CHAPTER 4

ANALYSIS AND DESIGN

4.1 Introduction

This chapter presents an analysis and design of JaTraConSim. In the analysis section, both functional and the non-functional requirements are identified. The object-oriented analysis (OOA) is used in system analysis. In the design section, the object-oriented design (OOD) and some aspects of the user interface design are presented.

4.2 JaTraConSim Requirement Analysis

There are two main concerns of the JaTraConSim requirements: functional and non-functional requirements. Functional requirements describe the functionality of JaTraConSim. Meanwhile, non-functional requirements describe aspects such as usability, portability, maintainability and expandability.

4.2.1 JaTraConSim Functional Requirements

This section lists the five functional requirements that are used to develop JaTraConSim simulator:

- Implement a visual simulation of road traffic - Including road, car and traffic light models.
- Implement a visual road map design tool - Allowing the user to construct a valid road system from a set of components. Components must include Straight Road, T-Junction and Crossroad sections.
- Offer different green light length in second distributions for the simulation.
- Implement the graphing of the traffic flow – Graphs include Speed and Traffic Density and Travel Time.
- Offer different traffic density distributions for the simulation.

4.2.2 JaTraConSim Non-Functional Requirements

In developing any software application, focus is not only given to functional requirement but also to non-functional requirements. These non-functional requirements are described as follows.

1) Usability

JaTraConSim is built in a way that portrays its functionality, one main issue, which reflects the functionality behind software tools, is the user interface. Therefore, the user interface is designed in a way that mimics the real traffic flow. The simulator should be user friendly. It will enhance and support rather than limit or restrict the understanding of JaTraConSim. Human interfaces need to be intuitive and consistent with the user knowledge in order to let them gain some knowledge through the simulator.

2) Portability

JaTraConSim can run on multiple platforms since Java is an independent platform. Java supports three distinct types of portability (Roulo, 1997): source code portability, CPU
architecture portability and OS/GUI portability. These three types of portability are explained in detail in Section 5.1.1.

3) Maintainability and Expandability

JaTraConSim program is developed to be easily updated, maintained, expanded and upgraded according to business changes in the future because it is used an object-oriented approach.

4) Easy to Learn

JaTraConSim is easy to learn and use. This is because users can hover their mouse on the button for a few seconds and brief instructions will be shown to them.

4.3 JaTraConSim Design

JaTraConSim is designed using UML (Unified Modeling Language). This is because JaTraConSim is an object-oriented application and UML is the best suited to design an object-oriented development.

4.3.1 The Unified Modeling Language (UML)

UML is a graphical language with sets of rules and semantics. The rules and semantics of models are expressed in simple English. UML offers standard way to write the blueprints of software development. UML can be used for both visual programming and object-oriented programming.
Chapter 4

Analysis And Design

UML is a combination of three main methods, which are Object Modeling Technique (OMT) (Rumbaugh, 1991), Booch technique (Booch, 1994) and Object-Oriented Software Engineering (OOSE) (Jacobson, 1993).

According to Brahmi (1999) the primary goals in UML are as follows.

1. To provide users a ready to use visual modeling language so that they can develop and exchange meaningful models.

2. To provide extensibility and specialization mechanism to extend the core concepts.

3. To be independent of particular programming languages and development processes.

4. To provide a formal basis for understanding the modeling language.

5. To encourage the growth of the OO (Object-Oriented) tool market.

6. To support higher-level development concepts.

7. To integrate best practices and methodologies.

The UML defines nine standard graphical diagrams, which are class diagram, use-case diagram, object diagram, sequence diagram, collaboration diagram, statechart diagram, activity diagram, component diagram and deployment diagram. It is not necessary to use the entire nine diagrams in developing an application. Therefore, JaTraConSim only used six diagrams.

4.3.2 Identifying JaTraConSim Use-Cases

The use-cases for JaTraConSim are shown in Figure 4.1. There are three identified use cases.
Figure 4.1: Use-Case Diagram

Design details of each use case are shown as sequence diagram in Appendix A. Figure 4.1 shows the overall picture of JaTraConSim design. The main use cases are described below.

A user handles the role of an actor. It produces the road map by designing the road map in the design mode. It can selects and chooses active road segment, drag and place onto the design screen. When the design is finished, a user has to select the simulation tab to perform simulation. But before that, a user’s created road map will be validated. If it is invalid, an error message will be prompt out asking for a user to recreate again the road map until it is valid. A user can enter the simulation mode if the road map is valid. The flow detail of this process is shown as an activity diagram in Figure 4.2.
Users can perform simulation activity and generate graphs after successfully entered the simulation mode. For instance, they can generate graphs for the “Speed and density” and “Travel time” of the car. The flow detail of this process is shown as activity diagram in Figure 4.3.
4.3.3 Identifying Sequence Diagram

Based on functional requirement, three main use cases are identified. Each use case is described by using a scenario. Likewise use-case diagram, sequence diagrams are used to model scenarios in the systems. However, their respective scenarios and diagrams can be found in Appendix A.
4.3.4 Identifying Class Diagram

From the requirement statements and the above use-cases diagrams, the class diagram is depicted in Figure 4.2.

![Class Diagram](image-url)

Figure 4.4: Class Diagram

4.3.5 Identifying Activity Diagram

Respective activity diagram shows overall picture in Figure 4.2 and Figure 4.3 above.

4.3.6 Identifying Component Diagram

Respective component diagram shows the static relationships exist between the deployable software components is depicted in Figure 4.5. The component diagram shows JaTraConSim.exe, an executable file consists of four main classes:
Simulation.class, MapFrame.class, SplashScreen.class and GraphFrame.class. The executable file has been bundled into one jar file named as traffic.jar.

![Component Diagram](image)

**Figure 4.5: Component Diagram**

### 4.3.7 Identifying Deployment Diagram

Respective deployment diagram shows the configuration of the runtime elements of the application is shown in Figure 4.6.
4.3.8 Model Management

This section is to describe how individual class of JaTraConsim fit together to form the JaTraConSim package.

4.3.8.1 Package Diagram

Package is a grouping of model elements. A package may contain other packages, both subordinate packages and ordinary model elements. Package is represented as a folder, shown as a large rectangle with a tab attached to its upper left corner.

In this project, JaTraConSim is contained within a package called ‘traffic’ and has five separate files each with a different function. The respective package diagram can be found in Figure 4.7. The files held within traffic consist the following.

Simulation – The class file that holds the ‘main’ method. The main method calls a new instance of the SplashScreen class and the MapFrame class. It uses a thread to keep the
splash screen active for a set period of time before disposing of it, allowing the MapFrame instance to then become active.

**MapFrame** – The main and important part of JaTraConSim, containing both the designing and simulation modes of the program as well as the window the application appears in. One instance of this class is created on application start-up, this class then controls the running of the whole program.

**GraphFrame** – Each instance of this class holds specific data relating to a certain graph. The class displays this information in the form of a graph on a separate (external to MapFrame) window. MapFrame creates and disposes of these instances as and when necessary.

**SplashScreen** – This is a simple window that displays a graphic image while the MapFrame class is loading. It is created and disposed by the Simulation class.

**Images** – A sub-directory of the ‘traffic’ package used to store all the images files used by JaTraConSim simulator.
JaTraConSim User Interface Design

JaTraConSim is used a graphical user interface (GUI) which consists of the windows or dialog boxes and GUI controls, with which users interact, as well as the objects needed to manage or control the interfaces.

JaTraConSim has an event-driven environment likes Windows, an application performs processing in response to an event. For example, selecting from menu, clicking on the OK button with mouse pointer, and clicking on or uncovering a window are events that trigger a response from the application, whether it is to resize the window, update a record, or display a dialog box. Furthermore, in an event-driven application, simply assign event-handling procedures to events; then the procedures execute whenever the event takes place. The procedure should be truly generic and not depends on lots of assumption about the values of memory variables.
JaTraConSim GUI objects includes the following.

- **Menus.** Menus are the principal means of user input in a Windows application. A menu is a list of commands that users can view and choose from.

- **Dialog boxes.** A dialog box is a temporary window displayed to let the user supply more information for a command. A dialog box contains one or more GUI objects.

- **Bitmap.** A bitmap is a binary representation of graphic image in a program. Windows itself uses lots of bitmap graphics; for example, the images representing various controls on a typical window, such as scroll bar arrow. The control menu and the minimize symbols also are bitmap graphics. Each bit or group of bits, in the bitmap represents one pixel of the image.

- **Strings.** Strings contain text, such as descriptions, prompts, and error messages that is displayed as part of the Windows program. Because these text strings are Windows resources separate from the program, can be edit or translate messages displayed by a program to another language without having to make any changes to the program’s source code.

- **Fonts.** Windows programs use fonts to define the typeface, size and style of text. For example, a particular character that a program can display on screen or print on a printer might be 10-point Times Roman bold. In this case, the typeface is Times Roman, the size is 10 points and the style is bold.
4.4 Chapter Summary

JaTraConSim analysis and design are presented by incorporating UML notation using use-case diagram, sequence diagram, class diagram, activity diagram, component diagram, and deployment diagram. Both functional requirements and non-functional requirements are also essential in developing JaTraConSim simulator.