Electronic Data Capture

Bachelor of Technology

End of Semester Report

Siyu Zeng
1255777
szen010

This report summarizes the BTech project achievements in semester one, introduces the project principles and discusses related topics and problems encountered. As a milestone, it also looks ahead in the future for possible improvements.

Table of Contents

Introduction

The appearance of Internet and rapid development of mobile applications have greatly affected people's everyday lives. The various roles of mobile devices have made them not only a platform for communication, but also a tool for entertainment and business. While traditional ways of information sharing are virtualized in many mobile applications, data processing has been a major task for them.

This report introduces the final year BTech project of electronic data processing. It firstly starts with an overview of the project. Afterwards, it discusses the requirements in detail. Through this several concepts related to the project are introduced. Then we describe our design principles and alternatives. At last, we conclude the achievements and explore possible improvements for remaining part of the project.

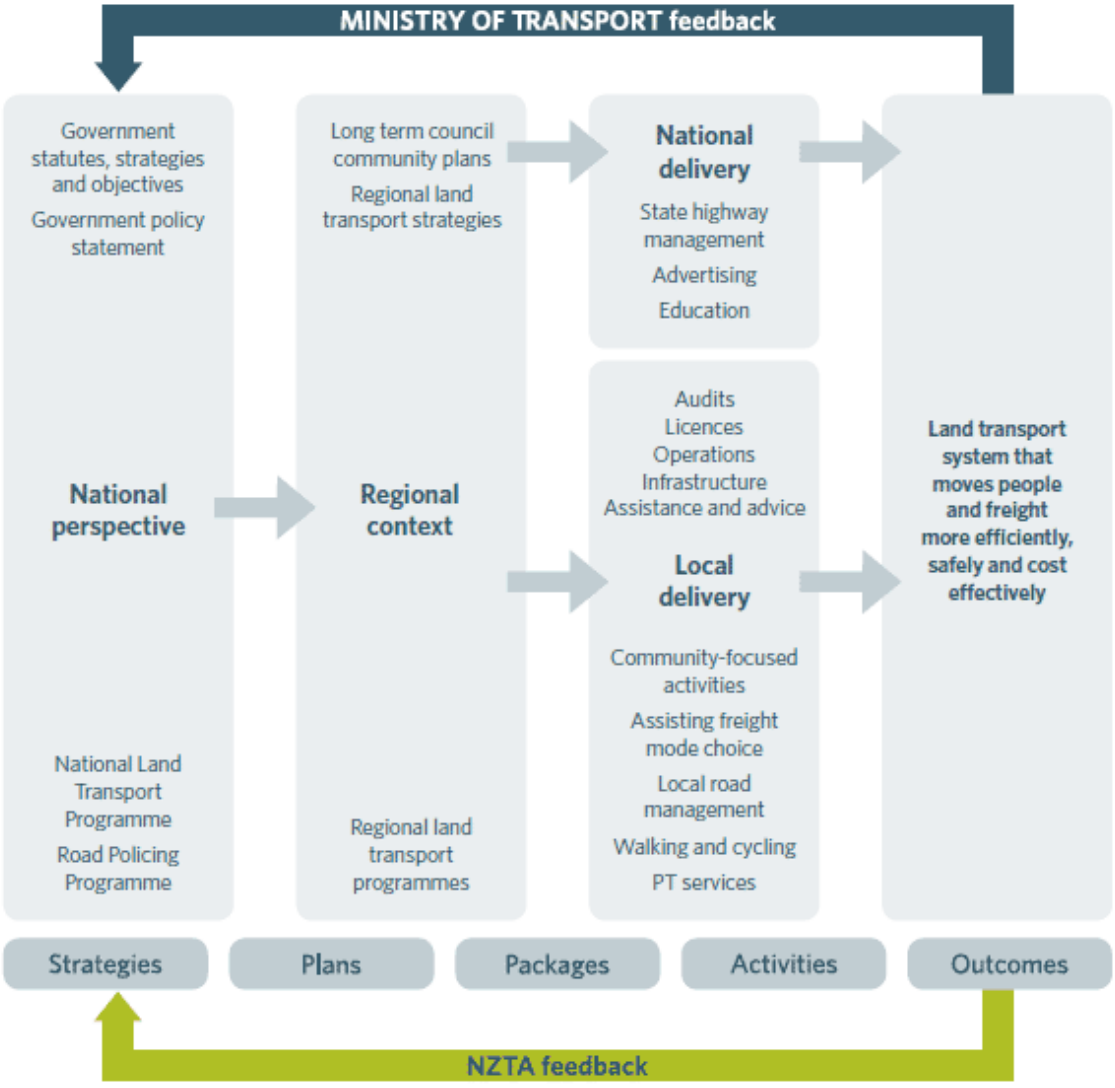
Agency

New Zealand Transport Agency (NZTA) [7] was established on 1 August 2008 and has 12 offices around New Zealand. It is a Crown entity governed by NZ Transport Agency Board and is responsible for contributing to an affordable, integrated, safe, responsive and sustainable land transport system. This includes the planning and funding of land transport which involves the enforcement of laws, regulations and rules, and the collection of revenue. It also involves ensuring that New Zealanders have access to land transport, including through building, operating and maintaining land transport systems. This is the work it undertakes with a number of partners.

As just one part of the transport sector NZTA closely cooperates with sector agencies and others with an interest in land transport. Its role is to provide a vital link between government policy making and the operation of the transport sector.

It maintains close working relationships with:

- transport operators and the general public, who use and interact with transport
- transport committees, regional councils and territorial local authorities, which are responsible for implementing transport projects and other activities funded through the National Land Transport Programme
- suppliers, including contractors and consultants
- the NZ Police, which provides a range of road policing services
- the Ministry of Transport, which is responsible for leading the development of strategic transport policy and monitoring performance of the sector's Crown entities.



The NZTA's interaction with the land transport system

Background

Traffic crash reports (TCRs) are the forms completed by police officers at the scene of all road crashes, including non-injury cases. They record the details of when, how and why the crash happened. They provide essential information to many groups of professionals in the road safety and road construction fields: (a) police officers and police intelligence analysts (b) road controlling authority engineers and safety planners (c) traffic engineers and consultants (d) researchers (e) central government, etc." The reports are used, among other things, to mitigate accidents of similar nature and to build a safer travel environment.

Currently, TCRs are paper forms. Manual entries on paper forms have several drawbacks including entry errors and illegibility. To analyze the data collected through paper forms, the data need to be re-entered into an electronic format. Data consistency checks are not possible in manual data entry: for instance, a single response to a single response question cannot be enforced.

The project turns up to solve these problems with the goal of developing a mobile application as a replacement to the paper form. Such electronic data collection, while reducing some of the issues seen with paper-based data collection, gives many other advantages: there is no need for dual data entry; some of the information can be auto-filled (e.g. time and location); some data can be auto-captured (e.g. driver information from the bar-code of the driving license); some media can be attached (e.g. photos, sketches, and audio). Thus the data collected can be transferred easily to the processing centre. This transfer can be in a manner that lends itself to easy processing (e.g. there may not be a need to re-code the data).

Requirement

Through the meetings with stakeholders, the requirement was explicitly confirmed. Some data fields which reveal to be unnecessary were discarded and additional ones such as airbag deployment were added. The key of the project is to maintain data consistency as well as to enable accurate capture of geo-location and crash diagram. Specifically, the requirement can be divided into four categories and should be fulfilled respectively:

Fast and Accurate Data Capture

As one of the main purposes of this project, the data entry process for police should be well accelerated. This relies on functionalities and user interface of the application. For example, to take advantage of such mobile applications, some additional features such as automatic data filling can be provided. This saves a great amount of time for police, although inconsistent data need to be changed manually by the police sometimes. Moreover, with the Global Positioning System (GPS) receivers on mobile devices, geo-location can be captured automatically. At last, the design of user interface should be carefully considered as different from paper-based TCR, mobile applications normally end up with considerable amount of pop-ups or dialogs which brings difficulties for users to manipulate. Think about a situation when a police has finished filling one section of TCR and wants to move on to next section. What he will do in paper-based TCR is to simply turn the page over. However, in the mobile application, he might found himself annoyed by a dialog asking “the data will be saved, are you sure to move on?”. Such behavior of the application will somehow slow down the process. Therefore, guaranteeing the easiness of using the application for fast data entering is a critical issue that also needs to be taken care of.

Data Validation and Consistency Check

With the current paper-based TCR, there is no way to ensure data validation while police are filling the form. This is embodied in the manual entry errors and the inconsistent information they are provided. It frequently happens in current TCR recording process as there are no indications when the police have made mistakes. Besides, the information that drivers provide can be inconsistent with the one stored in database. In this case, it is obvious that police have no way to check and correct it, which as a result, cause difficulties to identify the crash later on when the data are processed by NZTA. What is worse, it could also leads to the complete discard of a TCR. Hence, in our application, constraints must be put on some data fields to reduce manual errors. For example, use a dropdown box for police to select the options from rather than simply give them a textbox to fill in. A reminder can also be provided so that whenever a police forgets to fill in some parts of the form, he will be informed of it. The most important is to allow automatic data acquiring for some sections such as driver information and vehicle information so that police can compare the data provided by the drivers with the one they get from database. As a result, they can decide which information to respect by discussing with the drivers whenever there is inconsistent information provided.

Multiple Media Support

Despite of the automatic processing capability of mobile application, multiple media support is another key factor that makes mobile devices to a certain extent engage people's everyday life. Clearly, although not the only one, an advantage of such functionality is that it represents events or facts in a more intuitive and interactive way than simple texts with an imaginary description that is more understandable to humans. Since user-friendly and user-centered designs have become primary concerns in most software, it is always important to consider a design from human's perspective at some

points. Thus with the current mobile devices, some conventional text processing procedures can be enhanced or even replaced with media processing. Take driver interview as an example, normally what police do is to write down a whole paragraph whose length can vary from 30 to 150 words. This is definitely a time consuming and sometimes useless procedure as at the end of the day what NZTA needs is only a couple of key words describing the driver's status when the crash happens. With audio capture as a replacement, this problem can be solved because the only thing police need to do is to press the button for recording. Another example is the recording of environmental information such as road, light and vehicle factors. Such factors can be easily captured by taking photos as stronger evidences than words. Nevertheless, it is still the user's preferences that to be respected as the design decision of whether to conform to current text processing.

Data Reform and Transfer

Another key concern is in which format the data should be stored and transmitted. Because the TCRs completed at the scene of crashes are later passed to NZTA for analyzing, a commonly used format for transmission must be used to produce documentation of the reports which is both human readable and machine readable. Extensible Markup Language (XML) is the first format that went in our sight as many application programming interfaces (APIs) have been developed for software developers to use to process XML data, and several schema systems exist to aid in the definition of XML-based languages [8]. Furthermore, it is well supported by Wireless Application Protocol (WAP) and can be used to carry picture and audio file for our purpose. What is most important, a well defined XML schema ensures that the data represented must conform to it so that arbitrary data structure can be formatted to enable better management.

Goal

As mentioned before, the goal of the project is to build a prototype mobile application to explore possibilities of such application to take place of current paper-based TCR. While trying to stress on functionality and data consistency maintenance, the design should also consider likely problems that might be encountered during the implementation. This requires reasonable understanding of the project and research of related topics. Even though at the first glance it seems that the electronic TCRs are totally better than the old TCRs, there are still advantages that electronic ones cannot easily inherit. For example, due to inflexibility of mobile application lifecycle, the user might not feel that the application is completely dominated by him as normally he has to follow the screen navigations, while on the other hand the paper-based TCRs are quite straight forward with everything kept handy. This implies another problem, that is, the inflexible filling order in mobile application doesn't respect users' manners or habits while processing such kind of form. For instance, with current TCR the order that a police fills in the forms may be different depends on various situations. A simple case would be that a police might talk to the driver of vehicle two first and therefore fill in the associated information of it before he actually starts processing information of vehicle one. However, with the Model–View–Controller (MVC) design pattern in mobile application, a user usually cannot perform the recording procedure for next vehicle until he finishes the current one. This limitation can be solved by an innovative design approach as we will discuss later. Clearly there are other issues such as backing up TCRs for reuse and locking screen to reject careless touching events. Hence, the design should try to maximum the scope while following the basic principles of this project.

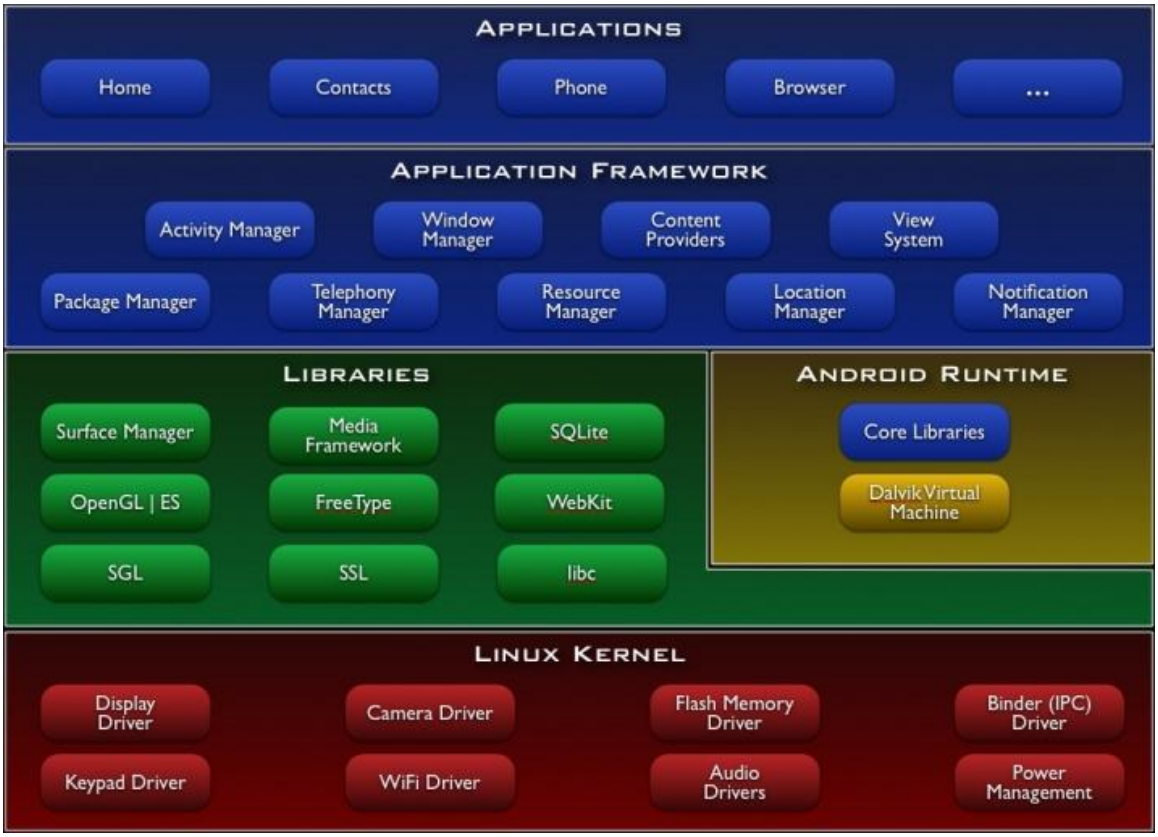
Design

The design is based on Android operating system which is developed by the Open Handset Alliance (OHA), led by Google, and other companies. The platform was created by Android Inc. which was bought by Google and released as the Android Open Source Project (AOSP) in 2007 with a group of 78 different companies forming the OHA that is dedicated to develop and distribute Android. As an open source software system, it is possible for a developer to code in any of the layers [6]. This openness has made Android one of the best platforms over the world.

The Android operating system is a multi-user Linux system in which each application is a different user [2]. Each application is assigned a different Linux user ID by the system which is used to set permissions for ensuring unique access to its related resources. Also, every application runs in its own process with its own virtual machine (VM) isolated from others'. A process is only started when any part of the application's components is needed and shut down when the component is no longer used. These properties have provided a secured environment which prevents resource sharing among different applications without permissions. However, Android does allow applications to share same user ID, to run in same process, to share same VM as well as to access device data with permission. In this case, all permissions must be granted at install time.

Android Architecture

As shown in the figure, android architecture consists 5 different layers among which Linux kernel is never touched by most programmers, although in theory programming can be performed on this layer with the approval from Google.

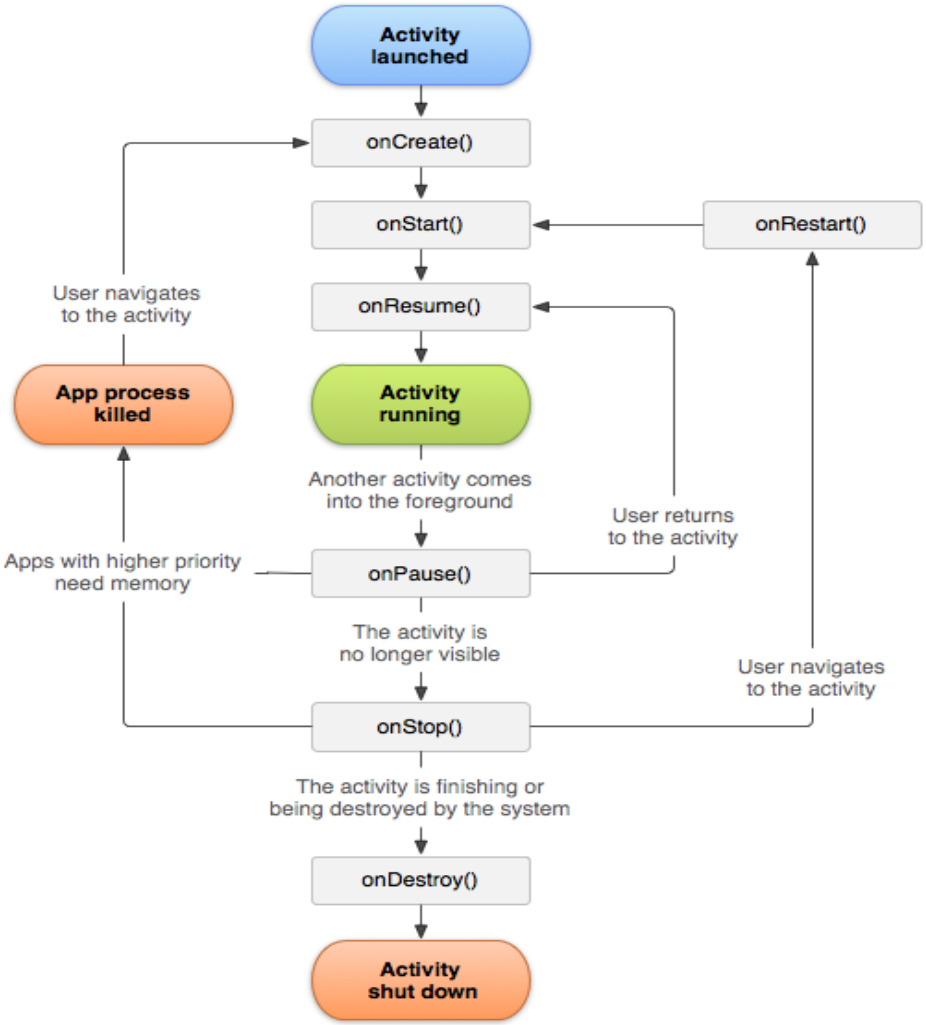


Start at the bottom is the Linux kernel which directly lays on hardware and is used by Android to manage device drivers, memory, process and networking. Above it are core libraries and Dalvik virtual machine that runs Android applications as well as the native libraries which are written in C/C++. Following is the application framework, which provides a framework of services and systems for programmers including Views, Content Providers, Resource Manager, Notification Manager and Activity Manager. At the top are the core applications shipped by Android and other applications written by programmers.

Activity Lifecycle

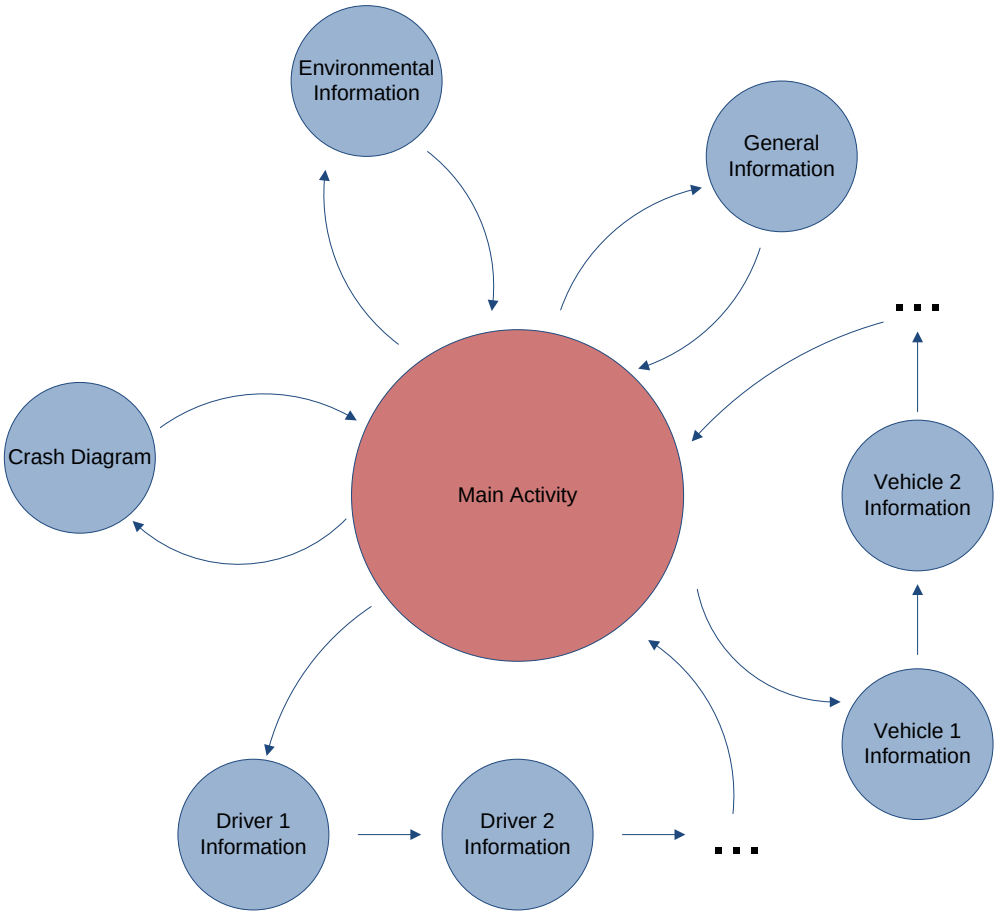
An Activity is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map [1]. Therefore, screen navigations in an android application mainly rely on

management of the activities' lifecycles. The figure below shows the entire lifecycle of an activity.



Typically, an Android application contains several activities and starts with the main activity. Different activities can have different views or actions. In order to switch to content contained in another activity, one must creates an intent specifying either the exact activity it wants to start or the type of action it wants to perform. When a new activity is started, the previous one is stopped and preserved on the "last in, first out" (LIFO) "back stack" then the newly created activity is pushed on top of it. Thus whenever user presses the Back button, the current activity is popped from the stack and destroyed with the previous activity resuming.

Specifically, the screen navigations in our application will be controlled by different activities bound with each other with different views as their contents. Each time the user press a button defined in the view for navigation, the new activity is brought to focus and its view is displayed. This always causes some causal order of the screen. For example, pressing Next button in view one allows user continues to view two and later if he wants to switch to view three, he has to press the Next button in view two. Meanwhile, there is no way that he can go directly from view one to view three, despite of the reason why he would do so. The reason why this happens is because the activities are logically chained or cycled without a centralized activity having access to all of them. Therefore, our first step of designing the activity lifecycles is to split TCR in several sections with a main activity that can navigate to all of them. The following figure shows the relationship among these screens.



Crash Diagram

At the early stages of our design, quite a number of discussions are surrounding the topic of crash diagram. Different from human's habit of using a pen for drawing, free sketching on a tablet doesn't give us a relatively good appearance. Meanwhile, this kind of free sketching doesn't actually reduce the process time and workload for users. Moreover, distinct drawing styles of different users leads to inconsistent appearance even with the same object. The practical results in past TCRs have reflected this fact. Therefore, we are left only two choices, to make use of existing map application for capturing our road picture and draw our own template objects on it or to provide the road template ourselves as well. After a few attempts of the former approach, we conclude that it is difficult to embed a third party map application in our own especially when it's not open source. Meanwhile, although Android supports map view that embeds GoogleMap, the zoom level is not enough for our purpose. This leads to the adoption of second consideration in which we make our own templates for different types of roads as well as objects. This is a significantly challenging task because we have to implement our own methods of handling touch events such as selecting, dragging, rotating and zooming. Also, a lot of geometric computations as well as algorithms need to be implemented to distinguish and to perform these operations.

Data Field Priority

After we have obtained the initial blueprint for our project, the next step is to define priorities of data fields. Clearly, there are a couple of fields which are usually not filled or used in current TCR and a few fields that are extremely important. The statistical results from NZTA give us the clue which fields can be discarded and which must be enforced. Also, during the meetings with stakeholders, the decision of abandoning some fields as well as adding extra fields is made. Enforcements on accuracy of several fields are also

stressed. By then, the amounts of data to be captured as well as their priorities in TCR are well defined and the requirement is mostly confirmed.

Two Design Alternatives

The previous discussions imply that we have not only one design option of how to implement TCR in our mobile application. Here we introduce two design alternatives that we found feasible during research. Although both approaches involve form filling, the way how application interacts with users significantly differs.

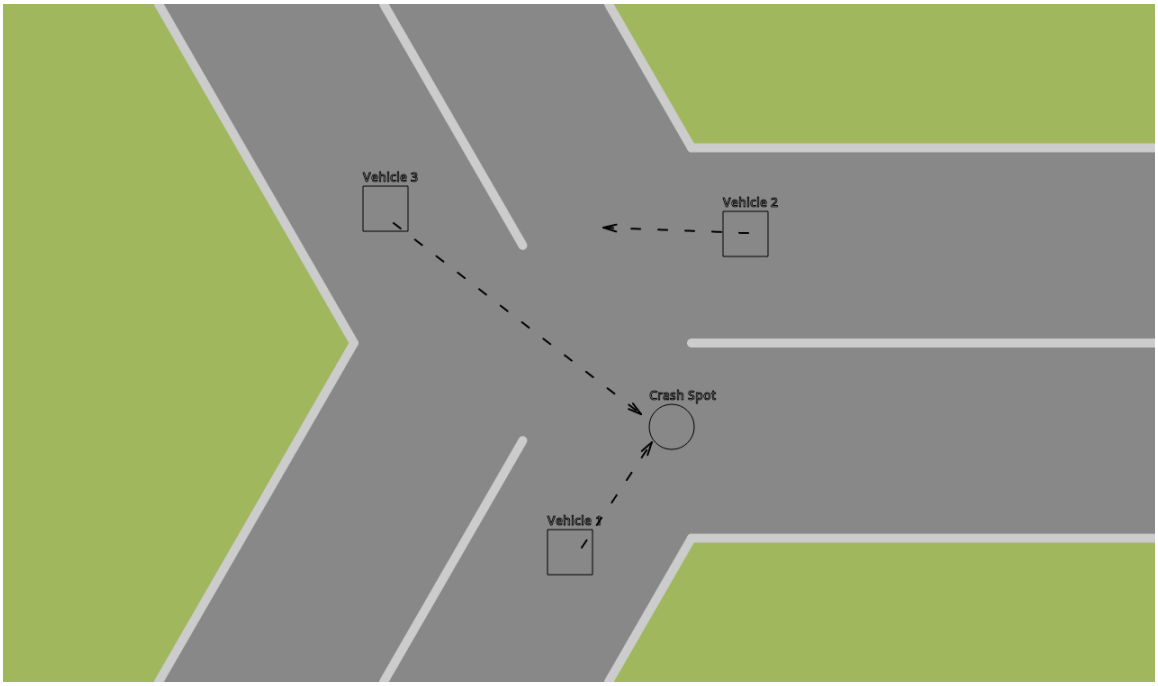
Screen Navigation Based

A direct way of interpreting current TCR in our mobile application is to let the users navigate among screens with each screen containing corresponding view that matches the original form. This is done by using simple logic on buttons in each view to switch among activities. Also, as an optimization mentioned before, a main activity can be provided as the central navigator so that the structure is no longer a ring topology but a star topology. Even better, an Android menu can be implemented together to make it a mesh topology. In this case, the user can have several routines to choose from and can switch to any other routine at any time rather than to strictly follow a single routine. This approach might be preferred by most police as it preserves the basic manner of users while filling such forms, that is, to go from section to section. However, it appears to be unattractive and still has some problems. Firstly, although the logical order of completing different sections can vary, the information of each section must be filled in sequence, that is, the user still has to follow a specific routine within a section containing several screens even if an Android menu is provided, because each section must have a fixed start screen and we cannot use menu for all these sub-screens. For instance, although not exactly the case, while filling information for vehicles and drivers, the user always has to go from first to last. The factor that causes this behavior is the way we implement our model for storing vehicle and driver objects. These objects are stored in collections such as Arraylists so that whenever user wants to add one, a new

instance is initialized and given the last index of collection as its ID as well as its position in the collection. This process predefines the ID for these objects which thus follows a numerical order. Even though this behavior can be solved by letting users enter ID themselves, it doesn't conform to the basic principle of this project, which is, to make the application as much responsible as possible for maintaining data consistency. Also, storing vehicle and driver information in separated collections doesn't result in solid relationship between them as the only way we can associate a driver with a vehicle is to compare their respective index in the collection. This can cause serious problem if the user wants to perform deletion when the operation is not correctly implemented. At last, it is difficult for users to identify the corresponding information of a vehicle later when they start to draw it on crash diagram. It sometimes causes mismatch between vehicle information and crash diagram which brings a lot of trouble for NZTA. With all these disadvantages, we look for a solution that can mostly solve these problems. This leads us to the second approach, which is graphic based.

Graphical User Interface (GUI) Based

Regardless of how we build our application, the users always need to complete forms at some stage of recording process. Yet, we can change the way how users interact with the application by importing a GUI as the basis for entering all information. The concept will be explained step by step. First, the application starts with a GUI which requires the drawing of crash diagram that imaginarily describe the crash. After the user has finished drawing, he can start filling in the according information. The following figure shows a complete diagram at this point.



Now since everything has been lain down, the user can start entering information. The way how we achieve recording of vehicle information is by popping up a dialog when use clicks on a vehicle. The dialog is basically a form that contains all relative data fields. To solve the driver and vehicle mismatch problem, we decided to put vehicle and driver information altogether in a same scrollable dialog. Therefore, the user will know exactly which driver and vehicle he's dealing with and never be able to get it wrong. However changing view without modifying our model cannot solve the problem as vehicle and driver objects are still different objects. Therefore, we decided to merge our driver object into vehicle object so that a vehicle will be a composition of driver object. By compositing the driver object in its corresponding vehicle object, each vehicle will have its own driver which results in indivisible relationship between them. The screenshot of the pop-up dialog is shown below.

With this approach, user still can enter information for other sections using the menu we provided while the order of entering information for vehicles is completely flexible. Each section will also be displayed as a pop-up dialog that located upon the crash diagram. This intuitive design provides user more control over the application which at the same time could result in unawareness of what has been done and what needs to be done. As a result, the freedom of manipulating the application without specified routine can make user forget to enter part of the information. Suppose a crash involving five vehicles, the user normally will click on vehicles in a random order to process. Since we combine vehicle and driver information altogether, there will be quite an amount of information for each vehicle. Therefore, after filling information for several vehicles, the user could end up with forgetting to record information for the remaining ones. To avoid this, we need to provide a reminder telling the user whenever there is information that has not been filled. User preference is another issue that we currently cannot locate. As we didn't take surveys of users' preferences for process TCR, we cannot conclude that this new approach will be well adopted.

XML Schema

Since our application design phase has almost come to an end, it is about time for us to think about how to define a schema for our documents. The primary advantage of schema languages is that descriptions in schema languages are more precise than those in prose and that we can rely on validators rather than carrying out human inspections [5]. There are several schema languages that have been proposed in the past, for instance, DTD and W3C XML Schema. The reason why we prefer XML Schema than DTD is because it aims to be more expressive than DTD and more usable by a wider variety of applications [4]. An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic syntactical constraints imposed by XML itself [9]. Therefore, a given document can be checked against the constraints to determine whether it is valid or not. In our project, the use of XML language is essential as it not only consolidates data consistency but also provides a human readable description of the captured data.

Progress

Through this semester, two meetings with stakeholders as well as several regular project meetings are held. The requirement is confirmed at early of the semester and right after than we start our design of the application. Junction and roundabout road templates have been developed. The user can select the according road type with the number of roads specified and the road is automatically drawn on the screen. Currently, only dragging has been implemented on the roads so that user can drag a road to any angle they want. In the future some additional feature such as rotating and zooming might be implemented. Also, a couple of object templates are created. They include vehicles, arrows and crash spots. Note that they are currently represented with simple shapes which will be changed to icons in the future. As mentioned by Wilding [3], Icons

can make an interface visually more interesting and are appropriate when they communicate better than text. Besides, we have started defining the XML Schema for our data structure.

Conclusion

Although current crash diagram looks pretty simple and the templates are still not completed. A series of improvements on both functionality and appearance will be achieved in the future. The problems encountered during implementation enhance my experience on user interface design and Android programming.

Future Plan

The first delivery of the prototype will take place in one mouth's time. By then, a workable application will be built and it will be capable of captured all required information. The information will be parsed into XML files which satisfy the XML Schema we define. This XML file is then sent out by email.

Road Templates

All types of road templates will be finished up with the functions that allows user to move around or rotate the whole template within screen, drag separated roads and possibly zoom it.

Other Templates

All objects will be split into four categories which are roads, vehicle, lines and others. In this way, the whole interface functions as a toolkit which allows user to select associated objects to draw from pop-up dialogs in these categories. Also icons will be

used as much as possible as a replacement of current simple shapes in order to provide a better appearance. Dragging and rotating on these objects will also be available by then.

Forms

All views of forms will be completed in a clean manner so that it is easy for user to fill in. Also data will be captured and stored so that they can be represented later for revision.

Geo-location

Although geo-location can be easily obtained, special algorithm needs to be implemented to ensure a more accurate result. Also we need to find a method to let use manually correct the position of the OverlayItem on GoogleMap.

Reference

[1]Android Developers. Activities. Retrieved June 2, 2012, from

http://developer.android.com/guide/topics/fundamentals/activities.html

[2]Android Developers. Application Fundamentals. Retrieved June 2, 2012, from

http://developer.android.com/guide/topics/fundamentals.html

[3]Cliff Wilding. 1998. Practical GUI screen design: making it usable. In *CHI 98 conference summary on Human factors in computing systems* (CHI '98). ACM, New York, NY, USA, 125-126.

[4]Dongwon Lee and Wesley W. Chu. 2000. Comparative analysis of six XML schema languages. *SIGMOD Rec.* 29, 3 (September 2000), 76-87.

[5]Makoto Murata, Dongwon Lee, Murali Mani, and Kohsuke Kawaguchi. 2005. Taxonomy of XML schema languages using formal language theory. *ACM Trans. Internet Technol.* 5, 4 (November 2005), 660-704.

[6]Mohsen Anvaari and Slinger Jansen. 2010. Evaluating architectural openness in mobile software platforms. In *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume* (ECSA '10), Carlos E. Cuesta (Ed.). ACM, New York, NY, USA, 85-92

[7]NZTA. 2009. NZ Transport Agency. Retrieved June 2, 2012, from

http://www.nzta.govt.nz/

[8]Wikimedia Foundation, Inc. 2001. XML. Retrieved June 2, 2012, from

http://en.wikipedia.org/wiki/XML

[9]Wikimedia Foundation, Inc. 2004. XML schema. Retrieved June 2, 2012, from

http://en.wikipedia.org/wiki/XML_schema