Implementation of a graphical prototype for a train driver advisory system

Xiaochong Tian
Abstract

Implementation of a graphical prototype for a train driver advisory system

Xiaochong Tian

In the past decade, Driver Advisory System (DAS) has become a very emerging topic. There are several Driver Advisory Systems existing in the market. Those systems’ goals are to increase punctuality and decrease energy consumption by implementing several algorithms. A study – “Solutions to the Problem of Inconsistent Plans in Railway Traffic Operation” which is conducted by Division of Visual Information and Interaction Department of Information Technology, Uppsala University. In this study, there is a part called train Driver Advisory System mainly focuses on human factors/usability. A project implemented a graphical prototype of a train driver advisory system for evaluation purposes. The difference between this DAS system with other DAS systems is to achieve human factors/usability better and support train drivers better.

In this thesis report, important theoretical aspects about human-machine interface design related with the train DAS graphic user-interface (GUI) design and the development methodology used in this project were described. All components of this train Driver Advisory System were described and requirements were analysed. The train DAS development process was described, especially the system architecture and how the system was implemented.
# Table of Contents

1 Introduction ......................................................................................................................... 3  
  1.1 Background .................................................................................................................... 3  
  1.2 Project objectives .......................................................................................................... 4  
  1.3 Delimitations ................................................................................................................. 5  
  1.4 Thesis Outline ................................................................................................................ 5  
2 Theoretical aspects and Methodology .................................................................................. 7  
  2.1 Theoretical aspects ....................................................................................................... 7  
  2.2 Methodology ................................................................................................................ 9  
3 Analysis .................................................................................................................................. 11  
  3.1 Human factors/usability Research in DAS ................................................................. 11  
  3.2 System Description ....................................................................................................... 11  
  3.3 Requirements specification ........................................................................................... 13  
  3.4 Technologies Description ............................................................................................... 15  
4 System Development ............................................................................................................ 18  
  4.1 Development tools and environment setting ............................................................. 18  
  4.2 System Design ............................................................................................................... 18  
  4.3 Implementation ............................................................................................................. 24  
5 Evaluation ........................................................................................................................... 28  
  5.1 Results .............................................................................................................................. 28  
  5.2 Evaluation of author’s work ............................................................................................ 29  
6 Conclusion and future work ................................................................................................. 30  
7 References ........................................................................................................................... 31  
8 Appendix: System Deployment and Demonstration of data generation .......................... 32  
  8.1 Tablet-side application deployment ............................................................................ 32  
  8.2 Server-side application deployment ............................................................................ 33  
  8.3 Demonstration of data generation ............................................................................. 38
1 Introduction

This chapter will describe the background, objectives and delimitations of the project and also list the thesis outline.

1.1 Background

In the past decade, Driver Advisory System (DAS) has become a very emerging topic. There are several train Driver Advisory Systems existing in the market. Those systems’ goals are to increase punctuality, make passenger comfortable and decrease energy consumption by implementing several algorithms.

A Dutch railway infrastructure manager ProRail developed a DAS named “RouteLint”. This DAS provided information to achieve efficient communication between train drivers and traffic controllers and improve punctuality, passenger comfort and energy consumption. The platform of RouteLint is PDAs.

“The discussed version of RouteLint does not include concrete driving advices; there is for instance no indication of recommended speed or advice for breaking or coasting. The included information is very limited and generates additional mental workload e.g. to calculate distance to other trains or to signals. It mainly shows if the next signal(s) can be expected to show stop or go, thus it supports decisions to reduce or increase speed. Still, these decisions can be contradicting to the traffic controllers’ intentions, since RouteLint does only display the current infrastructure state and not deviations of current planning from the time table.”[1]

FARE is a DAS following new railway traffic operation principles of the dense traffic in Switzerland. Those principles were introduced by Lüthi. FARE focused to prevent small deviation of ten seconds from schedule which could cause train conflicts.

“FARE has a connection to the real-time traffic plan and gives a well-founded decision support for driving speed. However, it does not show the current planning itself, i.e. train drivers have no information about the traffic situation, helping them to understand cause of a delay, or about planned arrival times for a train. Train drivers are supposed to strictly follow a schedule without much freedom for self-dependent planning.”[2]

CATO is a DAS developed by Transrail in Sweden. It aims to reduce energy consumption. The current real-time traffic plan, infrastructure information, and speed recommendations are combined in this system. “It combines information about the current real-time traffic plan, infrastructure information, and speed recommendations.”[3]

A study – “Solutions to the Problem of Inconsistent Plans in Railway Traffic Operation” was conducted by Division of Visual Information and Interaction Department of Information Technology, Uppsala University and funded by the EU FP7 project ‘Optimal Networks for Train Integration Management across Europe’ (ONTIME), FP7-SCP01-GA-2011-285243, and by the Swedish Transport Administration (Trafikverket). The figure 1.1 shows the architecture of solutions. There is DAS deployed on the train and acting as the connection between Train Driver and the real-time traffic plan (RTTP).
"The RTTP is the traffic controllers’ current plan, continuously updated through their human-machine interface (HMI). The RTTP is also transmitted to train driver advisory system (DAS), which gives the train driver detailed and updated information about the present traffic plane. At the same time, the DAS can send status information about the train and messages from the train driver to the traffic controllers’ HMI." [4]. It is system for train driver and traffic controller to make interactions.

1.2 Project objectives

The DAS in the project mainly focuses on human factors/usability and supports train drivers better. We would like to make interviews of train drivers and observe train drivers’ behaviour, evaluate the impact of such a DAS on the train driver’s work environment according to interview and observations results, and then adjust styles of the DAS following those results.

The project consisted of two main tasks. The first part is to implement a DAS-GUI prototype which is useful for experiments. This DAS-GUI should be with traffic information and driving advice running on an additional device (additional screen, computer or tablet). The implementation should make it simple to add further views to the GUI and modules to the DAS, adjust position, size and colour of text and other elements in the GUI, and add additional
scenarios to the DAS. The second part is to implement a traffic simulator module connected to a simulator. The DAS should receive traffic information such as train speed and position which was send from the simulator. It should be easy to adapt the GUI and DAS to some other simulators. The system will be evaluated by drivers’ interpretations of, reactions and thoughts on a certain type of advice and its design. A group of drivers would be invited to participate in a trial session. Interviews to drivers would also be involved during system goes on trial. We would evaluate how good/bad the advice its design was with observation and interviews.

An implemented DAS-GUI prototype needs to meet the requirements of DAS user interface example, there should be seven views in the graphical UI displaying different information such as speed advice, target point, traffic profile, and track information and so on, continuously writing technical documents,

An implemented traffic simulator module needs to meet the requirements of communicating with simulator, generating outputs and sending outputs to DAS, continuously writing technical documents,

Testing, debugging, evaluation and optimization, continuously writing testing, evaluation documents should be done during the entire development process,

Report according to technical testing, evaluation documents should be written.

When the above-mentioned objectives had been achieved, the idea of the project had fulfilled.

1.3 Delimitations

The DAS-GUI should run on android tablet. The implementation of a DAS-GUI should have higher priority to be done; the basic part of implementation of a traffic simulator module connected to a simulator should be done. If there is enough time, the implementation of traffic simulator module would be extended. If there is enough time and condition is available, the real evaluation from real train drivers would be conducted.

1.4 Thesis Outline

This section describes the thesis structure. There is a brief description of every chapter.

Chapter1 (Introduction) introduces the background of the project, the objectives of the project. It also describes the delimitations of the project

Chapter2 (Theoretical aspects and Methodology) describes important theoretical aspects about human-machine interface design and the methodology used in this project.

Chapter3 (Analysis) describes the research area in the project and analyses the entire
system and requirement specification; it also describes the technologies used in the system development.

**Chapter 4 (System Development)** describes the entire software development process.

**Chapter 5 (Evaluation)** describes all results we have achieved from the alpha users (DAS GUI designers and researchers related with the study) testing.

**Chapter 6 (Conclusion and future work)** describes the conclusion and discusses the future work.
2 Theoretical aspects and Methodology

In this chapter, important theoretical aspects about human-machine interface design will be described, and the software development methodology used in this project will also be described and the reason why the developer choose this methodology will be explained as well.

2.1 Theoretical aspects

This section describes the theoretical aspects which are applied in the DAS-GUI design.

2.2.1 Human-Computer Interaction

Human-Computer Interaction (HCI) is a computer science field including methodology, theory and practice research and development. The objectives of HCI are designing, construction, and evaluation computer-based interactive systems. Those systems are hardware, software, input/output devices, displays, training and documentation. Users can use those systems efficiently, effectively, safely and get satisfaction from using those systems. “HCI is cross-disciplinary in its conduct and multidisciplinary in its roots, drawing on – synthesizing and adapting from – several other fields, including human factors (e.g., the roots for task analysis and designing for human error in HCI), ergonomics (e.g., the roots for design of devices, workstations, and work environments), cognitive psychology (e.g., the roots for user modeling), behavioral psychology and psychometrics (e.g., the roots of user performance metrics), systems engineering (e.g., the roots for much pre-design analysis), and computer science (e.g., the roots for graphical interfaces, software tools, and issues of software architecture).” [6]

2.2.2 Usability

Usability is one quality factor and has many its own sub-factors such as fit for use, ease to learn, efficient for daily use and so on. “The entire field of HCI shares the single goal of achieving high usability for users of computer-based systems. Not fuzzy and vague as it is sometimes perceived, usability is tangible and can be quantified.”[7] Usability has six sub-factors:

1) Fit for use
2) Ease of learning
3) Task Efficiency
4) Ease of remembering
5) Subjective satisfaction
6) Understand ability

The first factor is adequate functionality. From factor 2 to 6 combine to ‘ease to use’ that
is the broad definition of usability. The first factor is also thought as usefulness. “For expert users, usefulness is perhaps the primary usability factor, the factor that provides immediate access and affordance to the functionality without getting in the way.”[8]. Nowadays, ‘ease to use’ becomes more and more important than earlier because more and more non-expert users are using computer system now.

2.2.3 Situation awareness

“Situation awareness (SA) is a theory that mainly deals with the information environment of operators. Operators who have SA are aware of the state of a system and the environment; they are able to project the future state. This is a prerequisite for influencing states of the system in accordance with their goals. Endsley defines three levels of SA:

1. Perception: Operators perceive the important information. This means first, that the needed information is available and second, that attention is directed to its important pieces. Without attention to the right information, operators have no chance to understand and control a system.

2. Comprehension: This level describes ability to understand the perceived information. It is the process of interpreting information towards one’s goals and to remember relevant parts of information.

3. Projection: If the important information is perceived and comprehended, it is possible to understand the dynamics of a system. This will allow projecting future states and supports decision-making in order to influence and control the state of a system.”[9]

2.2.4 User centred design

“‘User-centered design’ (UCD) is a broad term to describe design processes in which end-users influence how a design takes shape. It is both a broad philosophy and variety of methods. There is a spectrum of ways in which users are involved in UCD but the important concept is that users are involved one way or another. For example, some types of UCD consult users about their needs and involve them at specific times during the design process; typically during requirements gathering and usability testing. At the opposite end of the spectrum there are UCD methods in which users have a deep impact on the design by being involved as partners with designers throughout the design process.”[10]

There are many advantages and disadvantages of user centered design.
### Advantages

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>More efficient, effective and safe in result</td>
<td>Cost more money</td>
</tr>
<tr>
<td>Helpful in satisfaction level and managing users’ expectation of the result</td>
<td>Cost more time</td>
</tr>
<tr>
<td>Users have the ownership of the result</td>
<td>Need more design team member</td>
</tr>
<tr>
<td>Less design and easily integrated into the environment</td>
<td>More difficult to translate some types of data into design</td>
</tr>
<tr>
<td>More creative designs in cooperation</td>
<td>Too specific for the general use and cost more to transfer to other more users</td>
</tr>
</tbody>
</table>

### 2.2 Methodology

According to the history of software, the first piece of software was arguably created by Ada Lovelace in the 19th century, for the planned analytical engine. But the software development methodology framework actually emerged with the software engineering which was defined by two consecutive meeting to overcome software crisis. The oldest software development methodology was provided from D.Parnas in 1972 for software maintainability and reliability. Nowadays, there are various software development methodologies available for developers such as waterfall development, prototyping, incremental development, spiral development, iterative development, rapid application development, agile development, code and fix, lightweight methodologies and so on.

It is very important to select the most suitable software development methodology. The most suitable methodology will make the software development process more efficient.

According to the requirements and characters of this project and based on the compared analysis of different software development methodology. Scrum was selected for the below reasons:

- One of the project objectives is to develop a runnable prototype and only one developer was involved in. Some early stage methodologies such as waterfall and prototyping are not applicable for this project, since they are suitable for large projects and for more developers involved in. Therefore, some nimble methodologies are more adaptive for this project such as rapid application development, extreme programming and scrum.

- Another reason of choosing scrum is the development period. The whole project was divided into two parts: GUI development and Server-side part development. The GUI part was divided into several views. Scrum was used in every view and every part.
Therefore, Scrum was used in the development process. Scrum is a kind of software development process. It is iterative and incremental. Generally it was used for agile software development. It includes a series of practice and pre-defined process framework. The main roles are scrum master worked as project manager, product owner represented as owner of profit, development team including all developers. Although scrum was developed for software development administration, it was also suitable for software maintenance team or known as a method of plan administration.

In every sprint which was defined by development team, generally it is a period from 15 days to 30 days; development team would release available incremental software. The implemented characteristics come from product backlog in every sprint; product backlog is a series of summary requirements according to priority. What product backlog should be implemented in a sprint was decided in sprint planning meeting. In the sprint planning meeting, product owner tells the development team what product backlog should be implemented, development team promise what product backlog could be implemented in next sprint. In every sprint, the requirements are frozen; nobody could change the sprint backlog.

![Figure 2.1 the Scrum process](image)

A sprint is the basic unit of the development process in scrum. As mentioned above, the project was divided into GUI part and server-side part and the GUI part was divided into several views. So every views and server-side part are matching to every sprint. Each sprint starts by the planning meeting and ends by sprint review meeting where the progress and result of this sprint were reviewed and plan of next sprint was made. Every sprint lasts one week or several weeks according to the tasks of every part.
3 Analysis

This chapter will describe the research area in the project and analyse the whole system, this chapter will also describe the technologies used in the system development.

3.1 Human factors/usability Research in DAS

There are several train driver advisory systems existing in the market for different purposes. CATO from Transrail in Sweden aims to achieve energy-efficiency and on-time performance. RouteLint from ProRail in Dutch aims to communication-efficiency and punctuality-improvement. FARE from Lüthi in Switzerland aims to prevent small deviations of ten seconds from the schedule. Human factors/usability research in DAS has been sparse.

Our goal is to design the DAS-GUI following human factors and usability principles such as user centered design and situation awareness, implement it and evaluate it from real train drivers. We hope to use this DAS-GUI to compare the usability from other train driver advisory systems and support train drivers better.

3.2 System Description

This section will describe the DAS-GUI on the android tablet-side app and the web application on the server-side.

3.2.1 Tablet-side GUI

![Figure 3.1 DAS-GUI](image-url)

Figure 3.1 DAS-GUI
The GUI on tablet-side was designed by researchers from Division of Visual Information and Interaction Department of Information Technology, Uppsala University. User centered design was involved in the whole design process. Train drivers were invited to attend to DAS-GUI design. Their thoughts about what information should be displayed, what symbol should be used to highlight information and what color of every curve should be were collected and used in the DAS-GUI design.

The whole GUI could be divided into five views. Each view will be described below:

The view in the left-top corner is the speed view. This speed view includes the real-time actual speed of the train, and advice speed sent out by train plan system or train plan simulator, the difference between the real-time actual speed and advice speed, the current time, the time when the train will arrive the next station, next station name and the distance between the train position and next station. There is also a speedometer displaying the difference between the real-time actual speed and the advice speed. The arrow in the speedometer represents how much different between those two speeds and tells the train driver whether to accelerate or not to follow the advice speed. Situation awareness was implemented in this view. Train drivers were continuously provided with the advice speed information and the real-time actual speed information, be able to understand the behavior of the system, and finally to predict the future behavior of the system.

The view in the center-top is the general information view; this part displays the static information. In this project, the information could be the same as the information in the figure 3.1.

The view in the right-top corner is the target point view. This target point view includes target point information. This information is about the train in station such as the station name, the train should stop or pass by in the station, the track number which the train running on or stopping on in station, which platform the train should stop by, whether there is another train will meet, the time point when train should arrive the station before or after, the time point when train should leave the station. If the train past the target point, the target point should not show anymore.

The view in the center is the running view. This running view includes the altitude curve which represents the altitude of the train track, the real-time actual speed curve which represents a history piece of the real-time actual speed, the advice speed curve which represents a history piece of the advice speed and a future piece of the advice speed, the limit speed line, the length of the train and the current position of the train. This view is the most important view of the entire GUI. Situation awareness was also implemented in this view. Train drivers were continuously provided the height information and speed information, they would be able to understand the relationship between speed information and height information then predict whether to accelerate the train or not to avoid brake.

The view in the bottom is the track view. This track view includes the track information
such as how many tracks there are, the station information such as how many tracks there are in station, which tracks the train is running on and the position of the train. This part is also very important.

3.2.2 Server-side application

There is no traffic simulator involved in this project. So the server-side application should generate the fake data for demonstration and implement the communication feature. The fake data for demonstration also should be stored in database for repeatable used.

3.3 Requirements specification

This section will analyze the non-functional requirements and the functional requirements.

3.3.1 Non-functional requirements

- The system is based on client server model.
- The client works on android tablet with android version 3.2 (primarily).
- The client works on other kind of android version (may be in the future).
- The server should run on windows operation system.

3.3.2 Functional requirements

Functional requirements include following parts.

Part1 time scope
1, Demonstration need run for 10 minutes, it means that the period is from 1st second to 600th second

Part 2 Communication between DAS-Server and DAS-tablet
1, DAS-Server and DAS-tablet should communicate every second
2, DAS-tablet sends DAS-server the current time such as it is 1st second right now.
3, DAS-server receives the current time such as it is 1st second right now, it returns to DAS-tablet only the current position of the train in the 1st second.

Part 3 map scope
1, we assume that the highest speed of the train is 180km/h, so in 10 minutes the train will run out to 30km away at most.
2, we need to make fake height data for 30km. There should be a data point for every 25m, so we will have 1200 height data points.

Part 4 DAS-Server
1, the fake height data should be stored in database table in DAS-Server, at the first time when DAS-tablet communicate with DAS-server, DAS-server will send all height data to DAS-tablet
2. There should be a scenario generator in DAS-Server. We assume that the speed of train in the 1st second is 0 and it starts from 0 km position, we divide the whole 600 seconds to 12 parts, and every part is 50 seconds. Then we can input different accelerated speed in every part. The scenario generator will generate the position of the train in every second and those position data will be stored in database paired with time such as 1st, 0m; 2nd, 2m. Before demonstration, we can generate a new scenario or use the scenario which already stored in the database.

Part 4 DAS-Tablet
1. There should be a function in the app to calculate the actual speed by the position data which sent out by DAS-server by below formula. p represents the position, t represents the time.

\[
\text{Actual speed } = \frac{\Delta p}{\Delta t}
\]

Part5 Target point
The demo should have several target points, at least, one is a station named Pc, and another is a station named Kva.
The demo will last 10 minutes. So we assume that the train will pass the Pc station 5 minutes later after the demo starts and will stop at the Kva station 10 minutes later after the demo starts.
The information about target point should include parts below
1. A Signal which is F(it means the train should pass the station) or S (it means the train should stop at the station)
2. Name of the station(Pc,Kva)
3. Track number(the train should run on which track when it pass or stop at the station)
4. Platform mark(the train should stop at which side of the platform, it is H in Pc station)
5. Another train number that the train will meet at the station
6. Time point(the train should pass the station no more than a time point, the train should stop at the station no less than a time point and leave the station after a time point)

Part6 Advice speed
The advice speed should be predefined and stored in database.
The advice speed follows blow rules:
1. Do not validate the speed limit
2. Make sure the train will reach the target point in time
3. Try to achieve energy efficient
3.4 Technologies Description

Android

Android is the most popular and widely used mobile operation system right now. It is developed by Google which is one of greatest companies in 21th century. It is a kind of free and open-source code operation system based on the Linux kernel. It is mainly used for mobile device such as smart mobile phone and tablet computers. The main character of those mobile devices is with a touchable screen which is used as input/output user interface. Recently this touchable screen expands to other kind devices such as televisions OS named as Android TV, Android Auto which is running in cars, and Android Wear which is running in watches, and Android Glass. There are suitable versions of Android running on those devices.

Compared with Windows Phone and iOS, android is an open source mobile operation system. The source code of Android is released by Google under open source license. There is original Android UI developed by Google and many other versions of UI based on this open source Android such as TouchWiz developed by Samsung, HTC Sense developed by HTC, MotoBlur developed by Motorala, MIUI developed by Xiaomi and etc.

AChartEngine

AChartEngine is a framework for chart in Android system. Current released version is 1.0.0. The lowest version of Android SDK supported by AChartEngine is 1.6. In late 2008, there was no open-source library for charting/graphing/plotting libraries. It was developed by Dan Dromereschi. First version was 0.2.0 released in March 2009. It was the first open-source charting library in Android system based on Android SDK version 1.1.

Right now it could be used to do following type chart: area chart, line chart, time chart, scatter chart, bar chart, pie chart, bubble chart, doughnut chart, rang (high-low) bar chart, dial chart/gauge, cubic line chart and combined chart by above charts.

MySQL

MySQL is an open-source relational database management system (RDBMS). It was originally developed by MySQL AB which is a Swedish company. Right now it belongs to Oracle. The MySQL community server is still an open-source database management system. But the MySQL enterprise edition is the paid version with 30 days free trail. From the name of this database software, it can be found that it uses Structured Query Language (SQL). SQL is the most commonly used standard language which is used to access database.

Compared with other large databases such as Oracle, DB2, SQL Server, MySQL has its
owner weaknesses, but it is still a very popular database usually used for web application. Generally for personal users and small and medium sized enterprises, the features provided by MySQL could meet their requirements. Because MySQL is open-source software, they could decrease their cost by using MySQL. Linux works as operation system, Apache and Nginx works as web server, MySQL works as database, PHP/Perl/Python works as script interpreter, they composed to open source web application software stack named as LAMP.

**Apache Tomcat**

Apache Tomcat actually could be divided into two parts. Apache is web server; Tomcat is application server, it is an extension from Apache. Apache and Tomcat could be used as independent web server, but Apache could not explain java application such as Java Server Page (JSP), servlet. They are containers, but publish different contents. Apache is a html container like IIS by Microsoft; Tomcat is a JSP/servlet container like websphere by IBM, weblogic by Oracle.

Apache Tomcat is open-source software developed by Jakarta project belongs to Apache software foundation. It follows the technical specifications provided by Oracle, implements to support servlet and JSP and as web server provides some special feature such as tomcat administration, console, and security domain administration and so on.

**Web Service**

Web service is defined by W3C as a web application which is platform-independent, loose coupling, self-contained and programmable. It is a powerful programming language independent concept for data communication over a network. SOAP (Simple Object Access Protocol) is such a protocol used in web service. It is based on XML and HTTP, the kind of specification for exchanging structured information of web services in a network. It relies on XML for its message format and relies on HTTP for its message transmission. WSDL (Web Service Description Language) is a XML based language that provides a model for describe web service.
Figure 3.2 Web Service structure

The figure 3.2 shows the basic structure of the web service.
4 System Development

In this chapter, the entire software development process will be described. It starts from the methodology, and then presents the development tools and environment, describes the system design and explains how the system implemented.

4.1 Development tools and environment setting

Selecting the most proper development tools and setting up the efficient development environment is a very important task in the software development process. One aim of this project is to develop an android tablet app. Windows and Java SE environment are needed for the tablet-side app development. And in the server-side, Java EE environment, Web application server and database are required.

The development tools were chose and environment was set up by following steps:

1. Using which Java IDE with android development plugin really depends on the developer. But ADT (Android Development Tool) plugin with Eclipse is highly recommended, because ADT plugin with Eclipse is the most reliable Java IDE for android now. It makes the development environment setting up very quick and easy.

2. In the android app, there will be several kinds of charts should be generated. So an android chart library should be used or developer needs to develop his/her owner android chart library. Developing his/her owner android chart library is huge work. Using other android chart library is a better option. Of course, the library should be open source. AChartEngine was chose here because it is the android charting library highly recommended by google.

3. In the server-side, Eclipse was chose for tablet-side app development, so it is also chose for Java EE environment. Apache tomcat is chose as web application server and MySQL is chose as database because they are open source and the most popular software nowadays. The mysql-connector library should also be installed for Apache tomcat server connecting with MySQL database.

4. MySQL WorkBench should be installed for MySQL database testing and administration.

4.2 System Design

In this section, the details of system architecture will be described. There are the whole system architecture, design of the tablet-side and server-side. The database design is also included.
4.2.1 System Architecture

The whole system is based on client/server model. The application installed in the tablet works as a client. The web application deployed in windows works as a server. The figure below shows the overview of the whole system architecture.

![System Architecture Diagram]

The communication between client and server is web service. The web service was deployed in the server-side.

4.2.2 Tablet-side design

This section describes the tablet-side design including how client architecture was designed, how data interface was designed and the relationship between all classes. The design pattern was also described in the section, and it also explains why this design pattern was chose.

4.2.2.1 Client architecture

The tablet-side mainly is the GUI displaying information to train driver. But it also needs to communicate with the server to get information to display and control how the information refreshed and displayed in GUI.

Design pattern was involved in tablet-side because pattern could make source codes more
Driver Advisory System

understandable and works more efficient. The model-view-controller (MVC) design pattern is used in tablet-side, since it makes the coding manner more extensible and readable. It is much better than programming without MVC pattern. There are three layers in this design pattern:

- Model Layer: it consists of objects that represent the data in the application. For example, representing the speed, target point, running information to the model.
- View Layer: it presents layout and appearance of the system user interface, such as application's windows, views, and buttons and so on. Additionally, the view objects need to present notifications coming from the events.
- Controller Layer: it is a bridge between the model and view layer. It receives the notification from view layer and processes the action according to the data from model layer.

![Figure 4.2 Client Architecture](image)

The client architecture is not complicated; it shows from the figure above. It was divides into three layers.

View Layer: is responsible to display different information such as speed, station, and target point. Those parts composed the whole GUI.

Chart Generate Layer: it includes static chart and dynamic chart; it is the bridge between data and GUI, it generated different part according to their characters.
Basic Layer: it includes abstract chart, data interface, and system parameter. Data interface will communicate with the server to get data; abstract chart creates the general information for the view layer.

### 4.2.2.2 Data Stream Design

This section describes the data stream design in the tablet-side. It describes how data comes from the server.

![Data Stream Design Diagram](chart.png)

The figure 4.3 shows how data stream was designed. The data interface is the key part of the whole data stream. It can pull running data from the server; it also can receive running data from the server which pushes running data to data interface. When data interface has data, it will pass the data needed by the chart generate layer to generate the GUI.

#### 4.2.2.3 Classes design

This section describes the relationship between all classes and how they work together.
Android app starts up from main activity. From the figure 4.4, it shows that main activity is the main entrance of the whole app. The GUI was divides into four charts: Speed Chart, Station Chart, Running Chart and Target Chart. Those four charts match four classes to compose the whole GUI. System parameter is the class which control the configuration. The Chart generator class is the generator to create four parts. The DataInterface class gets running data from the server and then sends all data to the DataUtil class to generate the data needed by the Chart Generator. IGraChart and AbstractGraChart work for the Chart Generator to control and create the basic components of every view.

4.2.3 server-side design

This section describes the server-side design including how server architecture was designed and the database design.

4.2.3.1 Server architecture

The server-side is much simpler than the tablet-side because it only generates fake data for demonstration, input the fake data to database and get fake data to database to communicate with the tablet-side, let the tablet-side app to have the information which
needed to display on the GUI, also some configuration data also stored in database which is also needed to transfer to the tablet-side. The figure 4.5 is the server-side web application architecture.

![Server-side Architecture Diagram](image)

The fake data generation is one part of the server-side. It should be input through web navigator. The jsp files define items which should be input. DBManager class does the proper calculation and then stores the data to the database for repeatable use.

The server-table communication is another part of the server-side. It gets the fake running data from the database and communicates with the tablet-side to let the tablet to have the enough information to display on GUI. LoadMessage class does this work to get data from database and achieves http feature.

### 4.2.3.2 Database Design

This section describes the database design phase. This phase determined what kind of data needs to be stored in the database and find out the relation between them. This data also needs to be present as objects in the "Data model". In other words, the altitude information must be stored in the database. Information that should be stored includes the id as primary key, position, altitude. The station information must also be stored into database including id as primary key, station name, station position, number of track, track of the train, stop or not, arrive time, leave time and etc. The configuration data such as color of every curve should also be stored in database. The fake data about train running also should be stored in database. The fake data about limit speed also should be
stored in database. The fake data about advice speed also should be stored in database. The figure 4.6 shows the entire database diagram.

![Database Diagram]

4.3 Implementation

This section will present implementation details which are extended after the design phase. The implementation of table-side app and server-side web application will be described separately.

4.3.1 Implementation in tablet-side

This section will describe the implementation in tablet-side by paragraphs. Firstly it is the overview of the whole implementation in tablet-side, and then it will focus on the main view of the tablet-side GUI.
4.3.1.1 Overview of the implementation in tablet-side

There are four packages which the source codes in tablet-side were implemented into: basic, das, element, and util. The figure 4.7 shows the contents and the structure of those packages.

<table>
<thead>
<tr>
<th>Basic</th>
<th>Das</th>
<th>Element</th>
<th>Util</th>
</tr>
</thead>
<tbody>
<tr>
<td>AbstractGraChart</td>
<td>MainActivity</td>
<td>RunningChart</td>
<td>DataUtil</td>
</tr>
<tr>
<td>IGraChart</td>
<td></td>
<td>StationChart</td>
<td>HttpUtil</td>
</tr>
<tr>
<td>DigitalClock</td>
<td></td>
<td></td>
<td>StationUtil</td>
</tr>
</tbody>
</table>

Figure 4.7 tablet-side app packages

Below every package will be described briefly:

- The basic package contains the basic and important classes of the tablet-side app such as “AbstractGraChart” and “IGraChart” classes generate the basic components of the GUI. The “DigitalClock” class extends the system clock to achieve the display requirement of the time.

- The das package contains the “MainActivity” class which is the main entrance of the Android app. It extends from the “ActionBarActivity” class to compose the basic layout of the whole GUI.

- The element package contains the important view classes of the whole GUI – “RunningChart” and “StationChart” classes. The “RunningChart” class control and generate the altitude, actual speed, advice speed, limit speed information which is the main part of the whole GUI. The “StationChart” class control and generate the train track information and station information.

- The Util package contains the data generation utilities. The “HttpGet” class gets data from server and passes data to the “DataUtil” class. The “DataUtil” class does data modelling work to translate the data from database to the data displayed in RunningChart view. The “StationUtil” translate the data from database to the data displayed in StationChart view.

4.3.1.2 Running Chart View

The running chart view is the most important view in the whole GUI. It displays the altitude information, train length information, actual speed information, advice speed
Driver Advisory System

information, limit speed information. That information is dynamic and real-time following the train running. From this view, train drivers can easily know where the train is, the altitude of the train, the train is up slope or down slope, what’s the actual speed now and the speed suggested by the system and whether the train is above the limit speed or not.

It extends from the “AbstractGraChart” class and uses the AChartEngine framework to generate the altitude curve, actual speed curve, advice speed curve, limit speed line and train length curve.

4.3.1.3 Station Chart View

The station chart view is also an important view in the whole GUI. It displays train track information and station information. From this view, train driver can easily know which track the train is running on, how far the next station is, which track should the train run on in the next station.

It extends from the “AbstractGraChart” class and uses the AChartEngine framework to generate the track line.

4.3.2 Implementation in server-side

This section will describe the implementation in the server-side. The implementation in the server-side is much easier than the tablet-side because it only achieve very simple function. The figure 4.8 shows the contents and the structure of those packages.

![Server-side web application](image)

There are two functions, one is choosing color profile, and another is defining position. Two jsp file “ColorProfile.jsp” and PositionDefine.jsp” matching each function to define the static web content. Two classes “DefineColorProfile” and “DefinePosition” to handler the dynamic content of those two functions.
Driver Advisory System

The “DBManager” class handles the data input and output from the database and also handles the database connection as well.

The “LoadMessage” class provides the web service from which the app in the tablet-side can get the data stored in the database.
5 Evaluation

This chapter presents all results we have achieved from the alpha users (DAS GUI designers and researchers related with the study) testing.

5.1 Results

According to alpha users (DAS GUI designers and researchers related with the study) testing, the following results have been achieved.

DAS-GUI (Train Driver Advisory System-Graphic User Interface) has been implemented as Android tablet app. The actual speed of the train, the advice speed of the train, the difference between those two speeds has been displays in the top-left view, the information of next station such as planned arrive time, distance to the next station, system time and name of the next station also have been displayed in this view. General information has been displayed in the top-centre view. The target point information has been displayed in the top-right view. That information was displayed by scroll view. Every line is the information of one target point also known as station. Every line includes name of the station, the signal about the train should pass by the station or stop by the station, which track the train should run in the station, which platform the train should pass by or stop, the arrive time and left time. The running profile has been displayed in the centre view. Height curve, advice speed curve, actual speed curve, limit speed line, train line have been implemented as designed. Colours of those curves or lines could be modified. The track information has been displayed in the bottom view. In this view track information such as track number, track information in every station, train running on which track and train position have been implemented.

The traffic simulator module has not been implemented because the train traffic simulator was not involved. A server-side web application was implemented to provide fake traffic data which works as simulator module. The functions such as generation of fake demonstration actual speed data for ten minutes has been implemented, selection of curves’ and lines’ colours have been implemented as designed.

Client/Server model has been achieved that android tablet app receives data from server-side web application to show information. Communication between android tablet app and server-side web application happens every second to get real-time data.

Alpha users (DAS GUI designers and researchers related with the study) testing has been done. Modifying, debugging and optimization which are following alpha users’ feedback has been done.

The evaluation by train drivers’ interpretation has not been done. According to objectives, a group of drivers should be invited to participate in a trial session. Those drivers should
Driver Advisory System

be invited by the Division of Visual Information and Interaction, but now it is not ready. So this evaluation by train drivers’ interpretation should be done later. Observations and interviews to train drivers also should be done when it is ready.

5.2 Evaluation of author’s work

In the project, the author studied the UI design theories and the android app development, did the requirements analysis of the Driver Advisory System Graphic UI and the traffic simulator module including non-functional requirements and functional requirements analysis, chose and deployed the suitable development environments and technologies, chose and use the suitable development methodology, chose the open-source android chart library, open-source web application server tomcat apache, open-source database product MySQL to achieve the server-side web application, designed the whole client/server architecture, android tablet app architecture and the server-side web application architecture and database table schema, implemented the whole system, did testing, debugging and optimizing of the whole system and wrote all technical documents of the whole system.
6 Conclusion and future work

At the beginning of this project, the author had very little experience in software development, very little knowledge in train user interface and very poor knowledge in train traffic plan and train operation model. Understanding the basic concepts of Java API and android environment helps the author to develop an android tablet application by using APIs provided either from android library and other public resources. To get an overview of the Apache Tomcat project was a theoretical support to help understanding on the basic ideas of android communication and AChartEngine library. Researches on the different das systems developed for business was indeed a powerful technical support on related works. In the software development process, the software development methodology was applied. At the end, from system design and system demonstration results, the prototype system for evaluation purposes had been development and most requirements of this project had been achieved.

We have gained a lot of project experience especially in software development and understood the related theory from books more fundamentally. What we learned from the software development process is not only the coding skills, but other kinds of skills in software development work such as requirement analysis, architecture design and also some soft skills like communication with people, how to cooperate with others etc.

The prototype system is definitely not perfect. There are some shortcomings need to be improved. The following issues should be considered as future work:

- GUI details, some GUI details were not implemented as exactly as designed because of the library and other technical reasons. Those details should be improved in the future work.

- Charting library, the AChartEngine is a general android charting library for many purposes. In the future work, we should develop our own library for this app, it should be support the GUI better.

- Connection with Train Traffic simulator, in this project, the train traffic simulator was not involved. If the train traffic simulator involved, the GUI will display the real train running data. It will be better for evaluation purpose.
7 References


8 Appendix: System Deployment and Demonstration of data generation

In this chapter, the system deployment will be described. The table-side app deployment and the server-side web application deployment will be described separately. The evaluation also will be described in this chapter. It will explain how to generate the fake position data and how to choose the colour plan.

8.1 Tablet-side application deployment

To deploy the table-side application on the real android tablet, the first step is to pack the source code to an installable file with apk extension name. If the application was developed in ADT (android development tool plugin for Eclipse), the steps can be quite easy as below:

1. Find the android project in the project folder in ADT, it is usually showed in the package explorer on the left side.

2. Right click on the root directory of the project, and then choose the “Android Tools”-> “Export Unsigned Application Package” from the pop-up menu.

3. Click on “Export Unsigned Application Package” and then get a window to choose where to file will be located.
4. Click on “Save” and wait for exporting. And then there will be a file with the same name of the project and the extension name apk.

5. Copy the apk file to the android tablet and double click the file icon, it will be installed in the android tablet.

8.2 Server-side application deployment

To deploy the server-side application is a little bit complicated than tablet-side. The web application was deployed in Apache Tomcat and will access the MySQL database to get data. So MySQL database and Apache Tomcat should be installed firstly and then deploy the web application in the Apache Tomcat.

8.2.1 MySQL database installation

Follow the below steps to install MySQL database:

1. Double click “mysql-5.1.73-winx64.msi” icon to start installation.
2. Click “next” and use the default options to install MySQL.
3. Click “Finish” to complete the installation.

8.2.2 Tomcat Apache 7.0 installation

Follow the below steps to install Tomcat Apache:

1. Double click “apache-tomcat-7.0.27.exe” icon to start installation.
2. Click “next” and use the default options to install Apache Tomcat.

8.2.3 Application deployment

To deploy the application in Apache Tomcat server, the first step to pack the source code to war file. If the application was developed in Eclipse IDE, the steps can be quite easy as below:

1. Find the android project in the project folder in Eclipse, it is usually showed in the project explorer on the left side.

2. Right click on the root directory of the project, and then choose the “Export”-> “WAR file” from the pop-up menu.
Driver Advisory System

3. Click on “WAR file” and then get a window to choose where to file will be located.
4. Click on “Finish” and wait for exporting. And then there will be a file with the same name of the project and the extension name war.

5. Copy the war file to the Apache Tomcat installation directory such as “C:\Tomcat 7.0\webapps”, when the apache tomcat server starts, the war file will be unpacked to a directory with the same name and deployed in apache tomcat server.

8.3 Demonstration of data generation

8.3.1 Position Data generation

This page ‘http://192.168.1.101:8080/CATOWeb/PositionDefine.jsp’ could be used to generate the position data for evaluation. The figure 8.8 shows what could be input to generate the position data. From the figure, it can be found that we can generate 10 minutes position data for evaluation. An initial speed could be input by the first item. And then acceleration could be input for every 50 seconds. An advice speed offset also could be input to generate the advice speed from the actual speed. Click the ‘submit’ button to generate the position data and store them into database.

8.3.2 Color plan option

This page ‘http://192.168.1.101:8080/CATOWeb/ColorProfile.jsp’ could be used to
choose different color plan. The figure 8.9 shows how could be chose to have different color plan. From the figure, there are two color profiles now. It could add more color profiles by modifying the configuration of this page. Choose one color profile and click the ‘submit’ button to store the color plan into database. It could be achieved that different curve color combination.

Figure 8.9 color profile page