COMPSYS

Engineering Student Centre

Practical Work Report 'A' Grade Exemplar

By consulting this exemplar you agree:

- That you are consulting this report for informational purposes only
- That you will not remove this Practical Work Report Exemplar from the Engineering Student Centre
- That you will not take notes from, photograph, photocopy, or otherwise duplicate any material contained in this report

2015

Work Report: TDG

LUKE HARRIES - 1827106

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

30 July 2015

Traffic Design Group Ltd.
103 Carlton Gore Rd,
Newmarket, Auckland

Dates Worked:
24/11/14 to 27/02/15

Summary

My experience at TDG in Auckland exposed me to both general and sub-professional work experience. The project at the core of my sub-professional experience involved building an app that brought an analogue road surveying technique into the 21st century and increased the efficiency of the surveying process significantly. The experience of working on that app, and working as a technical assistant helped me to gain new skills and perspectives. Working at TDG showed me how social dynamics and office culture were critical to an efficient and happy workforce, particularly when projects involved many staff with different skills.

Acknowledgements excluded for privacy reasons.

Table of Contents

Table of Figures

Introduction

Traffic Design Group Limited, which trades as TDG, is the largest specialist transportation engineering consultancy in New Zealand. TDG has offices around New Zealand, in Auckland, Hamilton, Tauranga, Napier, Wellington, Nelson and Christchurch. TDG offers a wide range of consulting services, including parking and access, traffic management, traffic modelling and analysis, and many more. TDG has been involved in numerous high profile infrastructure projects in New Zealand, like the Wellington Inner City Bypass and the Auckland International Airport master plan (TDG Ltd., 2015).

Figure 1: TDG Ltd. logo.

About the TDG Auckland Office

TDG Auckland is located on Carlton Gore Rd in Newmarket. TDG is based on level 1 in SMC House, a late-90s office building which is one of several of the same style on Carlton Gore Road. The building includes a large two-level basement secured car park, as well as an unsecured car park at the rear for visitors.

TDG occupies the western side of the first floor. The entrance to the office opens into the reception area on the street-facing end of the office, where the conference room, admin and executive offices are located. Managerial and senior staff offices are located down the hallway, and an open-plan office area is located at the back end of the office. This end of the office overlooks the western train line, and the new University of Auckland Newmarket campus. With the exception of the admin, management and senior staff, all other staff are based in the open-plan area of the office. This layout enables efficient collaboration between engineering staff, and facilitates a social work environment. Pictured in Figure 2 is the floor plan of the office. My work station is indicated by the red arrow.

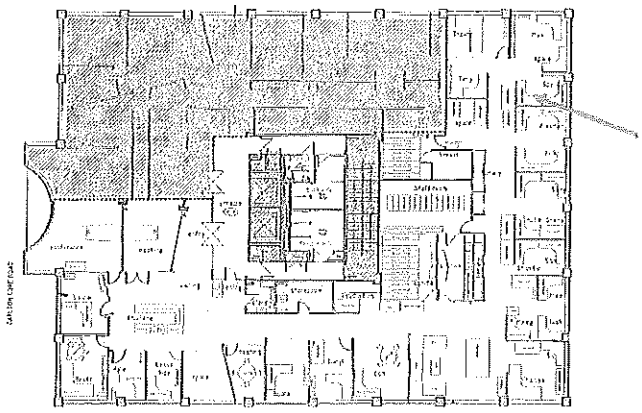

Figure 2: Floor plan of TDG office.

A kitchen and large break room are located near the open-plan end of the office. The kitchen includes an oven, stovetop, sink, refrigerator, dishwasher, espresso machine and boiling and filtered water taps. The full range of kitchen amenities means that many staff opt to arrive in the office early to beat the traffic, and are able to still cook or prepare a full breakfast. Inside the break room is a large table that occupies most of the room, with enough seats for all office staff to be in the room at once.

At TDG, it is normal practice for staff to take their morning break at the same time, with a few packets of biscuits and a fruit bowl provided each day – a long running company tradition that helps to foster strong team relationships. Another long running tradition at TDG is the morning-tea shout; all staff must provide a morning tea for the other staff on their birthday, and on their anniversary of joining the company. The communal morning teas were a great chance to socialise with other staff, or catch up on the morning's news.

The office also included some recreational facilities for staff, such as a TV in the break room with Sky TV – which over summer was often showing the cricket – as well as a couch area and a ping-pong table, which was often used during the day for short 'first to five' style matches. The overall effect of these amenities was a noticeably happy staff, who were focused and hard working throughout the day.

Each week at TDG begins with a 'Job meeting' during Monday lunch time. In order to reduce the amount of work time the meeting cuts into, it was held during the lunch break, and lunch was provided to the staff; the food was ordered from somewhere different each week, and staff placed their orders the week before. During the meeting, each staff member reads through their progress and the status of each of the jobs or projects they are working on. This was also a chance to communicate with other staff whose input was required for a project.

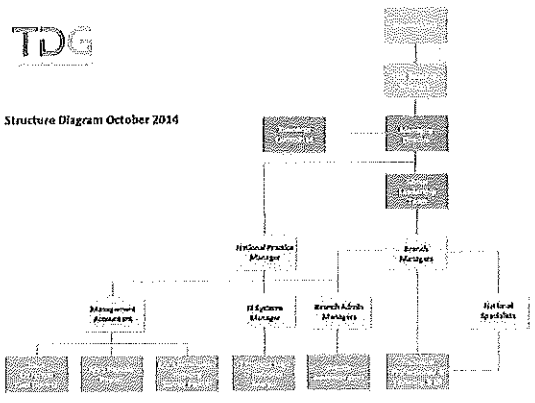

Figure 3: TDG staff organisation structure diagram.

Figure 4 outlines the staff structure at TDG. Day to day operations in the Auckland office are overseen by branch manager Don McKenzie, who was also the engineer overseeing my sub-professional work.

Administration in the Auckland office was managed by Kyle Martin. There are just over 20 staff working in the Auckland office, and approximately 80 staff employed by TDG nation wide.

Description of Work

My professional relationship with TDG began several years ago, when I worked as an office assistant, helping with admin work, printing and preparing documents, filing, and digitising survey data. Earlier in 2014, I conceptualised an iPhone app idea as part of an ENGGEN 403 business plan assignment, which solved an issue I had noticed at TDG in the past – effectively updating and simplifying an analogue road surveying technique. I approached TDG with this idea, and TDG agreed to take me on to develop the application for them, as well as assisting other technical staff in the office.

My time at TDG was partly sub-professional work, and partly general work. The first few weeks of work was all general, as some of the equipment I needed to start work on the software project was not yet available. Once the project kicked off, I spent one or two days each week working on the general tasks. Figure 5 is a photo of my workspace in the office, showing my PC workstation, as well as Macbook used for iOS development.

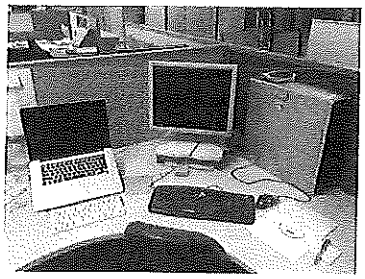

Figure 4: My office work space.

General Work

For the first 3 weeks at TDG I engaged exclusively in general work, due to the unavailability of some equipment needed for the sub-professional project, and because some of the relevant staff required to start working on the requirements development were not available to do so. As such, I worked with TDG's IT Systems Manager (as a technical assistant, working on IT and network related tasks as required. Beyond the initial few weeks of working on general tasks, I spent one to two days each week on additional general work over the course of my time at TDG, also working with senior Global Information Systems developer

Preparing new workstation PCs

The first and most significant general task was to oversee and execute the setup and preparation of around 20 new workstation PCs, which were to replace older hardware in the office. The main aspects of this task were installing and setting up the Windows operating system, setting up the correct network and server access permissions on each machine, and installing software packages. The installation of software packages on each machine ranged from typical graphical installers, to more low-level command line installations, particularly for the custom TDG-built software packages and some of the more technical engineering software packages.

Although not particularly taxing, the task required patience, as the larger software packages such as the computer aided design (CAD) tools and transportation modelling tools tended to take a long time to install and required more complex than normal set up. Part of the installation involved setting the program default settings, so to minimize the disruption to work flows when the old computers were swapped out. This included reinstating macros in Microsoft Office. Depending on the program, I was able to partially automate this process by exporting the user defaults and importing them onto each machine, but for many applications this process remained a manual task.

Some engineers, like those doing complex modelling and simulations, required additional computing power. For these engineers, I was required to upgrade the memory in the machine so that it could handle the large data sets that these simulations generate. As the PCs were fairly standard desktop towers, this was a straightforward process. Still, precautions were taken to ensure that the internal components were not damaged by Electrostatic Damage (ESD). An anti-static wristband was worn at all times while dealing with internal components.

Cataloguing and testing computer equipment

The next general task involved cataloguing and testing all of the computer and networking equipment in the office. This included networking cables and switches, keyboards, mouses, displays, and video cameras among other things.

Each item had to be labelled and catalogued. Items were then tested to confirm they they were still functional, and visually inspected for any damage. Details of the testing was recorded in a spreadsheet so that items with defects could be easily identified by their label, and to enable a programme of priorities for upgrade or replacement.

The testing process was different for each device. For testing ethernet networking cables, a Network Cable tester like the one pictured in Figure 5 was used. This device was able to indicate the correct operation of each twisted pair in the cables, and this was noted down. Some cables had only one defective twisted pair, and were still kept depending on which pairs were damaged, as not all networking protocols require all four twisted pairs. Other cables had more defective pairs, and were discarded. A basic knowledge of common twisted pair protocols like *100Base-TX* and *1000Base-T* was required, so to make a judgement on whether the cables would still be usable.

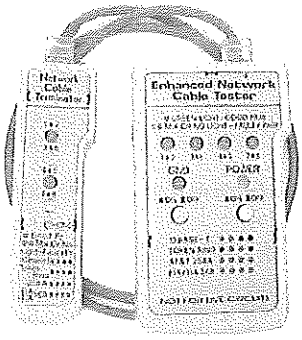

Figure 5: Example of an Ethernet cable tester.

Testing keyboards and mouses was straightforward. The devices were plugged into a working computer, and each key, button and other controls (like scroll wheels) were tested to verify that they worked fully. Devices with one or two faulty buttons were discarded, as these were not appropriate for office use. To test the LCD displays, I built a simple tool using HTML and JavaScript which displayed, in full screen, each of the primary pixel colours (red, blue and green) as well as white. This allowed the easy detection of dead pixels (where the pixel would not light up) and stuck pixels (where the pixels were stuck on one colour and did not turn off).

For the other equipment, such as camcorders, it was important to test every function. Camcorders were often used for long survey recordings, so these devices were left to record for several hours at a time to ensure they were still capable of this – a small number of cameras appeared to function correctly at first, but ran into errors after an hour or more of recording.

Software Quality Analysis of TDG's new GIS application

The final general task I assisted with was the Software Quality Analysis and user testing of TDG's new GIS application, which was being developed by one of TDG's GIS specialists Spencer Han. My previous experience with the SQA team at Navico was put to the test here, as I was required to thoroughly test the web-based application for bugs and errors, as well as to assess usability and performance.

The application, which was a visual map-based database of TDG's thousands of jobs and projects, required thorough testing to ensure that engineers would not run in to trouble when

using the service out in the field. Every function was tested multiple times to ensure correctness. To investigate whether error-handling was up to scratch, I tested every function that required user input with as many types of erroneous inputs as possible, from numbers and symbols to non-Latin scripts. This type of error-handling testing is important, as user input is generally the greatest source of uncertainty when developing software.

As the application was web-based (i.e. the software is run on a server and the user accesses it through a web browser), I was tasked to ensure that the application could be run across all types of browsers, on all devices. Through this testing, it was revealed that on smaller devices such as mobile phones, the user interface was very difficult to use, and test was often too small to read easily. This led to the implementation of a *responsive* design, where the user interface changes automatically depending on the size of the screen, or *view port*.

Sub-Professional Work

The sub-professional part of my work at TDG was a software project that required designing and developing a mobile application that could survey road curves and determine the 'safe curve speed', also known as the curve advisory speed. Curve advisory speeds are common on New Zealand's winding roads, and are used to indicate to motorists the maximum 'safe' speed at which they should travel through the curve or corner. These advisory speeds are sign posted before the curve when the maximum safe speed is lower than the legal speed *limit* of the road.

TDG is often commissioned to assess a stretch of road, or *road corridor*, and determine which curves require sign posted advisory speeds, and what those speeds should be. Currently, these speeds are calculated using a dated analogue method. A *ball bank meter*, sometimes known as a *side thrust gauge*, pictured in Figure 6, is mounted to the car's windscreen. A small ball bearing rolls along a curved track, and measures the amount of lateral (side to side) force exerted on the vehicle. The driver of the vehicle must maintain a constant speed while driving through the curve. Because the driver cannot safely drive and read measurements at the same time, this process requires either two people in the car, or the survey must be filmed with a video camera that is set up

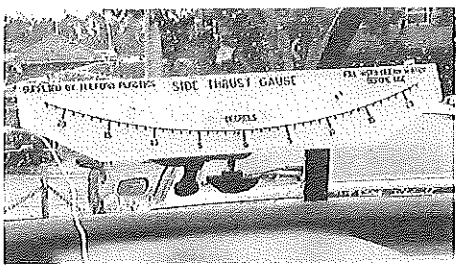

Figure 6: A side thrust gauge.

behind the driver, while the driver reads the speed out loud from the speedometer. Once back in the office, the survey data must be manually entered into the computer, and the speed is manually calculated as a function of the ball bank reading and the speed. This process must be repeated several times at different travel speeds, and for each direction through the curve.

It is quite clear that the analogue surveying method is resource intensive, time consuming and inefficient. To add further complication, TDG only has one side thrust gauge, which was part of a one-off manufacturing run many years ago, meaning TDG has no way to buy another. The alternatives were not attractive either; replacement gauges, analogue and digital, are expensive, and do not solve many of the inefficiencies of the process. A fully automated solution, called *CARS*, costs thousands of dollars a year on a subscription basis (Reiker Inc., 2014) – not a viable or economic long term solution.

Instead, the mobile application concept uses advanced sensors already included in many mobile devices to streamline and automate the process, significantly reducing the time and resources required to conduct this type of survey. Company-provided phones at TDG are predominantly iPhones, so it was decided that Apple's iOS would be the target platform. Apple iPhones include all of the sensors required for this application; an accelerometer chip digitally measures force exerted on the device in all directions, and a GPS module for geo-location can be used to calculate the speed at which the device is travelling.

Initial research and testing

The first step in designing the application was to thoroughly research the survey process, and determine if the target hardware platform would be capable and accurate enough for this type of surveying. Researching such a niche civil engineering topic proved to be difficult, but achievable with the help of TDG's Wellington based librarian Merryn Hedley.

In New Zealand, the key source of information about curve advisory speeds is the Manual of Traffic Signals and Markings, or MOTSAM. Appendix A3 of MOTSAM contains detailed information about curve advisory speeds in New Zealand, including the criteria for erecting advisory speed signs, where they should be placed, and how the advisory speed should be

determined (New Zealand Transport Agency, 2001). Figure 7, taken from Figure A3.1 of MOTSAM shows how the advisory speed should be calculated from the observed speed V_o and the ball bank reading B, and how the result should be rounded for signage.

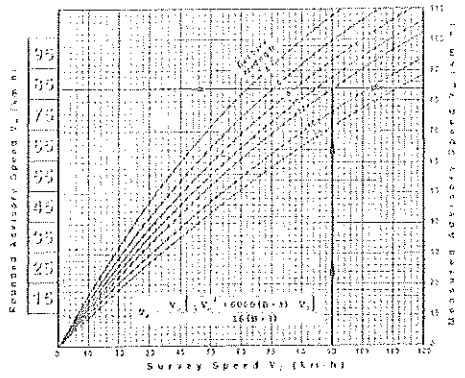

Figure 7: Graph and equation for determining curve advisory speeds. From MOTSAM A3.

One potential hurdle was in the different units of measurement for side thrust. The formula given by MOTSAM deals in ball bank meter degrees (°), but data from the iPhone's accelerometer is in g-force units. A paper from the Transportation Research Record journal suggested that the relationship between Ball Bank indicator degrees, or BBI° and g-force units was a linear relationship, meaning that the BBI° value could be derived from the g-force values with a simple scale factor and offset (Carlson & Mason, 1999). This relationship is shown in Figure 8. This relationship was confirmed by an experiment conducted where a simple iPhone app displaying accelerometer data in g-force units was filmed alongside the side thrust gauge.

$$\mathrm{BBI} = 1.115 + 52.627(\mathrm{ULA})$$

Figure 8: Equation relating Ball Bank Indicator degrees to Unbalanced Lateral Acceleration (ULA) in g-force units.

The speed values derived from the iPhone's GPS needed to be verified as well. This was tested in a similar way to the accelerometer, by conducting an on-road test and comparing the value displayed by the iPhone to the value read off the car's speedometer. The results of this testing showed that the readings from the car's speedometer were consistently about 4km/h higher than what was measured by the iPhone. It was explained by the engineers in the office that vehicle manufacturers often deliberately calibrated car speedometers to read a few km/h higher than the actual speed, to compensate for possible error in the measurement. The fact that the difference in readings was a fairly constant value suggests that the GPS speed was displaying an accurate measurement, simply without an offset. An article from the Journal of Biomechanics also supported the conclusion that the GPS sensor would be adequate for measuring speed, showing that GPS generally determines velocity within 1.5 km/h of the actual velocity (Witte & Wilson, 2004).

Requirements gathering and application design

With initial testing and research completed, the design phase of the app could begin. The first step in designing the application was to determine what functional and non-functional requirements the application needed to satisfy. To do this, I collaborated with senior transportation engineer George Eivers, based in TDG's Napier office. George and his team in Napier would be the main users of the app, as the Napier office was often commissioned to do corridor analyses.

Functionally, the application needed to correctly measure and calculate advisory speeds of curves. George explained that video recording was also an important requirement, as recordings of surveys were often used by engineers to review where existing signage was, and where new signage should be placed. Another important functional requirement was the ability to export survey data to a spreadsheet, as the raw data sometimes needed to be reviewed, or included in documents and reports. Ideally, the application's survey function should be able to do two types of tasks. The first type involves surveying specified curves that are already known, and calculating their advisory speeds. The second involves surveying much longer stretches of road in order to detect which curves, if any, need to be surveyed again to calculate their advisory speed. Finally, the application needs to be able to operate without a data connection, as the locations in which surveys are undertaken are often remote, and lacking cellular coverage.

Based on the guidelines given to me by George, and how the application was intended to be used, I was able to extrapolate additional non-functional requirements for the application to help guide its design. The app was being designed for use in a vehicle, and this necessitated the requirement that the app should not need any user input during operation. If input was required, the user should be able to provide that input without taking their eyes and focus off the road. To achieve this, the design would allow users to pre-plan surveys on a map, so that the app could automatically start and stop surveying based on the GPS location – a technique known as *geo-fencing* – eliminating the need for the user to interact with the device while driving. Where user input was required, for example, to cancel a survey, the user interface (UI) consisted of large buttons that could be tapped easily without the user needing to focus on the device screen.

The app also needed to be very stable, as corridor assessments were often along continuous stretches of road up to 50-60km long. This means that the app needs to be capable of running continuously for up to 45 minutes at a time without crashing or running into other problems.

Some non-functional requirements that usually constrain mobile applications were not applicable for this project. Mobile applications usually strive to minimize impact on the device's battery, but as this application is designed to run in a car where users can keep the device plugged into the 12V power socket, it is not necessary to be bound by that constraint.

With the requirements in mind, the next step was to develop design concepts for the app. This involved sketching on paper what the app could look like, and how different 'views' in the app would be linked. Sketching designs on paper allows many alternative designs to be conceptualised much quicker than using a graphics program. Figure 9 shows how designs are refined on paper before a single line of code is even written. Deciding how the views will link to one another is an important part of the design process, as it determines the 'flow' of the application. Keeping the end user in mind throughout this design phase is critical; the design should explain itself, and users should be able to use and navigate the application without being told or instructed on how to use it.

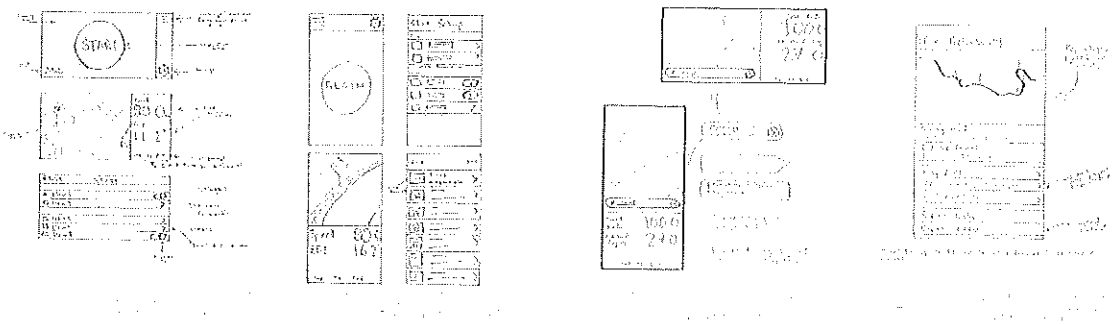

Figure 9: Preliminary design sketches of the user interface.

Development

The design phase creates a base from which the application can be built and developed. The application is built using Xcode, the Integrated Development Environment (IDE) for Apple platforms. Xcode includes tools for developing code as well as building the user interface. My previous position as a software development intern at Navico Auckland provided a good knowledge base for this project, particularly as that project also involved developing applications for iOS devices.

The initial development of the app involved creating the basic 'outline' of the app's navigational structure. Starting with the main menu, each view was built to the point where it could interact with and progress to another view. The result of this first iteration of development was effectively an app 'skeleton' that included all of the core views within the app. The overall structure of the app can be seen in the app "storyboard", which shows the interface of each view, and the connections between views. From there, each view could be fleshed out and refined until all the functionality had been implemented. The aim of this process was to focus on *working* software – that is to say that progress was generally measured by completed and working functionality, as is the case with Agile software development methodologies.

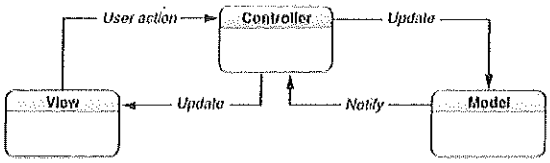

Figure 10: Diagram showing information flow in the MVC pattern.

The development of the app followed the Model-View-Controller or MVC design pattern, which is one of the core principles of iOS app design. The MVC pattern stipulates that the View (i.e. the user interface), Controller (the back-end code) and the Model (the database/data storage) are all separate entities that interact with one another, forming the full app. Figure 10 is a diagram that shows the overall flow of information and control in the MVC pattern. Using the MVC pattern means that the user interface can be designed separately to the back-end code, and elements of the UI can then be linked up to variables or functions in the code.

The basic data model was also developed at this time. The data model consists of a simple *graph* that shows how entities in the database are related, and what properties each entity should have. Figure 11 shows the data model for this application.

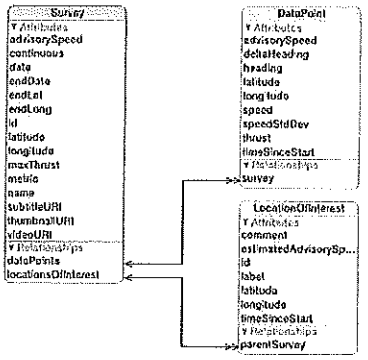

Figure 11: Graph of the data model, showing how data is structured and related in the app's database.

Once the outline of the app was built, functionality was developed and refined in a fairly logical order, starting with the core functions and then moving on to the auxiliary functionalities.

The first functionality to be implemented properly was the surveying function itself. As is often the case in software development, the implementation of this function first required some supporting functions, in this case functions were required to handle the incoming stream of sensor data. Classes were developed for the GPS and accelerometer data types. These classes worked by 'subscribing' to data from the operating system. Each time new data was available from the hardware, the operating system broadcasts a notification, which triggers a method within the class that had subscribed to those notifications. The class method then reads the data, and processes it as necessary – for example the accelerometer handler converts the incoming data from g-force units to BBI° units. The data is then stored in a variable that can be read by all functions in the application. It is important that only these classes can write to those variables, to avoid read/write conflicts. Data from the sensors arrives at irregular intervals, so this method of reading and storing as it arrives means that other functions can read the value at regular time intervals, and always get the latest value.

Once these data handler classes were implemented, the survey function could be implemented. This function used a continuous timer to poll the accelerometer and GPS data at regular intervals. The repeating timer is set to 'time out' after a set time period. For each time out, a specified function is run and the timer is restarted. The time out function reads the GPS and accelerometer data, and stores the BBI value, the speed, the GPS location co-ordinates, and the exact time. At the end of the survey, all of the data points are analysed, and the advisory speed is calculated based on the data point with the maximum lateral force.

Mid-way through the development process, it was decided to transition the app's code from Objective-C – the previous *de facto* language for iOS development – to a new language called Swift. Swift is a new language developed by Apple specifically for its iOS and OSX platforms, and has the benefit of being compatible and interoperable with Objective-C, meaning that an app could have code in both languages and still compile and build into a running app. As such, new classes were written in Swift, but completed classes mostly remained as Objective-C. The motivation to migrate to swift was based on future-proofing; it is expected that Swift will quickly replace Objective-C as the primary language for iOS development. Also, many new

functions and Application Programming Interfaces (APIs) are more straightforward to implement in Swift. Learning swift was a fast process, as the language is very thoroughly documented by Apple and has been widely adopted by other developers and tutorial publishers.

One challenge in the development of the survey functionality was that the accelerometer data was updated so rapidly that the values tended to fluctuate with the slightest movements, like on a straight stretch of road with an uneven surface. The result was an incoming signal with a lot of noise, which caused data analysis to be inaccurate. In order to clean up the signal, a low-pass filter was implemented. This filter worked by adding a proportion of the new value to the inverse proportion of the previous value. This meant that small and momentary changes had less effect, but large and persisting changes would have a greater effect on the new value. Figure 12 shows the equation for this filter, and the corresponding Objective-C code.

$$a_{n+1} = (a_n \times k) + (a_{n+1} \times (1 - k))$$

```
double newAccelX = (accelerometerData.acceleration.x * kAccelFilterFactor.doubleValue) + (accelX.doubleValue * (1.0 - kAccelFilterFactor.doubleValue));
double newAccelY = (accelerometerData.acceleration.y * kAccelFilterFactor.doubleValue) + (accelY.doubleValue * (1.0 - kAccelFilterFactor.doubleValue));
double newAccelZ = (accelerometerData.acceleration.z * kAccelFilterFactor.doubleValue) + (accelZ.doubleValue * (1.0 - kAccelFilterFactor.doubleValue));
```

Figure 12: Low pass filter equation and code, where a is acceleration and k is the filter coefficient.

Other challenging features to implement were video recording, and the ability to superimpose recorded sensor data over video playback. To master audio/video technologies on the iOS platform, I turned to Apple's World Wide Developer Conference sessions, which are recorded seminars presented by Apple engineers explaining their new technologies. By watching a number of these sessions I was able to understand how media like audio and video are structured in software, and was able to synthesize and add a subtitle track to the video container.

Also challenging was implementing the geo-fencing system that would automatically start and stop surveying based on map locations pre-set by the user. Although the iOS platform includes built-in functions for geo-fencing, the testing found that these APIs were unpredictable and unreliable, and so wouldn't be suitable for this application. Instead, the function was implemented from scratch, by polling the user's location and calculating the distance from the survey area. To reduce resource consumption, the location was polled initially, and then the time between subsequent polling was determined by the distance: if

the user was tens of kilometres away, the app could wait longer, but if they were only a few hundred meters away, the app would poll frequently.

Screenshots of the final product are shown in Figure 13. The layout is designed specifically to adapt to different screen sizes and orientations. Note the large button and tappable areas, designed to be fast to find, reducing driver distraction.

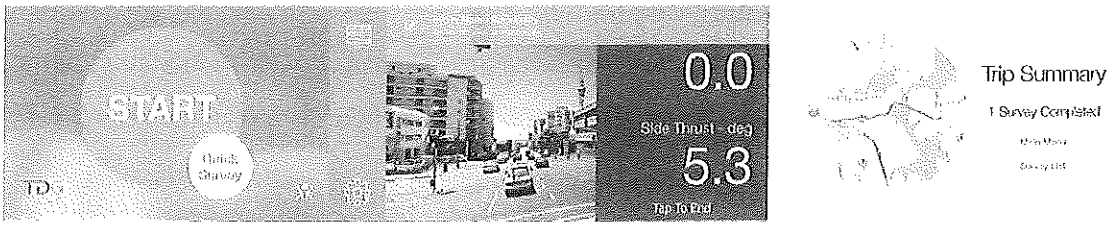

Figure 13: Screenshots of the final app. From Left to Right: Main Menu, Survey Mode (simulated), Post Survey Summary.

Testing

Throughout the development process, the app needed to be tested in a number of ways. The application was frequently tested at a code level, using a breakpoint technique where the program is 'paused' at certain locations in the code to ensure that variables contain expected values. The app was also tested using the Xcode "Instruments", which are a suite of test bench utilities for assessing performance, memory leaks and other metrics.

The app functionality itself was tested after the implementation of each major feature. This testing could sometimes be done in the office using simulations, but was mostly done by testing the app on real roads. I was able to use cars in the TDG fleet to conduct this testing during the day. As the app is mostly intended for open-road use, I had to venture out of the city in order to conduct testing: Whitford in East Auckland, the Waitakere ranges, and North Auckland were all ideal areas for testing. The set-up for in car testing was very simple, as shown in Figure 14. It involved a Logitech windscreen mount, which had the advantage of being magnetic for fast and easy mounting and dismounting, and it left the camera exposed for clear video recording. On road testing revealed many issues that might not have been detected in the office, such as the tendency for the device to sometimes overheat on

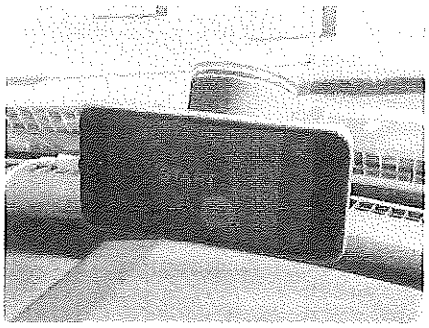

Figure 14: Survey and testing set up, showing phone on magnetic windscreen mount.

sunny days – requiring a reminder in the app for the user to consider using windscreen air-conditioning while surveying.

Another important aspect of 'real-world' testing was to ensure that the app would run correctly on all of the target hardware it supported. The app was being developed for the then latest version of iOS (iOS 8), which meant that every iPhone since the iPhone 4S, released in 2011, could potentially run the app. Differences in hardware meant that compatibility was not guaranteed though, so in order to verify this, the app needed to be tested on each device. It was decided that support would be dropped for the iPhone 4S, but kept for the iPhone 5, 5S, 6 and 6 plus. This was due to the lower quality of the iPhone 4S's accelerometer hardware, which reduced the integrity of the data it gathered. This was not a major issue for TDG, as almost all company issued phones were iPhone 5 or later.

When the main phase of development was completed, the app was made available to TDG staff for beta testing. This was done using the TestFlight service offered by Apple. Beta testing required that the app was fully compliant with Apple's app store rules and regulations. TDG staff were then invited via e-mail to download Apple's TestFlight app onto their devices, where they could install beta versions of the app. Before making the app available for beta testing, two additions were made. The first was the addition of a service called InstaBug. InstaBug enabled testers to send feedback from within the app using a two finger gesture, and could include screenshots and a message. The other was a service called Fabric.io, which automatically logged over the internet any situations where the app crashed. The advantage of Fabric.io was that it included information about what instructions were running, so that bugs causing crashes could be fixed much faster. At the end of my time at TDG, the beta phase had not yet concluded, as the busy Napier team was yet to thoroughly test the app. In order to keep development on track after I had left, a handover video conference was held with TDG's new developer in Wellington, who would be responsible for the App's eventual distribution on the App Store.

Documentation

An important step in the development of new software is to adequately document it. This is for a number of reasons, the most critical being the continuation of development. Because the software stays with the company, and not the initial developer, there needs to be documentation in place so that subsequent developers can quickly pick up where the previous

one left off. This includes the need for thoroughly commented code, as well as separate documentation detailing the design approach, the technologies used, and any planned features are yet to be implemented.

As well as internal documentation, there needs to be adequate documentation for end users; although an aim of app design is to make the app usable without instructions, there still ultimately needs to be a guide for users, in the case that it is not obvious to them how to operate the app.

For the internal documentation, a report was written with the details of the app and its design, as well as an archive of all design and planning documents – mostly scanned from the pen and paper versions. The user-facing documentation was written in an instruction manual style, with step by step instructions. This documentation is bundled in with the app, and presented to users when they first launch the app. A challenge in writing this type of documentation is to ensure that the instructions are clear and simple, and free of technical jargon that the users might not understand. In the case of this app, jargon related to transportation engineering is appropriate, but software jargon is not.

Reflective Appraisal

Much of the work carried out at TDG built upon the skills I had learnt at university and at my previous position at Navico. In carrying out the general tasks, I was able to apply much of the practical knowledge I had gained through university, such as installing software via command line, and being able to critically inspect hardware, and quickly learn new protocols as required. This general work experience taught me that attention to detail is critical in a professional environment: tasks like setting the application defaults might have seemed superficial, but the financial benefits of avoiding workflow disruption would have been significant for any firm, but particularly a medium sized firm like TDG. I also saw the value of good organisation: as I saw how much of the computer equipment stored at TDG was no longer usable, I considered the efficiency gains of no longer having to deal with and replace faulty hardware, and knowing that hardware will simply work.

Carrying out Software Quality Analysis at TDG allowed me to put into practice many of the SQA principles taught to me by the team at Navico. In actually carrying out this analysis myself on software that I had not developed, I was able to gain an appreciation for why it is often

important to have a fresh set of eyes to test software: even the most skilled developers can overlook errors or miss important features. I also gained a sense for just how thoroughly software needs to be tested; even smaller scale applications can require many hours of testing to ensure that all bases have been covered, and as many bugs as possible are intercepted before the software is released to the end user.

In carrying out the sub-professional mobile app project, I was able to build upon many of the skills I developed at Navico. The app for TDG was much more complex than anything I had done previously, and so I was able to push my skills to a new level. I became fast at adapting and repurposing code, and using the many resources available to me to break through any brick walls I faced. Some of the more specific skills I gained during the project included becoming familiar with database management, and dealing with different types of media, like video, subtitles, and live camera feeds. Once again, I was able to learn a new programming language (Swift) on the job, and become proficient in a matter of days.

One of the key differences between the iOS project I undertook at Navico and the project at TDG was that while the Navico project was only ever intended to be a proof of concept, the TDG project was destined for a public release. This meant that I had to pay closer attention to usability and stability. By the time the app was ready to be released to TDG staff, many hours of on-road testing had proved the stability of the application, showing that I was indeed able to build an app suitable for public release. The project at TDG was also unique as I was a software developer working among civil engineers, and as such I learnt how important it is to have a wide range of engineering knowledge. In order to gather and interpret software requirements, I had to draw upon my basic knowledge of civil engineering principles, such as moving body mechanics. I also showed that I was able to read and understand literature written specifically for civil engineers, and interpret them for the purpose of software design.

Working at TDG, I gained an appreciation for the way that good social practices in the workplace led to a happy and hard working team of employees. During my first week I was surprised at how almost the entire office would stop working at the same time to take their break together. It was a chance for the entire staff to connect, and for me, a chance to get to know my new colleagues. The result was a team that got along well and collaborated efficiently. Several new staff joined the company during my time there, and I saw that they were instantly welcomed and quickly became a part of the team. This type of strong team

socialisation is especially important for firms like TDG, where projects often require a high degree of collaboration between staff with different skill sets.

Conclusions

Overall, my time at TDG was very rewarding. I was able to build upon many skills and develop them further, to a degree warranted by software intended for public release and sale. The time spent undertaking general work was particularly interesting, as it gave some perspective into how important lower-level technical tasks could be, and how critical it was to maintain a high attention to detail. Working at TDG showed me that teams with strong social links are efficient teams, and that this is especially important in engineering firms where projects often span many staff from different departments and disciplines. As a student nearing the end of my degree, the skills and perspectives gained from this work experience will be of immense value to me as I finish my studies and enter the workforce.

Bibliography

Carlson, P., & Mason, J. (1999). Relationship between ball bank indicator readings, lateral acceleration rates, and vehicular body-roll rates. *Transportation Research Record: Journal of the Transportation Research Board , 1658.*

New Zealand Transport Agency. (2001). *MOTSAM Appendix A3: Guidelines for the installation fo curve warning and advisory speed signs.* Retrieved July 20, 2015, from NZTA: https://www.nzta.govt.nz/assets/resources/motsam/part-1/docs/motsam-1-appendix-a3.pdf

Reiker Inc. (2014). *CARS: Curve Advisory Reporting System.* Retrieved July 20, 2015, from Reiker: http://www.riekerinc.com/Total-Solutions-CARS/CARS-PRO_Broch1.pdf

TDG Ltd. (2015). *Projects Portfolio.* Retrieved July 20, 2015, from TDG: http://tdg.co.nz/what-we-do/portfolio/

Witte, T. H., & Wilson, A. M. (2004). Accuracy of non-differential GPS for the determination of speed over ground. *Journal of Biomechanics , 37*, 1891–1898.