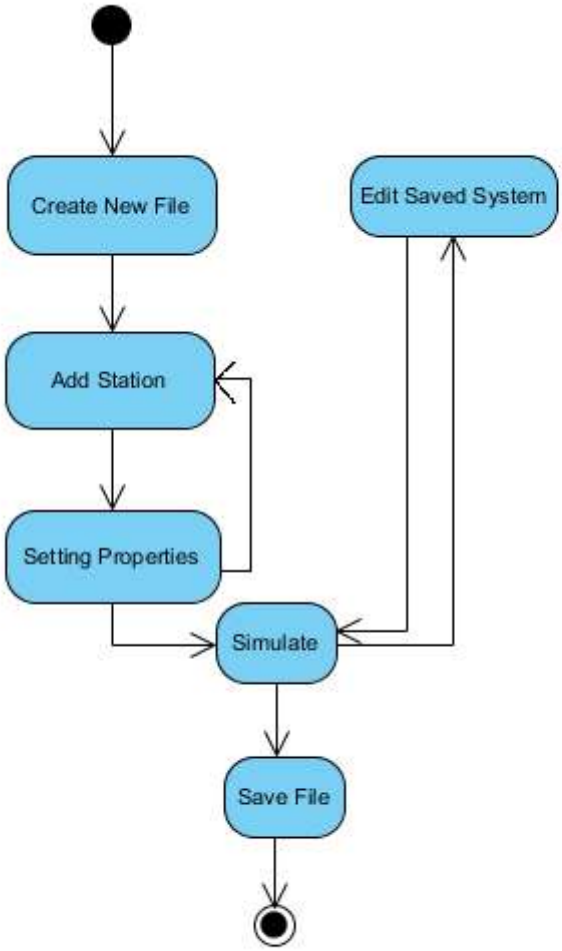

Figure 3 Create New File State Diagram

Open File Belonging Another User

In this railway simulator program, users are allowed to see another users' files. Even though they can see systems, they are not allowed to edit these system. They are only capable to see another users' systems. Initially, user must be login and s/he chooses existing file belonging another user. In Figure 4 this functionality's state diagram is given.

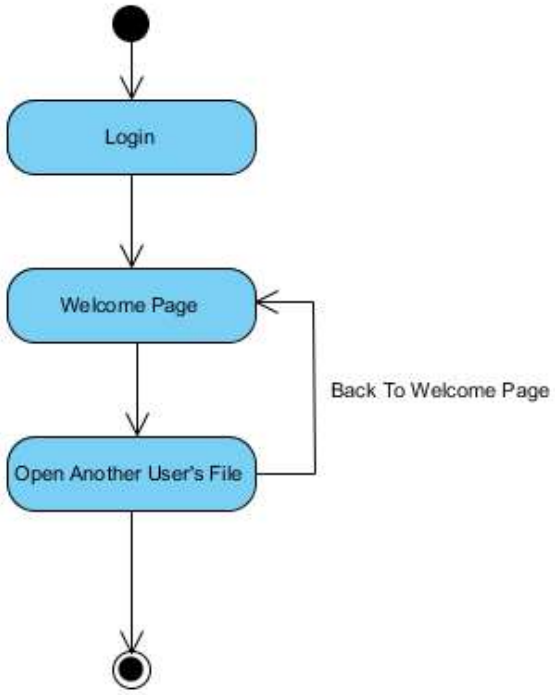

Figure 4 Open File Belonging to Other User State Diagram

Remove Desired Train

Figure 5 explains how user can remove desired train in system. At the system design stage, if user wants to remove one of the trains, s/he should reach the system that includes desired train. When the system is open, user can choose any train as his wish. Therefore, train is removed and simulator allows user to continue system design.

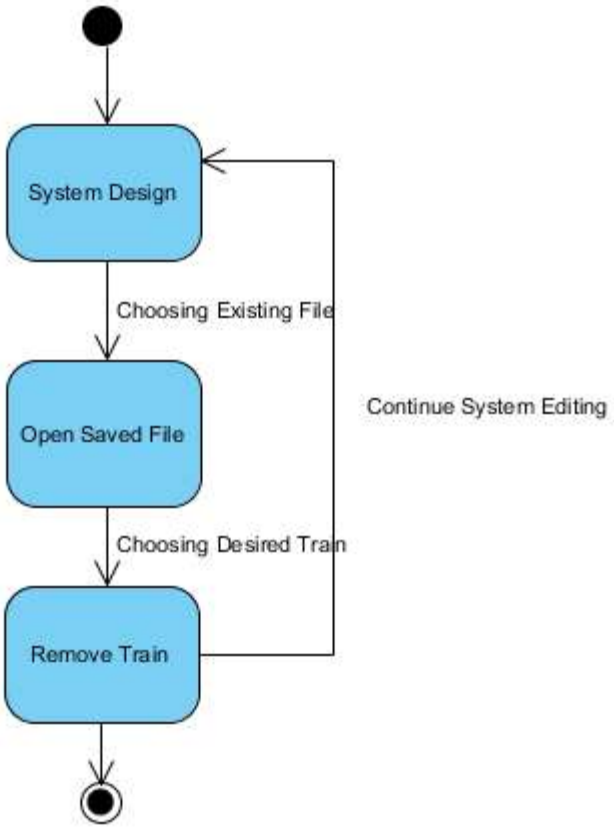

Figure 5 Remove Desired Train State Diagram

Dynamic Models

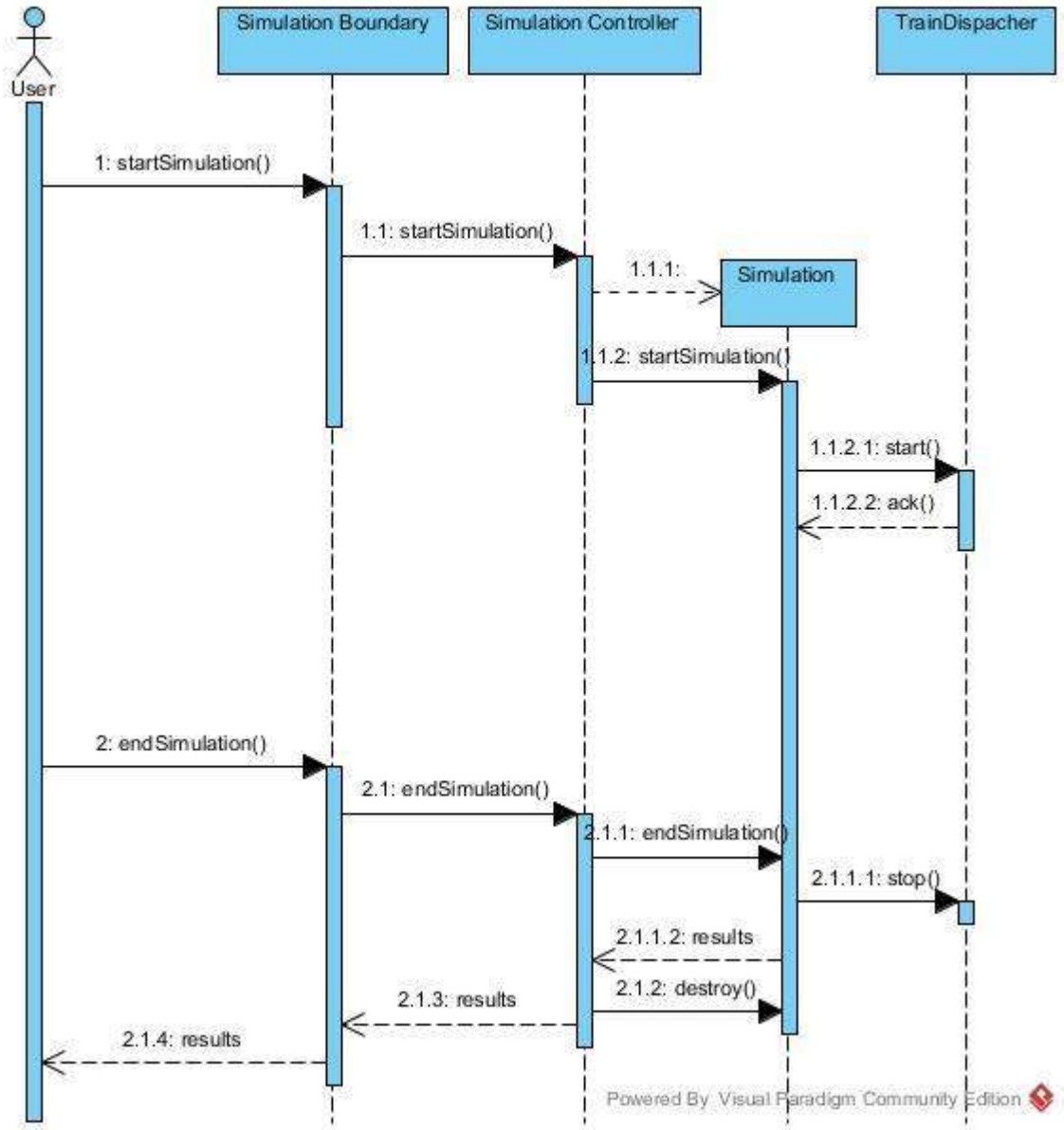

Figure 6 Sequence Diagram for Simulating the System

In Figure 6, 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.

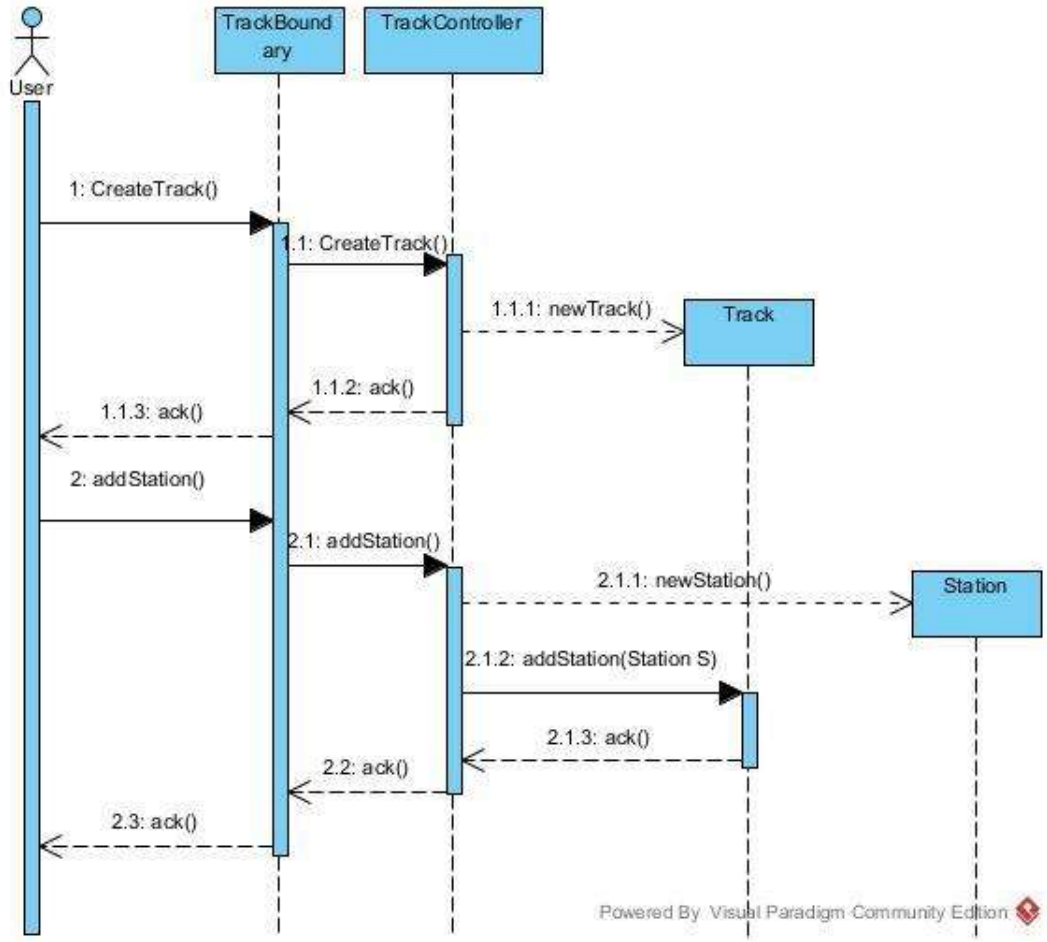

Figure 7 Sequence Diagram for Creating a Track

In Figure 7, 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.

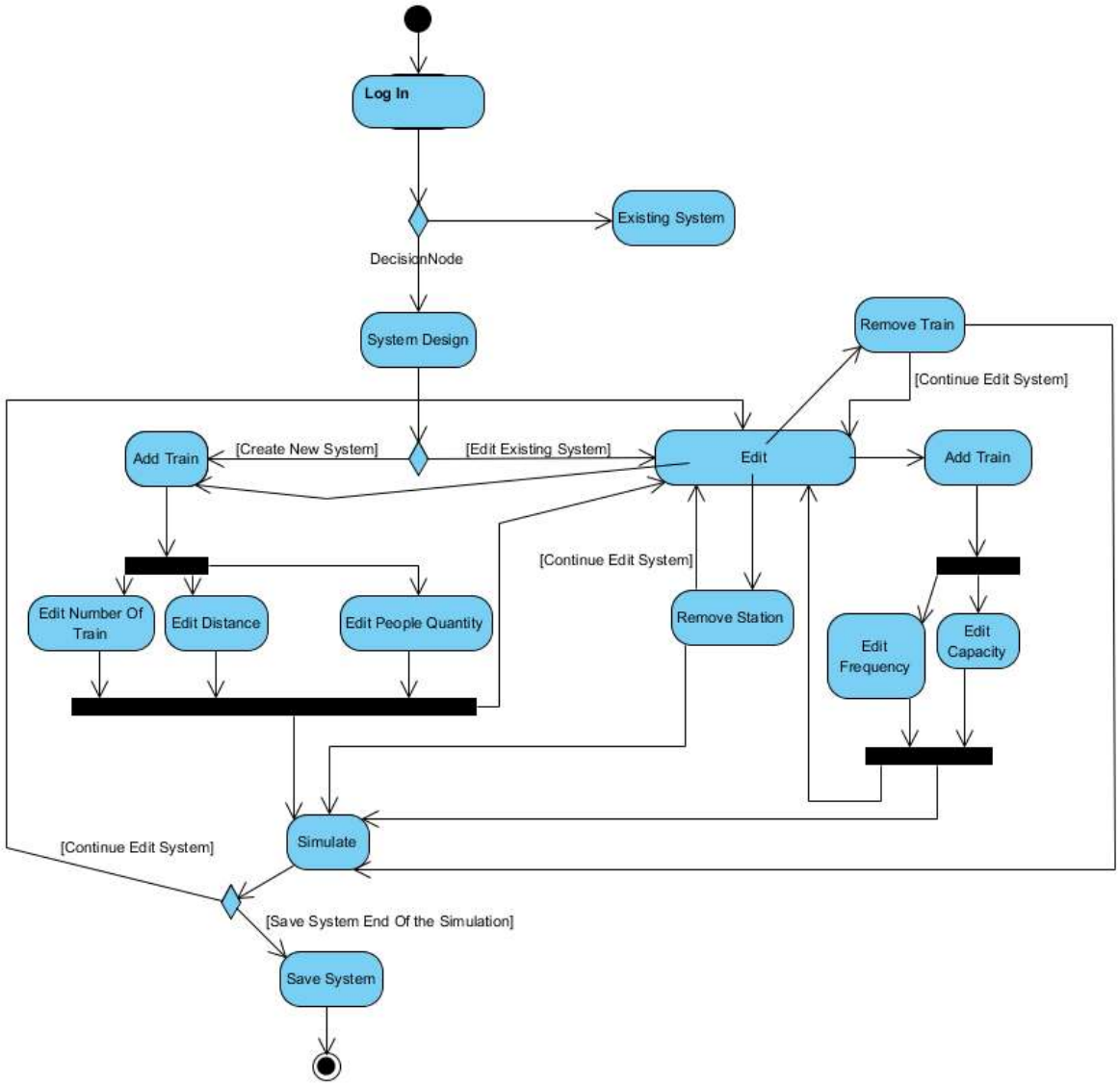

Figure 8 Activity Diagram

User Interface

User interface will be the secondary goal for the project to achieve. However, a simple design is given below in Figure 9 and Figure 10. Since the system requires authentication for a user, there should be a login screen as well as a new user sign up screen. For the functionalities that are required for RSim, there should be a create track dialog and an add station dialog. After the creation of the track and during the simulation time, there should be a general view of the system. In this stage, we consider the graphical interface as a secondary goal and therefore don't have concrete and comprehensive mock ups. During the implementation stage the look and feel of the system will be determining while developing the GUI parts in case we decide to use the GUI.

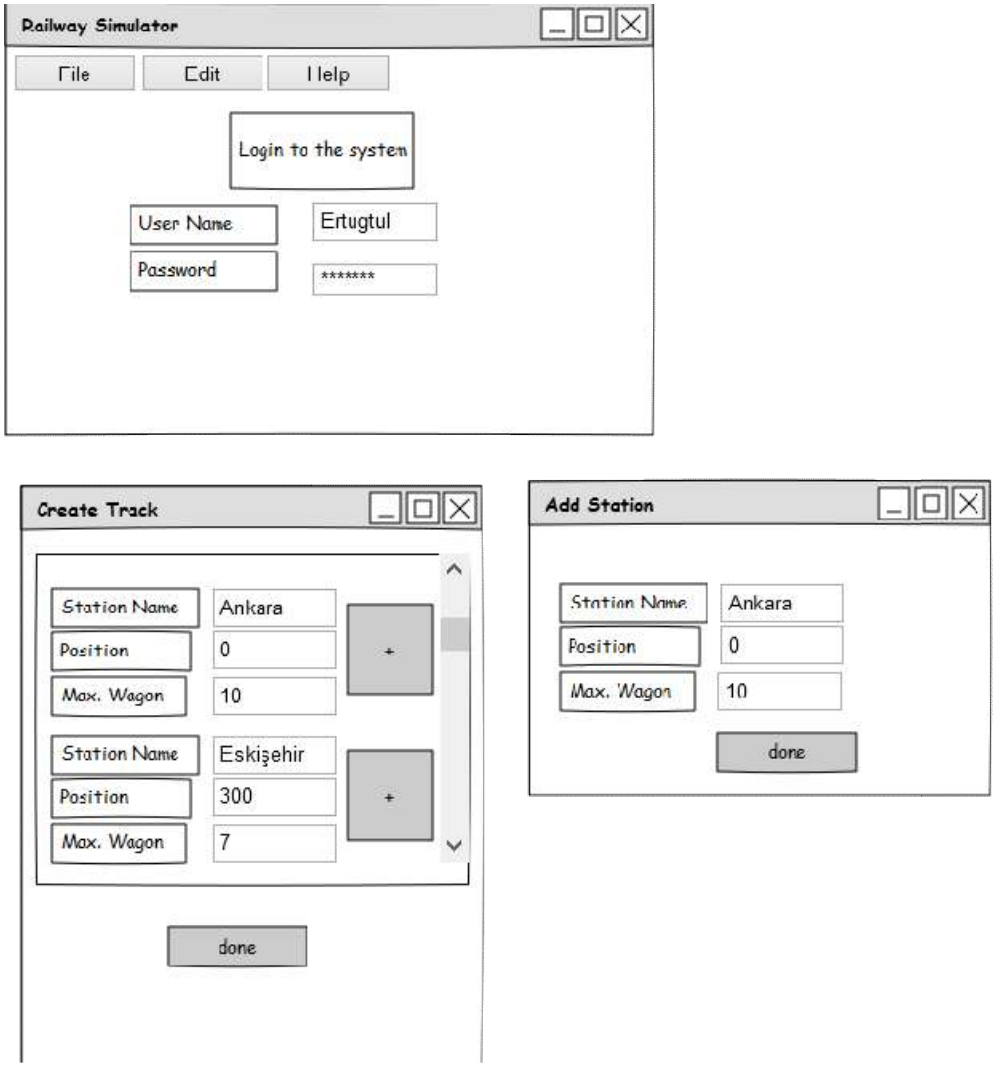

Figure 9 Mockup for login, add station and create track windows

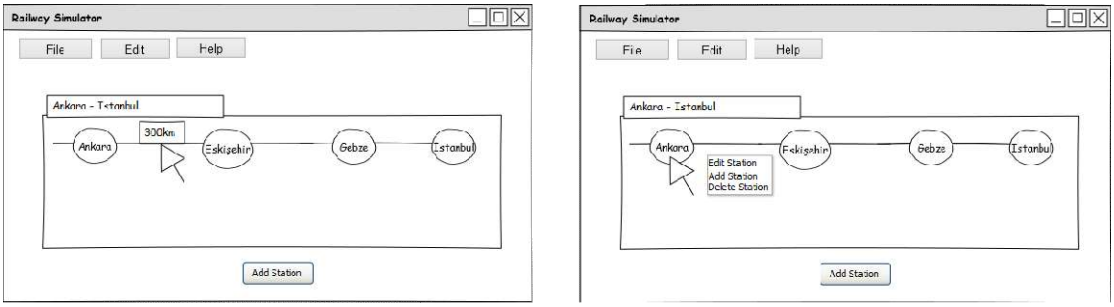

Figure 10 Mockup for General View of the System

Domain Lexicon

The problem domain of the RSim project is the railway systems. In railway system there are multiple tracks that serve as lines of transportation. There are trains that consist of waggons. Trains have their departure hours in a timetable. There might be delays in the trains so a train might be late. There are places that the trains stop and the passengers can enter to the train or leave the train. Those places are called stations. The stations can be included on only a single lines or multiple lines. The tracks and lines can be intersected at junctions. Junctions can be at a station or outside of a station. The lines can have single track as well as multiple tracks to allow two trains to go at the same line at the same time. The list of the words that are encountered on the problem domain are listed below:

- Train
- Railway
- Waggon
- Passenger
- Station
- Timetable
- Ticket
- Track
- Line
- Junction

Design

Design Goals

Describing design goals is the first and one of the most important parts of system design because it labels the qualities that our system should focus on. We determined our design goals as ease of use, good documentation, ease of learning, reliability, tradeoffs, robustness, portability and readability and defining them properly will help us to reduce complexity of the system.

Ease of Use& Ease of Learning :

Ease of usability and ease of learning are important key concept in terms of user's comfort. It is important for user to interact with the system easily without prior knowledge and external help. This design goals can be achieved by providing clear instructions and user documentation and simple user interface. For example in our railway simulation system we will build a simple GUI for adjusting parameters and showing the state of the system.

Good Documentation:

The documentation of the system should be organized well enough to make it easier for new developers to understand the system. Design and analysis reports, documentation of source code should be easily understandable by them to satisfy this design goal.

Reliability:

Reliability is one of the most important design goals. In order to satisfy relialibility requirements, system should have ability to run consistently and system crush shouldn't result in long term data loss, should adhere to boundary conditions, should define faults and tolerate them. Achieving this goal can only be possible by implementing testing process at every stage of development of the system.

Functionality:

In our project our main goal is to create non-complex railway simulation system so that user can interact with the system easily without any external help. Therefore we will try to avoid complicated functions in our system.

Robustness

Robustness is the ability of a system to handle invalid user inputs and unexpected situations. For example in our railway simulation system if user enters negtive number while specifying number of trains system should be able to handle this situation in order to satisfy this goal.

Portability

Portability is an important factor for a software because it provides usability of the same software in different environments. We determined that our system will be implemented in Java. Java Virtual Machine(JVM) plays an important role for portability because nowadays we have JVMs which are written almost for all platforms.

Readability

It is important for developers to understand the system from reading the code. If code is easy to read, it will be easy to understand which makes it easy to debug, maintain and extend.

Sub-System Decomposition

Considering the needs of the project we decided to use MVC architectural pattern. Following the MVC pattern, the subsystems are determined as model subsystem, controller subsystem and interface subsystem where the model subsystem will contain the entity objects of the system and the controller subsystem will modify the entity objects. On each change entity objects will notify the views associated with them and the view objects will update themselves. Interface subsystem will be also responsible for the user interaction where it will get the input from the user and trigger events in controller objects. In figureX the component diagram of the subsystem decomposition is given.

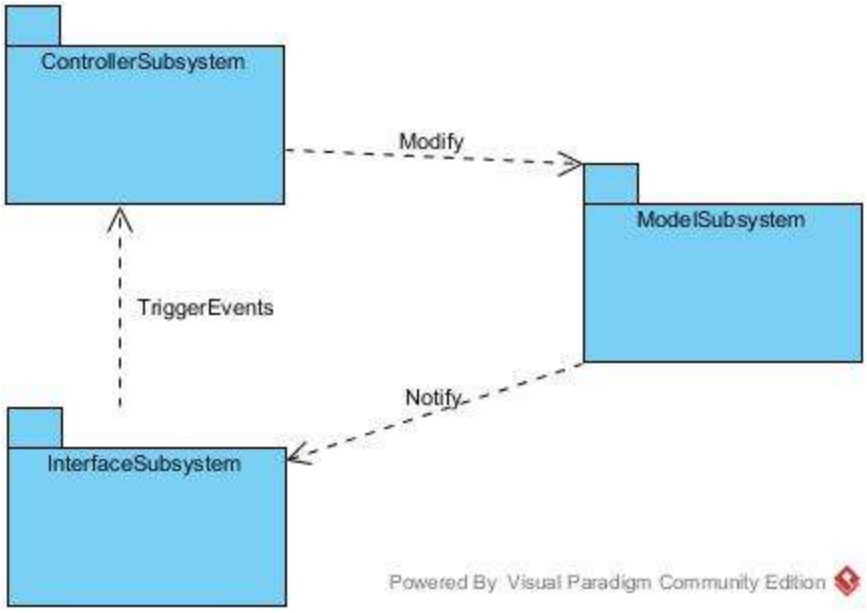

Figure 11 Component Diagram

Architectural Patterns

In order to manage with the complexity, we decided to use MVC architectural pattern. MVC (Model View Controller) architectural pattern is a strong architectural pattern where the entity objects are separated from the view objects and the responsibility for modifying the entity objects are given to the controller objects.

Inside the model subsystem, since we had many classes we decided to use meaningful packages for improving understandability of the subsystem. In Figure 12, redesigned class diagram is given.

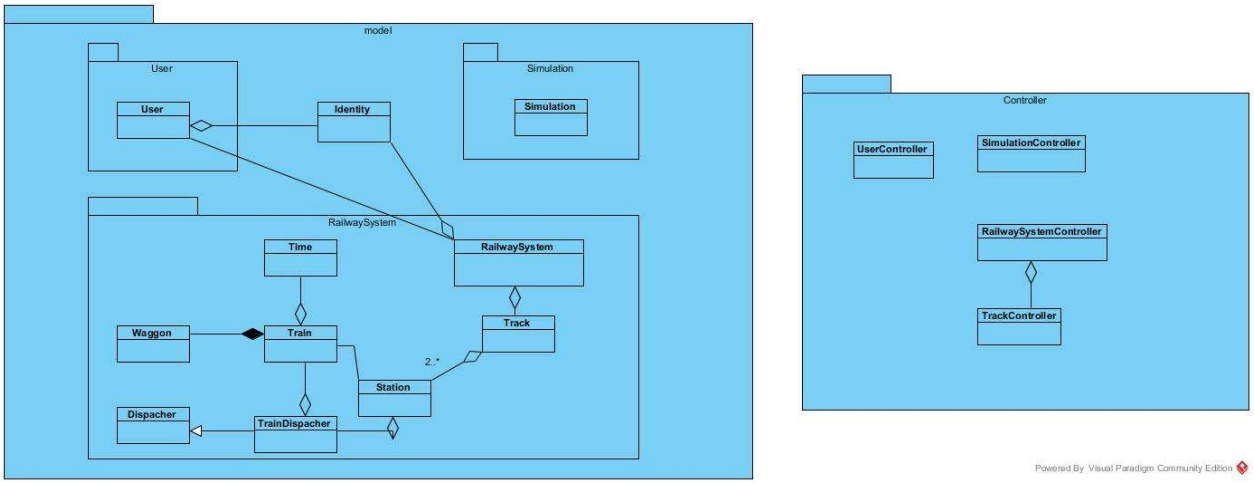

Figure 12 Class Diagram

Hardware/Software Mapping

Railway Simulator is designed to be working on a single node offline, hence there will be only one computer which the software will run on. With a decision, implementation language was chosen to be Java, hence the real software will run on a JVM which will run on top of the host machine. This way, the RSim software will be platform independent. In the Figure 13, deployment diagram is given.

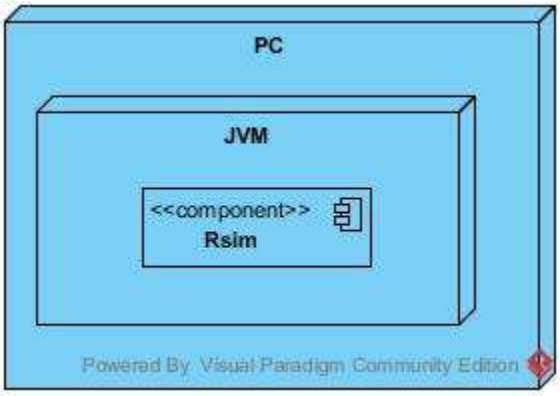

Figure 13 Deployment Diagram

Addressing Key Concerns

Persistent Data Management

In Railway System, the persistent data management will be done by object serialization. The Entity objects will be serializable and by object serialization tools that the programming language will provide; the state of the railway system, state of the simulations and all data that will need to be persistent will be stored on the disk with special file types to RSim. All of the settings and user records will be stored in a special directory that will be created on the user's home folder. On the first run of the program. Since the persistent storage functionality will be separated from other functionalities in the system migrating to another aproach such as using a database system will be easy for future developments.

Access Control and Security

In Railway System Simulator RSim, users will need to log in to the system to be able to simulate or modify the railway system design. Since the system will run on a local computer, there is not much security concerns. Users will be able to modify their own design and simulate it and assign permissions for other users. The permissions on the system will be similar to the operating system permissions. The permissions are listed below:

- Modify Design
- Read Design
- Simulate Design

Different than the operating system file permissions, these permissions won't be disjoint. For example the one who will be able to modify the design can also read the design and simulate the design. The one who is able to simulate the design can read the design but not able to modify the design. The one who is able to read the design will only be able to read the resign. The one who doesn't have the read permission won't be able to perform the other operations.

Global Software Control

In the RSim, event-driven global control flow is used. This is because, event-driven supports that our system’s flow is controlled by events, this global control flow mechanism is one of the most proper control mechanism for our project. In this simulator project, we decided to work with model view controller design patter. By using this pattern, it is provided that simulator waits for an event, which could be mouse click or keyboard inputs, updates the views. As a result, when an event occurs, detection will be made by a part and handling will be made by another part. User Interface subsystem gets the input from the user and delivers it to the Controller subsystem. Controller decides the event's type and changes it in the views related with the event. When event is occurred, different types of object decide action so that there is

no main controller. This kind of design is called as decentralized design and it allows distribution of dynamic behavior to objects.

Boundary Conditions

Initialization

- When a user wants to use our simulator s/he will encounter the login page. User needs to enter his own information which are password and user's name. After entering personal page, user can create new system, see another users' systems and edit existing system which is belongs to user. This is because simulator cannot allow user to edit another users' systems.

- Having chosen the creating new system, user will encounter "add station" page. User can decide properties of station and change them as his wish. After adding station system leads user to following functionalities: adding new train, removing station and removing trains. If user wants to add new train, he can edit train's properties such as capacity, frequency. If user wants to remove trains or station, it is required to decision that which train/station will be removed.

- When user chooses editing system, user open saved file and s/he can do followings: adding new station, new train or removing station and train.

- User simulates the system when s/he finishes design of system. User is supposed to enter time interval of simulation. Eventually, user can see its result and s/he can continue to edit system or save system as a file and then finish the program.

Termination

- User can simulate the system anytime and acquire the system's result.
- After simulation, simulator provides user that s/he can save the system as a file and finish simulation.
- After simulation, if user wants to continue working on simuator, simulator provides that and user can continue edititng the system.

Failure

If user wants to modify the railway system that belongs to an another user, simulator does not allows the user. This is because, users can only see other railway systems but they cannot edit another railway systems.

Conclusions

The railway simulator is a system that allows user to design railway system and simulate how it works. This simulator provides user to creates his own railway system and makes changing on the system. Additionally, user is capable to see other railway system that are designed by other users.

In particular this is system is developed in three major stages. First one is analysis process. In this process, we have described our problem and considered about solutions. At the beginning, we have defined functional and non-functional requirements. After requirements definition, we created class diagram according to classes which will be used in the project. Lastly, dynamic model of program is created which are sequence diagram, state charts and activity diagram. These diagrams help us to analysis project easily and shows that how system works, what kinds of functionalities are provided to user.

In this process of development of program, we have started design system and defined major behavior of the simulator. The system is consist of subsystems and we decomposed our system. It is decided that, simulator is applied model view controller design pattern so modularity of the system will be increased.

Consequently, in this level of development of program, we design major structure of simulator. Technical background of the system is defined deeply. According to analysis raport and this design process, we realized what kinds of problems we are possible to confront and we endeavored to solve them.

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>.