OULU UNIVERSITY OF
APPLIED SCIENCES

Charles Nebo

DEVELOPING IPHONE APPLICATION

DEVELOPING IPHONE APPLICATION

Charles Nebo
Bachelor's thesis
Autumn 2015
Business Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Business Information Technology

Authors(s): Charles Nebo
Title of Bachelor's thesis: Developing iPhone Application
Supervisors(s): Tapani Alakiuttu
Term and year of completion: Autumn 2015 Number of pages: 52

IPhone application popularity and demand are continuously on the increase since 2008 when Apple launched iPhone. Consequently, many applications are built targeting different user needs. But developing iPhone applications do have challenges. This research study explores current development study for developing iPhone application where the development environment and technologies are problematic. This paper intends to review some software development methodologies, methods, and techniques for the development mission. And try to develop iPhone application for release in the App Store using Model-View-Controller design pattern.

The objective of this thesis is to gain understanding of developing iPhone application to be released in the App Store using a software development process: focussing on functionalities Camera and Camera roll, Map, Email, collapse and expand of user interface using Xcode 5.x. The research methodologies applied explain five software development methodologies: Model-View-Controller Design Pattern, Agile Methodology, and Waterfall methodology, Requirement Elicitation Method and Mobile Phone and Telecommunication Method. And a Case study examining practical training experience for developing iPhone application in a software company named Taalvisio Oy.

The result found a functional user interface including camera and camera roll, map and email which shows that software development process is important for successful development of iPhone application. And also provides means for quality and better integration of iOS best practices. This research was carried out on behalf of a commissioner (ClouMob Oy) a medium size software Development Company who was deeply interested in applying software process in mobile application development.

Keywords: iPhone application, mobile application, mobile users, mobile developers, Software development process, Software development Kit (SDK), Programming environment

CONTENTS

1 INTRODUCTION

IPhone applications are programs running on iPhone device terminal capable of communicating and changing information with the services in the network. Its popularity and demand are continuously increasing since 2008 when the first iPhone device was launched. Not only has the use of applications eaten deep into the fabric of the society, it has become part of everyday life activities. Users are can perform online banking, online shopping, bill payment and social media interaction and storing sensitive documents while on the go. They are continuously demanding for more applications to saturate their individual needs.

In the business fora, the big and medium size companies no longer want intermediaries between them and their customers. Companies engage directly with their customers through applications to offer products and at the same time have online interaction with them. In the development arena, developers are those who create applications and services that utilize the functionalities in the device to meet the expected need of users. They need to figure out how to make these applications available to users. But to become a successful developer is not easy due to development challenges.

Firstly, most of the applications in iPhone device are relatively small, averaging several thousand lines of code, with one or two developers responsible for conceiving, designing and implementing the application. Secondly, developers adhere quite well to recommended sets of Apple best practices but rarely use any formal development processes to organize tracking of their development efforts and gather few metrics. Thirdly, there is a sharp divide between applications that run entirely on the iPhone device called native applications and web applications which have small device-based client with execution occurring on the remote server. Fourthly, developers need to understand the user groups in terms of technology adoption and product characteristics users' value.

Today we are living in a world where mobile applications make sense in our daily lives and communities. The buildings of recreational applications are paving the way to more complex business applications which require security, reliability and high quality applications. But the challenge is how to develop iPhone application of proven quality for end-users. The practice of developing application that is secure, reliable and of high-quality will be achieved by applying software development process. It ensures that development of applications is reliable, secure and

high-quality. Moreover, its techniques transfer easily to the application domain. And, resolves issues emanating from the iPhone device and evolving technologies with ease.

1.1 Research Question

The objective of this research study was to provide the current development study for developing iPhone application to be released in the App Store by giving answers to the question “What model is usable for developing iPhone application”?

1.2 Research Objective

The objective of this research is to develop iPhone application that can be released in the App Store using software development process; focussing on functionalities such as the Camera and Camera roll, Map, Emails, Collapse and Expand of user Interface using Xcode 5.x. This application is developed using Model-View-Controller Design pattern. The development process targets the following objectives: Developing iPhone application that is robust, reliable and of high-quality using MVC design pattern; and ability to target and resolve issues of device limitations and evolving technologies.

1.3 Research Methods

The research methods applied include review of five software development methodologies, methods and techniques as presented in the industry and literature and case study examining practical training experience for developing iPhone application in Taalvisio Oy. These methods are selected because software engineering techniques transfer easily to the application domain. Moreover, it targets and resolves issues emanating from the iPhone device and evolving technologies with ease. Adding more to this, it makes applications to be more secure, robust and high-quality. The software development methodologies reviewed were as follows: Agile method, Waterfall method, Requirement Elicitation method, Model-View-Controller Design Pattern and Mobile Phone and telecommunication method. The aim was to select appropriate software development model for developing iPhone application.

1.4 Research Limitation

There are few limitations in this research. This research study does not aim to be complete in its coverage of review analysis but aims to focus on reviews of methodologies based on methods and techniques. Moreover, this study will not discuss user experience which is affected by industrial design issues related to applications.

The remainder of this paper is organised as follows. In Chapter 2, the background knowledge, related works and frameworks are presented. While chapter 3 describe the theoretical study which includes the software development methodologies. Chapter 4 describes the development processes enabling application to be released in the App Store and chapter 5 describes the case study for developing iPhone application in Taalvisio Oy. Chapter 6 presents the result with screen shots of functional app designs while chapter 7 gives the summary of the main conclusions. Chapter 8 presents discussion on result and future research.

2 BACKGROUND INFORMATION

Mobile applications have been around since the 20th century, during this time mobile applications were controlled by either mobile network operators or the device manufacturers. The end-users were given the opportunity to subscribe and download games, ringtones editors, calculators, calendars, wallpapers and other utility applications. But the processes involved in the application installation and downloading were horrible and unintuitive.

In 2007, Apple launched iPhone device, a high-level computational device with virtual keyboard, touch based user interface, and sensor technology which revolutionized the way people communicate (iOS Developer Library 2015, date of retrieval 13.5.2015). Besides the traditional voice call, users enjoy gaming, built in camera, audio video playback and recording, instant-messaging, emails, and internet browsing and social media browsing just to mention but few. IPhone device was subsequently optimized for internet communication and usage.

IPhone device release in 2007 was closely followed by the launch of Apple Store in 2008 a distribution channel capable of reaching every single iPhone user and manage other Apple products. The App Store is frequently changing as the iOS platform continues to improve in performance, thus rapid changes exist due to high demand for iOS applications. App Store integrates with iPhone devices so that application download is easy and user experience is optimized. With iPhone application, users can facilitate and execute transactions, place orders, make electronic payments and communicate with voice calls and different social media channels.

Statistically, there are approximately 1.2 million applications in the App Store and five hundred thousand of them belong to the iPhone category. Recent report showed that seventy-five billion applications were downloaded by users in 2014 (Statistics portal 2015, date of retrieval 13.2.2015). The current report released by forester research predicts the revenue generated from users buying and downloading applications for mobile device will reach $38 billion in 2015 and the number of users are likely to increase in the near future (Bilton 2011, date of retrieval 14.6.2015).

The literature research on iPhone application is an active research. Eminent scholars hold their views about the drive for mobile applications. They argue that mobile application business consists of users who have mobility related needs such as communication, technology and services.

Technology provides hardware and software infrastructure needed to offer services to end-users. The primary providers in this area are the device manufacturers and network equipment vendors while the secondary providers are the retailers, component makers and platform vendors who provide software platforms such as the operating systems and development platforms (Camponovo et al., 2003, date of retrieval 18.1.2015).

According to (Von Hippel, 1986, 2005, 791-805) technology brings value to the user when it is taken into use because it has the tendency to change the society when it diffuses into the market place and change users' behaviour. Technology can be used to develop applications because users are in need of them. However, it requires participation of lead users in developing them.

In technology acceptance model (Davis, 1989, 319-340) agues the main factors explaining adoption of an application or service as perceived ease of use and perceived usefulness. The adoption of application or new technology by end-users can take some time, but end-users would adopt application or new technology only when they are sure it would satisfy their needs. Adoption of application is accepted if users believe it is useful (perceived usefulness). Conversely, adoption of application or service is rejected if users are constrained by the system in maximizing performance of the application or services (perceived ease of use).

2.1 Users and Developers Point of Views

The layman point of view of mobile users is those who make and receive calls. However, in recent times, some users can send and receive text messages, multimedia messaging service (MMS), email, internet access, short range wireless communication, infrared Bluetooth, business applications, gamming and high level computing abilities (Wikipedia, 2015, date of retrieval 13.5.2015).

Recent studies argue that mobile users are characterized as those with needs. Mobility is their most important need. It has characteristics such as ubiquity, localization, reachability, convenience, instant connectivity and personalization. Mobility gives the user the freedom of movement to use mobile services on the go; ubiquity gives the user the possibility of using mobile services anywhere independent of the user location; localization enables users' location information to be exploited so that services can be offered; By reachability, users can be reached anywhere anytime from selected persons and context; convenience enable users to be always at hand; instant-connectivity

imply that user services must always be on; and personalization ensures that users can use their device to store personal information (Camponovo et al., 2003, date 18.1.2015).

In addition, users argue for more user experience from the mobile device. They want bigger screens, increased memory, and high speed processing capabilities, memory capacity and increased power reserve.

On the other hand, developers are those that create applications and services which utilize the functionalities in the device to meet the expected need of the end-users. Developers use software development kit (SDK) to design commercial applications and walk away with plenty of cash in their pockets. The SDK is a closed source and as a consequence requires developers play by Apple rules. But the monetization aspect energizes and motivates both the individual developers and third party software developing companies to creating innovative applications for users. These applications are released through the App Store to enable users increase the utility associated with the usage. Most encouraging is that other mobile device makers such as Samsung, Google, Blackberry and Microsoft have their App Stores.

App Stores have become one of the fastest growing businesses in history. They are owned by device makers whereas the developers provide contents that increase the value of the App Stores. Developers are interested in selling applications through the App Store because App Store creates a good medium to reach many users worldwide. They would set the price of their apps to approximate to 70% of the revenue transaction whereas Apple retains 30% of it. Consequently, with many App stores abound, developers are energized as they try to catch up with new business opportunities to provide contents to the App Stores. And at the same time decide which mobile platform to join and provide content or risk cross platform problems. And this decision usually affects app success in the App market.

It is argued that developers need to understand the platform of choice in order to effectively provide contents to the App Stores. Being a successful developer is not as easy as one could guess because in the application and services development markets, professional knowledge of software life cycle is simply not enough. Developers must understand the domain constraints arising from the mobile platform, evolving mobile technologies, environment particularities and device limitations in order to effectively meet the needs of the end-user (Alahuhta 2011, 72).

2.2 Research Framework

Figure 1 shows conceptual framework for this paper. Framework consists existing theory that is used for this particular study. It demonstrates understanding of theories like Agile methodology, Requirement Elicitation Method, Mobile Phone and Telecommunication Method, Model-View-Controller (MVC) Design Pattern and Waterfall Methodology. Understanding the methods or models is relevant to the research question which is under investigation. Framework also demonstrates concepts that are important to the topic which relates to the broader areas of knowledge for consideration. It represents the knowledge embedded in the development process, design, wireframing, iOS application and Xcode 5. X and UI (User Interface). These are needed to design BizinfoApp for a company Taalvisio Oy iPhone application. Additionally, the conceptual framework also consists functional screens which are part of development, design and final result of the created BizinfoApp.

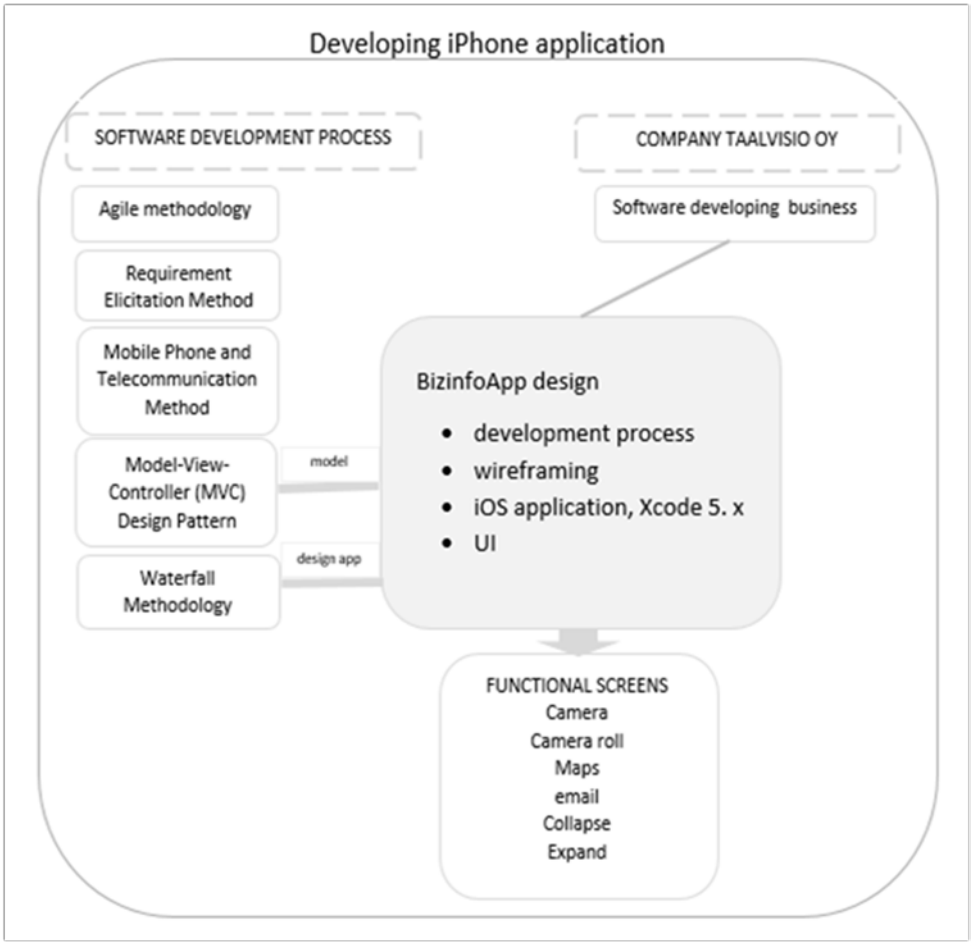

FIGURE1. A Conceptual framework of the study

3 THEORETICAL STUDY

The theoretical starting point for this research study is based on review of software development methodologies, methods and techniques as presented in the industry or literature such as Waterfall method, Agile method, MVC design pattern, Requirement elicitation method and Mobile Phone and Telecommunication method. These theories are selected because software engineering techniques transfer easily to the iPhone application domain. Moreover, it targets and resolved issues emanating from the iPhone device and the evolving technologies. In addition, it makes development of application more secure, reliable and high-quality.

The theoretical reviews follow this order. The theoretical concept is discussed first, followed by the methods and techniques and continue with advantages and drawbacks in each model if any.
A Software process modelling is an abstract representation of reality that excludes much of the world infinite details. It is a development process which developers follow during software development. Its objective is to determine the order in the development process and evolution and to establish transition from the current phases' analysis and coding. Each model is described by sequence of activities. The development phases may vary in each model, but the model will always include: (Valacich et al. 2004, 21).

- Planning
- Requirement
- Analysis
- Design
- Development
- Testing
- Maintenance

In the next chapter, I will discuss some software process methodologies like Waterfall method, Agile method, Model-View-Controller design pattern, Requirement Elicitation method and Mobile Phone and Telecommunication methods respectively.

3.1 Waterfall Methodology

The concept of Waterfall model refers to the first software development Lifecycle (SDLC models) to be introduced in software development in the 1970s. It is a process model which provides sequential flow approach to software development process. It was introduced at a time when software engineering hardly succeeded in making complex software systems. But it helped to eliminate many difficulties encountered in software projects and become the bases for most software acquisition standards in government and industry.

Method / Technique

Waterfall approach to software development emphasize that any phase in the development lifecycle can begin when the previous phase is completed as seen in figure 2 on page 16. Its technique helps developers to re-visit earlier completed phases in order to review and verify the operation as required. But with this model, testing can only start after the project is finished. Furthermore, Waterfall model is most effective when applied to large or complex project with well-defined requirements. The major disadvantage of waterfall model is that it emphasizes on extensive revisions and refinements which generate elaborate documents as criteria for early requirement and design. (Boehm 2006, 61-72)

Some advantages and Drawbacks of Waterfall Model

There are some advantages in using Waterfall model. One of the advantages is that the framework shows some reliability since each phase must be completed before the developer goes to the next phase. The sequential approach is to ensure each phase is error free. Secondly, the framework enables developers to re-visit earlier phases especially in times of need to make some corrections. Thirdly, its sequential nature stipulate that the analysis phase and design must be done at the initial phase to enhance quality and measured progress of applications as shown in figure 2. Conversely, the drawback of Waterfall model is that the sequential development maintained in the framework has strict sequential flow which does allow changes to be affected in the system only at the end of the project. Waterfall critiques cling on the sequential approach as a major flaw in this model. For instance, testing of the application or product can only begin when all the phases have been completed. Other drawbacks include huge documentation generated from the system processes which can become a problem especially when these documents are incomprehensible. In addition, missing requirements specified by users might be difficult to address. Also, it is difficult to decide

the beginning and the ending of a development phase and this most often than not contribute to project delay.

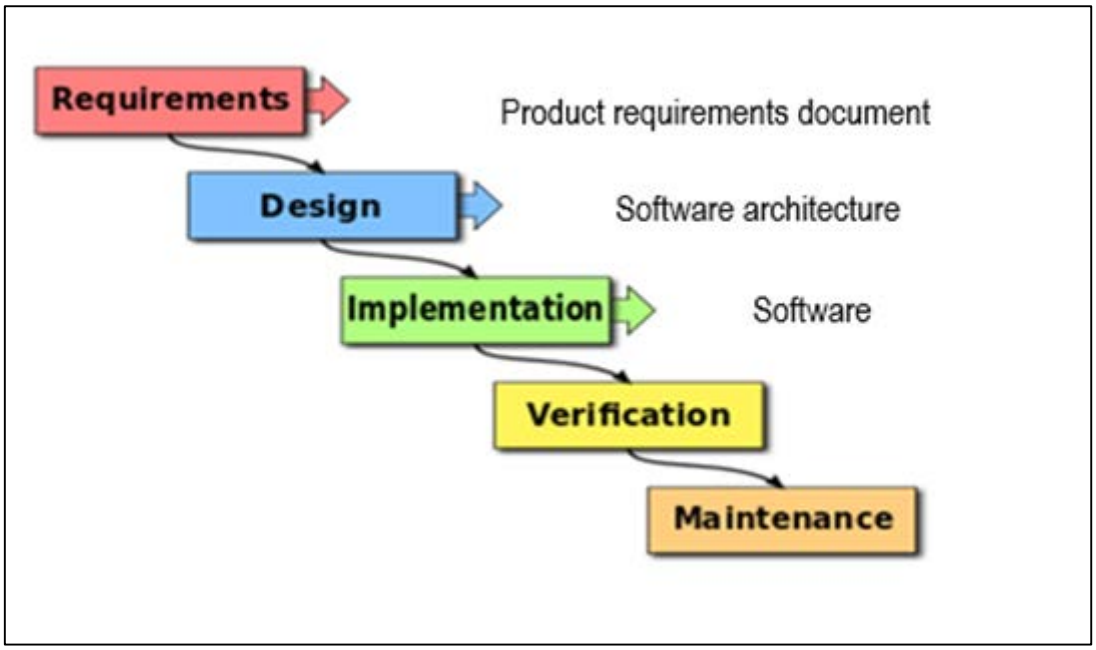

FIGURE 2. Waterfall development model (Wikipedia 2015, date of retrieval 13.5. 2015)

3.2 Agile Methodology

The concept of agile method refers to an evolutionary approach in software development lifecycle (SDLC) which became popular in the 1940s through Lean manufacturing and Agile manufacturing in the 1990s. Its major contribution to software development includes strong emphases on adaptability of enterprise to a dynamic environment (Salo 2006, 2). Agile vision statement is enshrined in its agile manifesto and states that individuals and interactions are more important than processes and tools, working software is more valuable than comprehensive documentation, customer collaboration is preferred over contract negation and adaptability is valued higher than creating and following a plan (Agile Alliance 2001, date of retrieval 23.4.2015).

Method / Technique

The fundamental principle of agile method is simple design, large number of releases in a short time frame, extensive use of refactoring, pair programming, test-driven development and seeing change as advantage. This is achieved by the combination of iterative and incremental processes

with focus on adaptability and customer satisfaction and rapid delivery of software product as seen in figure 3 (page 18). The Iterative and incremental development processes include a software production which is broken into incremental builds. These builds are included into the number of iterations which basically involves team members who play different roles in the project such as planning, requirement, analysis, design, and coding, testing, deployment. At the end of the Iteration, a working product is displayed to the customer and stakeholders (Boehm &Turner 2003, 16).

Project Document

Agile project document includes the vision statement which details the road map in clear terms. Others are planning document which is used for identifying the next step of the product future and launch date; sprint document which shows the minutes of meeting for each sprint and how team members commit to their goal by identifying requirements that support the goal and the individual tasks it will take to complete the goal; sprint review is a review that takes place between 10-15 minutes and details of what team members have achieved the previous day, work schedule for the current day and discussing problems as they emerge.

Project Management

Agile project management may include team members, product owners, scrum master and agile mentor. Team members are responsible for creating the product while the product owners are among the stakeholders of the project. The scrum master is the project leader who plays additional supportive role to the team by clearing road blocks and makes the agile process to work. Agile mentor has tacit knowledge of implementing agile project and shares the experience with the team members.

Some advantages and Drawbacks of Agile Model

There are some advantages of agile development Model. One of the advantages is that agile projects are most effective in small projects with novel activities. It encourages fast delivery of finished products. Agile process is geared towards customer satisfaction by rapid changing design as demanded by users. Others include adapting to changing circumstances; reduction of cost due to fast delivery of finish product; the performance of the functionalities is tested early. Missing requirements are always addressed. Innovation and flexible software designs are encouraged. Users have direct participation in defining the human computer interaction.

Conversely, its drawbacks include the fact that frequent requirements are expected; Documentation for future developments is neglected. Costly iteration can add to the project budgets and schedules;

thus the additional cost can outweigh the benefits. Because Agile has no architecture, projects following agile method may face unknown risks which can affect the development of the project.

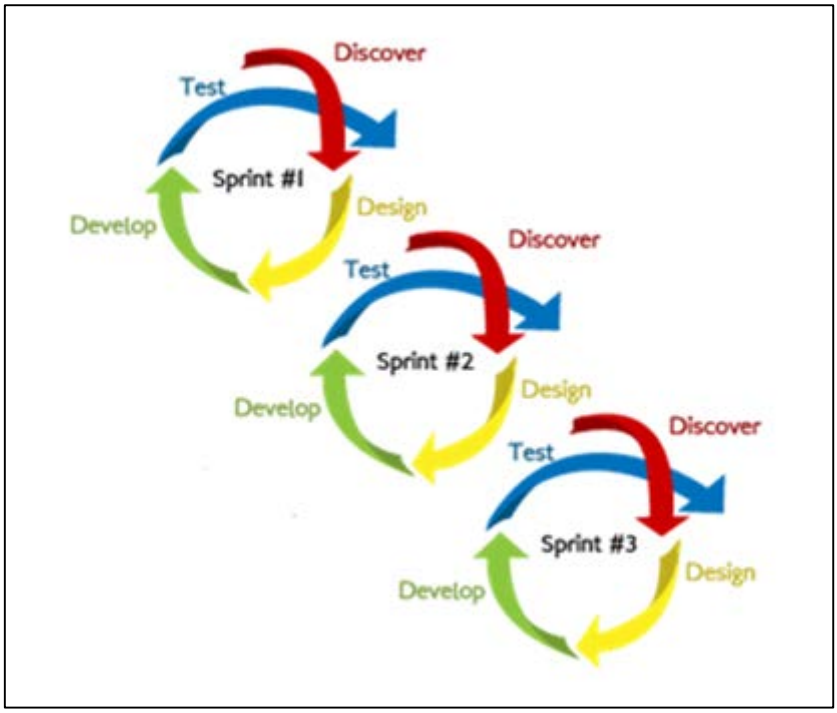

FIGURE 3. Agile development model (Saif 2012, date of retrieval 18.2 2015)

3.3 Model-View-Controller (MVC) Design Pattern

The concept of Model-View-Controller design pattern refers to the three roles or responsibilities that are needed to display a screen to the user. It is important because all iOS applications follow this design pattern. Furthermore, it represents how the communication between those components such as Model, View and Controller work. Model-View-Controller (MVC) design pattern was developed by Norwegian computer scientist Trygve Reenskaug in 1979 while in Xerox Palo Alto research centre. Model-View-Controller (MVC) design pattern made its first debut in the design of Smalltalk-80, since then, it had become so popular in the design of interactive Graphical user interfaces in many software industries (Wikipedia, 2015).

MVC is important because not only do all iOS applications follow this pattern, but it provides a high degree of user interaction with graphical user interfaces. Furthermore, its objective is to enhance usability of applications. The concept behind Model-View-Controller was achieved through division of interactive application into three components such as the Model, the View and the Controller. These components work together to carry out the functions of application and how each component

is described by the class. In order to display anything on the screen for the user to see, there is the need for the components to work together in three roles.

View Layer

The view is the user interface consisting of the graphics, button, sliders and tables. The view is responsible for displaying data to the screen from the controller and receives user gestures and relaying them back to the controller. The view never directly communicates with the model. Figure 6 (page 27) indicates there is no data flowing back and forth, no method calls, no interaction between the view and the model.

Controller (Application Logic Layer)

The controller consists of the logic and glue that binds the model to the view. It is responsible for updating and receiving information from the model and handles user interaction from the view.

Model Layer

The model is responsible for retrieving data from whatever data source that powers the application. It does not have to know how to display the data that it contains, but makes sure that it encapsulates all data thereby fulfilling one of the principles of object oriented design.

Methods/Techniques

These three components play different roles that define the way objects communicate with each other. The role of the controller object is to sit between the model object and the view object and serve as the middle man between them. The controller object gets data from the model object and feed the data to the view object. If user interaction occurs, information is passed from the view object to the controller object. Furthermore, when data is to be updated, the controller object tells the model object what data is to be updated. These are the hard and fast rules, however, if these rules are not followed, the application will still work but this is the best practice. And by doing so, it allows swapping in and out components easily. For instance, if you want to change the view, then swap with different views and there is no need to make changes in the model. In the same vein, model can be swapped with another class or another data source and it will not affect the view. In this manner, loosely coupled system is maintained or the components are decoupled say to speak. This is important because there is need to make each component as re-usable and easily, swappable and maintainable as possible. Following the Model-View–Controller design pattern helps promote code maintainability and re-use. The advantage of adopting MVC design pattern is

that many objects in this application tend to be more re-useable and their user interfaces tend to be more defined. Applications designed with MVC design are extensible than other applications. Furthermore, cocoa technologies and architecture are based on MVC design (iOS Developer Library 2015, date of retrieval 14.6.2015).

Some advantages and Drawbacks of MVC

The separation of components makes it easier to design applications. It enables swapping of components in and out as desired. The model can be displayed in a variety of ways and the controller and view can grow as the model grows. It is easy to reuse and maintain. Most importantly, it provides high degree of user interaction. Also helps to maintain robust, secure and understandable code.

Conversely, the drawback is that the separation of concern of different entities in the MVC design may lead to tight coupling in a large project.

3.4 Requirement Elicitation Method

The concept of requirement elicitation techniques refers to methods used by analysts to determine the needs of customers and users so that software system can be built to satisfy needs. The objective of requirement elicitation is to gain knowledge of the problem domain with a view to producing a requirement model. It addresses the issues of conflicting requirements and enables software systems to attain greater degree of usefulness and stability. Elicitation is an iterative process and cannot be done without recognizing the actors who are called the stakeholders (Zhang 2007, date of retrieval 17.8.2015).

The stakeholders

Stakeholders are actors in the elicitation process. They are people of different backgrounds, skills and knowledge and have distinct ways to store, recognize and express their knowledge about the problem domain. They include the end-users, customers, decision-makers and the work context. Software requirements must be extracted through them during the development process because they have the tacit knowledge of the problem domain. The problem domain is the source of all requirements and good understanding of them ensures the stability and usability of the software system.

Requirement Elicitation Method

In requirement process, elicitation can start with the help of the stakeholders such as users, customers, decision makers and designers including the work context. Stakeholders represent people from diverse background and with vested interest. Because the work context changes, elicitation process takes place in sessions with focussed groups. Each session involves interviews, goal analysis, task analysis and scenario analysis applied in-line with the work context. Requirements in each session are extracted, recorded and analysed. The result is the conceptual model (see figure 4 on page 23). Thus, these activities are repeated consistently as required until the desired system is reached and documented as requirement specification.

Requirement Elicitation Technique

In a requirement development process, stakeholders and development team must communicate so that domain knowledge relevant to the system development is extracted, recorded and documented. The domain knowledge can be extracted through varieties of techniques that facilitate the sharing of knowledge and promote communication. Thus, there are four elicitation techniques which are currently discussed in literature namely

- Observation method
- Conversation method
- Analytic method
- Synthetic method

These methods present a specific communication model between the development team and the stakeholders but the development team have the right to decide which method is applied based on the situational context.

Conversation Method

Conversation method is the verbal communication between the stakeholders and development team. It is a natural way of expressing needs and ideas in face to face contacts. The process involves extracting relevant information during an interview, workshops, brainstorming or using focussed groups. The main objective is to communicate relevant information concerning the problem domain. It is considered the best way to elicit requirements in a face to face contact but labour intensive due to huge documentation. It involves setting up meeting, analysing documents and ad hoc duties in between.

Observation Method

The observation method is a self-study of the domain environment by the development team with a view to understanding the work context and human activities. Sometimes it is difficult for the development team to articulate how work is done in an environment. Observation method becomes handy as the development team immense themselves in the situational context in order to obtain evidence that help understand the pattern of work. Furthermore, observation method is deployed if stakeholders have difficulty articulating their needs, especially when the development team are seeking for better ways of understanding the work context. It is an effective way of gaining understanding of the work context and how users perform their work. It increases the analysts' familiarity with the culture and style of the people in organisation. It is easy to apply; it allows analysts to have direct assessment of the work domain and how work is performed. The disadvantage is that the stakeholders are easily offended and may change their behaviour depending on the attention they get from the development team.

Analytic Method

Analytic method is a method for extracting requirements from existing documentation. It comes from two distinct sources such as requirement re-use and documentation studies. It captures knowledge that is not directly expressed by domain experts. By reading the existing documentation, developers can understand the requirement tasks embedded in the workflows and product futures. The disadvantage of this method is that it does not capture requirements directly from users and customers.

Synthetic Method

Synthetic method is the combination of conversation, observation and analytical methods into a single method. The objective is to improve communication between developers and customers in the requirement process. It consists of scenarios, storyboards, prototyping and JAD sessions. (Zhang 2007, date of retrieval 17.8.2015)

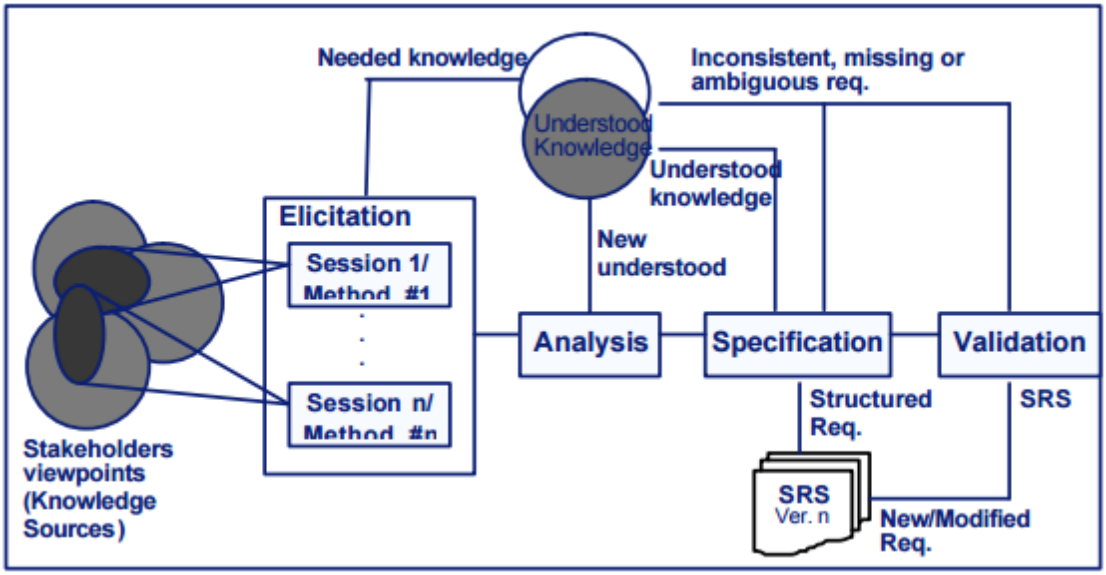

FIGURE4. Requirement development process (Zhang 2007, date of retrieval 17.8.2015)

3.5 Mobile Phone and Telecommunication Method

The concept of mobile telecommunication refers to the process of sending, transmitting and receiving voice and data information in space for the purpose of communication. The basic components of this communication system are the transmitter, the channels and the receiver. The transmitter initiates the signal through a channel that delivers it to the receiver. But without the radio spectrum, there is no signal. So what is radio spectrum?

Radio spectrum has a link to how mobile telecommunication technology was developed. It is defined as the electromagnetic waves which are transmitted through space. The objective is to provide coverage as the mobile user move from one cell to another in the system (Gruber 2005, date of retrieval 20.1.2015).

The architecture of mobile telecommunication

The architectural design that defines mobile telecommunication system has five main components namely: radio base station or radio access, network switch, subscriber database, telecommunication network and the mobile terminal.

The telecommunication network is the traditional fixed telecommunication system and responsible for sending and receiving voice calls and text messages with the aid of the network switches and subscriber database. The network switch which is attached to the telecommunication network is

responsible for routing calls to and fro the mobile terminal as the mobile user moves from one cell to the other. While the subscriber database is responsible for preparing and distributing bills incurred by the mobile user.

Method / Technique of Mobile Network

Firstly, a metropolitan city or rural area where the radio base station is situated is divided into smaller cells in order to provide coverage to those cells. Then the radio base station or the radio access, which is responsible for monitoring the mobile user, is attached at the top of the gigantic hexagon grid of the base station as shown in figure 5 (page 25). As the mobile user moves from one cell to the other, radio signal is received and transmitted to and fro the base station and the mobile terminal. For the mobile terminal to accommodate many calls simultaneously, the cells have to be smaller in order to sufficiently achieve maximum channel re-use. This is the concept that define mobility in wireless communication (Gruber 2005, date of retrieval 20.1.2015).

Evolution of Mobile Technologies

Since the discovery of Mobile telecommunication system, several successful technological progresses had been made to enhance the evolutions using mobile technologies. The aim was to increase the number of users than with any other communication technology. For instance, the 1G system was the first mobile system in the world. Its objective was to provide analogue voice calls to users. But roaming was an outstanding problem in this system. Secondly, 2G systems provided the digital voice calls with short messages and better data services and roaming possibilities. However, there was the need for system interoperability and global standard in the world. Thirdly, 2.5G was the enhancement from the 2G and the 3G. It provided higher data rate and resolved the problem of global standard and interoperability. Fourthly, 3G systems came with higher data rate. It provided services such as voice and video calls, and broadband wireless data. The 4G systems provided efficient and faster transfer of data (Gruber 2005, date of retrieval 20.1.2015).

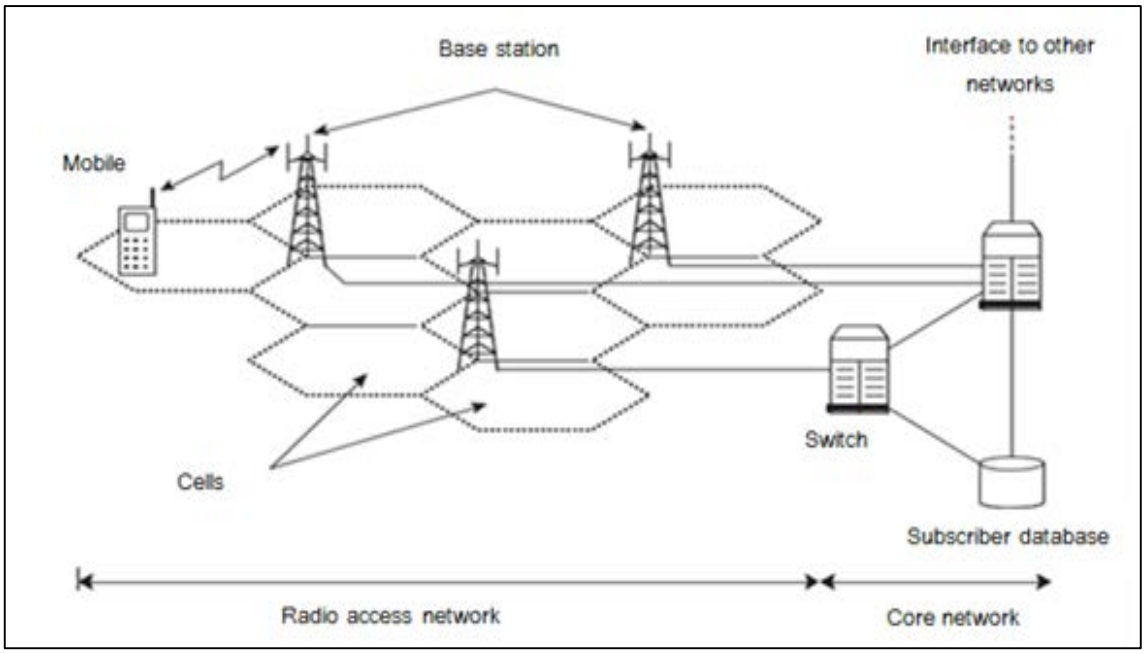

FIGURE 5. Architecture of Mobile Telecommunication System (Cox 2008, date of retrieval 17.8.2015)

4 DEVELOPING IPHONE APPLICATION FOR APP STORE

This chapter shows a simple iPhone Application design and some functional requirements designed for Taalvisio Oy which demonstrates how to develop iPhone application which can be released in the App Store using software development process. The functionalities consist of Camera and Camera Roll, Map, Email, Collapse and Expand of User Interface (UI), and storyboard using Xcode 5.x.

4.1 Application Model

The development model for this app is the Model-View-Controller design pattern (MVC) while the development process is the Waterfall model. The selection was based on the review of software development Methodologies and best practice for developing iOS applications. MVC design pattern refers to the three roles or responsibilities that are needed to display a view to the user and how the communication between these components works. It is separated by three important layers, Model layer, View layer and the Controller (logic layer).

View Layer

The view is the user interface consisting of the graphics, button, sliders and tables. The view is responsible for displaying data to the screen from the controller and receiving user gestures and relaying them back to the controller. The view never directly communicates with the model. Figure 6, below shows there is no data flowing back and forth, no method calls, no interaction between the view and the model.

Controller (Application Logic Layer)

The controller is the logic layer and glue that binds the model to the view. It is responsible for updating and receiving information from the model and handles user interaction from the view.

Model Layer

The model is responsible for retrieving data from whatever data source that powers the application. It does not have to know how to display the data that it contains, but makes sure data is encapsulated thereby fulfilling one of the principles of object oriented design.

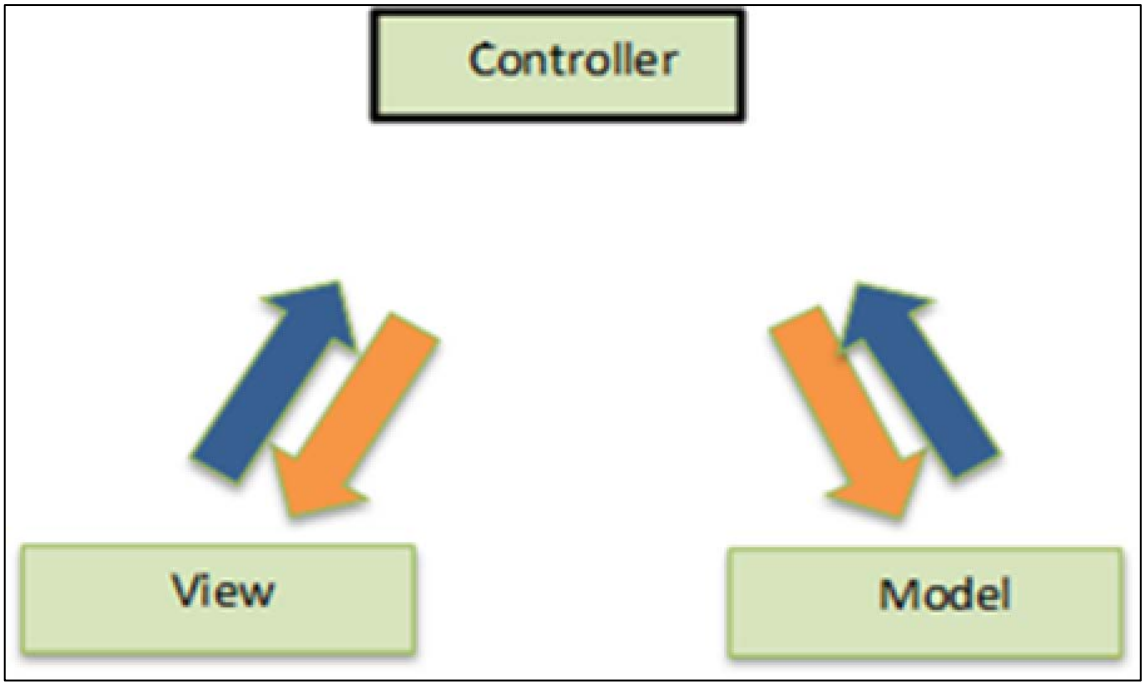

FIGURE 6. Model-View-Controller (MVC) Design Pattern Model

4.2 Designing Application Using Waterfall Development Process

The app designed follows Waterfall development process. It is a process developers' use for completing a project. Waterfall development process approach enable developers to go from one phase to the other and to re-visit completed phases in order to review and verify the operation as required. The phases include the Business Gathering, App requirements, Technical Verification, Functional Verification, Wireframing and Design Composites, Implementation and Testing. The advantage of using Waterfall development process for the App design is because it ensures that developments of applications are reliable, secure and high-quality.

4.2.1 Business Requirement Gathering

Business requirement gathering enable IT professionals to determine the needs of the customers and users so that software system can be built to satisfy user needs. The goal of requirement gathering involves working with the client to determine what is going to be built by asking

thoughtful questions. And by combining expertise with the goal of the client to arrive at common set of expectations on how the application is going to function.

4.2.2 Application Requirements

App requirements show the goal of the application to be built and the target functionality of the app. It must be defined at the beginning of the app development. The goal of BizinfoApp was to connect people through exchanging of information. The functionality for this app as I have mentioned earlier will focus on camera and camera roll, map kit, email, collapse and expand of user interface. The developer chose this goal because of the need to help people who struggle to connect with people. The project name for this App is called BizinfoApp and will henceforth be referenced as BizinfoApp throughout the remainder of this paper. Summary of the features of BizinfoApp is as follows:

- Slide out Menu Screen
- Camera and Camera Roll Screen
- Map Screen
- Email Screen
- About this App Screen
- Collapse and Expand of UI
- Rate this App, submit user feedback
- Analytic Ads

4.2.3 Technical Verification

The goal of technical verification is to uncover any potential road blocks or workarounds that might need to be taken care off. This phase validates the functionality and requirements earlier decided in the business requirement phase to make sure it is feasible.

In BizinfoApp design, the developer went through each of the requirements and at least makes sure that the BizinfoApp idea and how to accomplish it were brainstormed. When there were objections or doubt about whether it was possible or not, then the developer had made some online research to ascertain how people would have done it. Also when there was doubt about how the BizinfoApp performance would be, the developer had created a prototype and try to mimic the scenario to test how the BizinfoApp was going to perform.

4.2.4 Functional Specification

The functional specification breaks the user experience down into screens and describes the functionality in each screen. Basically it is about how the app is structured in terms of screen flow. Functional specification is also about documentation that describes the functional behaviour of system development. The documentation describes the functionality needed by the system users and the input and output properties of what the app will do in any particular scenario.

In BizinfoApp design, the developer had concentrated on how the BizinfoApp would be laid out by screens but not how the screen will look like or where all user interface will go (this will be the information architects job). The developer had described the functionality of the screens in great detail so that it would be easily referenced by the programmer when building the app. In addition, proper documentation helped the developer to keep projects organised and make functional specification easy to understand by the screen flow. Below is the functional specification description of BizinfoApp.

BizinfoApp: Description of functionality by Screens

1. Menu Screen
    - Access to the slide out menu
    - Display the category of all apps

2. Camera and Camera Roll Screen
    - Access to the slide out menu
    - Display the category (Camera, Get picture, and Save)
    - Display buttons for the Camera and Get picture and save.
    - Display “No image” and “number of images” to the user.
    - Display Skip button (navigation)
    - Display remove Ads (connects with Ads agency for advertisement)
3. Map Screen
    - Access to slide out menu
    - Display text Field and search button
    - Display map category (hybrid and standard)
    - Display buttons for Zoom and Map Type
    - Display skip button (navigation)
    - Display remove Ads (connects with Ads agency for advertisement)

4. Email Screen
    - Display send Email button
    - Display skip button (navigation)
    - Display remove Ads (connects with Ads agency for advertisement)

5. About this App Screen
    - Display the developer information and version number and history
    - Update and change log
    - Send email Button (contact to the developer)
    - Display remove Ads (connects with Ads agency for advertisement)

6. Slide out Menu
    - Display buttons for:
    - Camera (Goes to camera and camera roll screen)
    - Map (Goes to map screen)
    - Email (Goes to email screen)
    - About (Goes to about screen)
    - Give feedback (Launches the email client)

4.2.5 Wireframing

Wireframing is the transformation of the functional specification of the iPhone application into different screens so that it can display the business objective of the organisation.

The aim is to indicate how the elements of the application will be laid out in the user interface. And creating wireframes before starting app development will prompt discussions, questions and force the developer to think about how the user will use the application. It also provides visual understanding of the business objective on each screen. The functional elements on each screen is arranged in a way that is intuitive to the user and yet flexible to be functional as well. Wireframing was done bearing in mind the best practice stipulated by Apple in iOS 7 Human interface Guidelines (Apple, 2014 iOS 7 Human interface Guidelines, date of retrieval 12.9. 2015). It provides the developer the opportunity to make structural changes that might be costly outside this phase.

In BizinfoApp design, the developer used pencil and paper to layout the elements on screen because it is easy to edit and allowed the developer think better. The aim was not to make a world of art but to produce a sketch of how the elements are laid out in the screens. Afterwards the developer reviewed the wireframes with people and got common understanding with the client. At this stage, it was easy for the developer to collect feedback and make necessary changes that would have been costly outside of this phase. Below is the sketch of BizinfoApp Wire frames using paper and pencil (figures 7-12).

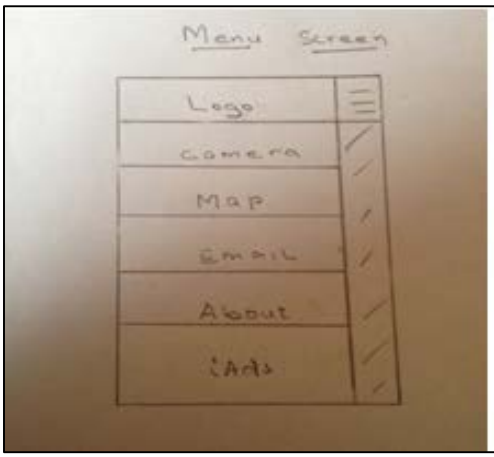

FIGURE 7. Wireframing Menu Screen

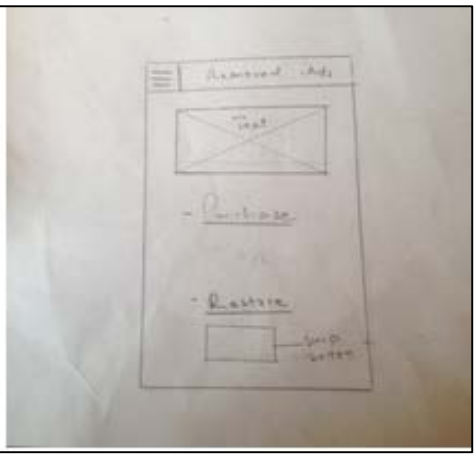

FIGURE 8. Wire framing Removed iAds Screen

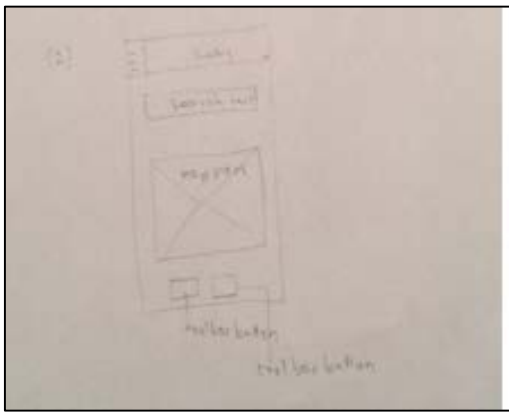

FIGURE 9. Wireframing Map Screen

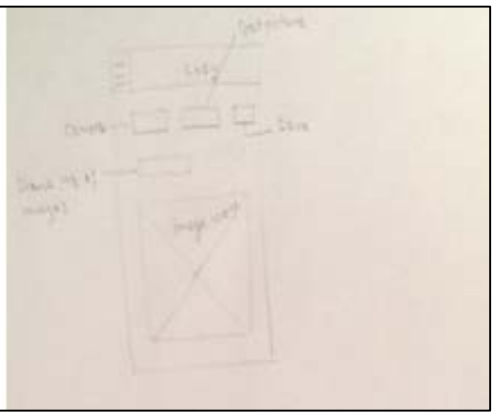

FIGURE 10. Wireframing Camera Screen

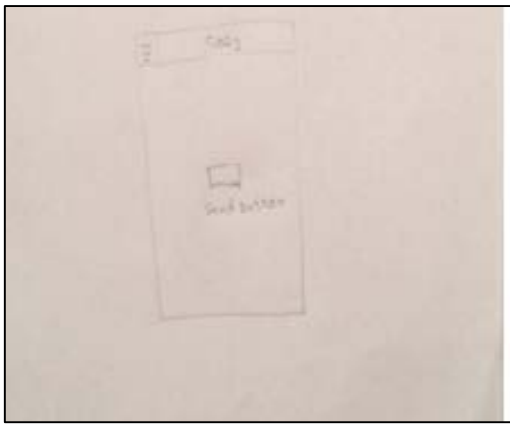

FIGURE 11. Wireframing Email Screen

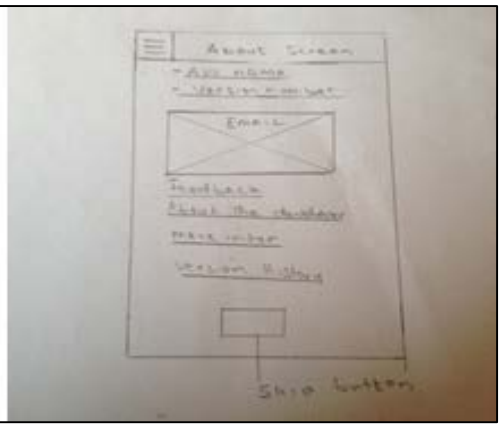

FIGURE 12. Wireframing About Screen

4.2.6 Integrating Design Composites for the Application

The design composites take the wireframes to a new level. The designer takes the wireframes and uses them as a framework for integrating designs for the app. Many designers use tools such as Photoshop, sketch or illustrator for this task. The design composites produced will be a professional design app reminiscent of the app the client will get at the end of the development.

In BizinfoApp design, the design composites consist of the Logo, Icons, Branding, and wire framing of the BizinfoApp Screens. The developer used graphic tool such as Photoshop to accomplish this task. The Logos, Icons, branding, colours, sizes and resolution were carefully designed according to iOS 7 Human Interface guidelines published by Apple. Making sure that the human interface guidelines were followed was important because Apple wanted the look and feel of its applications to be similar in accordance with its studies and research, otherwise the App will be rejected from entering into App Store. Below figure 13-17 show design composites for BizinfoApp using Photoshop.

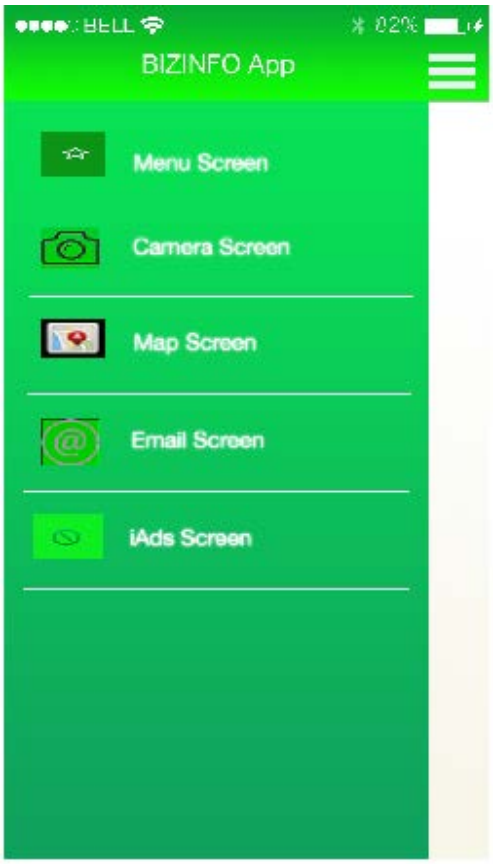

FIGURE 13. Design Composite for Menu Screen

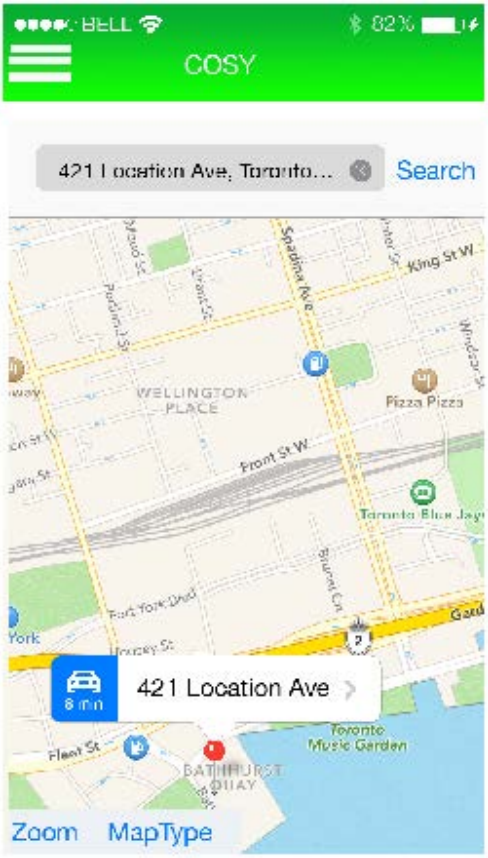

FIGURE 14. Design Composite for Map Screen

FIGURE 15. Design Composite for Camera Screen

FIGURE 16. Design Composite for Email Screen

FIGURE 17. Design Composite for App Logo

4.2.7 Implementation

The development phase is where the coding happens. It consists of three layers namely Model Layer, View layer and Controller (logic Layer).

The Model Layer

Xcode is the programming environment for coding, testing and bug fixing iPhone applications. It is made up of the programming language called objective c for writing codes or commands that tell the app what to do. Xcode has the same structural design with Model-View-Controller design pattern which is why it is the application programming Interface (API) of iOS applications. It is made up of App Delegate class, View controller class and main storyboard class. The model class is left for the developer to create and customize. The MVC design pattern contribution to software development has been discussed in this paper (see chapter 3).

A. Obtaining Data Model

A data model is the data that power the app. The data model for BizinfoApp is the menu items such as the camera, map and the email. These menu items are hardcoded in the model layer of the application.

B. Modelling Data

In modelling the data, chances are the data model would be a listing of some menu items that are displayed on the screen for the user to see. In this regard, the developer thinking was to use classified listing of data. While Table View was used to display the menu items for the user, data collection such as the dictionary and arrays were used for classified listing of data. But

the big question was how classified listing of data is represented in BizinfoApp? This prompted the need to create an object for listing the data model. So the developer created a class to represent the listing of menu items. The listing of data model at high level abstraction of data has properties such as menu title and menu Icon. Since there is a listing class with these properties, the need for multiple objects to represent each of the listing was important. The outcome is the menu items that the user sees when the app is launched. Each menu item has a particular screen to go when clicked. For instance, Camera goes to the camera screen, Map goes to the map screen, Email goes to the email screen, About goes to the about screen.

C. Navigation

Navigation helps the user to go from one screen to the other. After implementing the menu items, the developer noticed that menu items were static and the user was not able to navigate to different screen as planned. Therefore, prepareForSeque method was implemented to enable the user navigate each screen with ease (see figures 18-22).

```
//
//  MenuItem.h
//  BusinessInformationApp
//
//  Created by Charles Nebo on 20/01/15.
//  Copyright (c) 2015 charles Nebo. All rights reserved.
//

#import <Foundation/Foundation.h>

@interface MenuItem : NSObject

@property(strong, nonatomic) NSString *menuTitle;
@property(strong, nonatomic) NSString *menuIcon;
@property(nonatomic) int screenType;

@end
```

FIGURE 18. Source code for Menu Items Properties

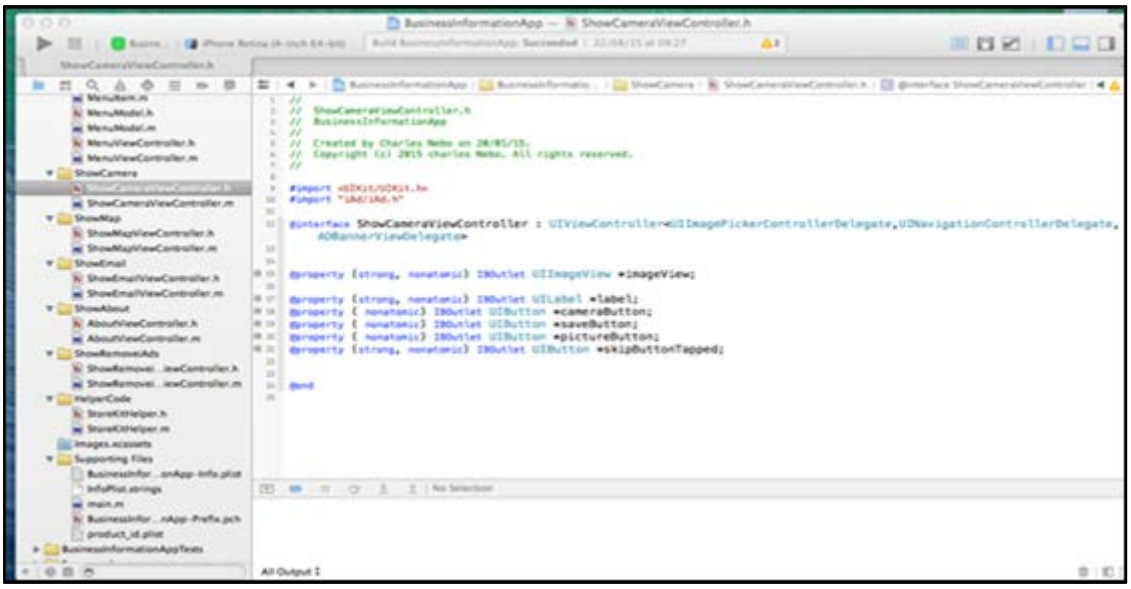

FIGURE 19. Source code for Menu Items Implementation

```
//  MenuModel.m
//  BusinessInformationApp
//
//  Created by Charles Nebo on 20/01/15.
//  Copyright (c) 2015 charles Nebo. All rights reserved.
//

#import "MenuModel.h"
#import "MenuItem.h"

@implementation MenuModel

-(NSArray *) getMenuItems
{

    NSMutableArray *menuItemArray = [[NSMutableArray alloc]init];

    MenuItem  *item1 = [[MenuItem alloc] init];
    item1.menuTitle = @"Use Camera";
    item1.menuIcon = @"CameraMenuIcon";
    item1.screenType = ScreenTypeCamera;
    [menuItemArray addObject:item1];

    MenuItem  *item2 = [[MenuItem alloc] init];
    item2.menuTitle = @"Use Map";
    item2.menuIcon = @"MapMenuIcon";
    item2.screenType = ScreenTypeMap;
    [menuItemArray addObject:item2];

    MenuItem  *item3 = [[MenuItem alloc] init];
    item3.menuTitle = @"Use Email";
    item3.menuIcon = @"EmailMenuIcon";
    item3.screenType = ScreenTypeEmail;
    [menuItemArray addObject:item3];

    MenuItem  *item4 = [[MenuItem alloc] init];
    item4.menuTitle = @"About";
    item4.menuIcon = @"AboutMenuIcon";
```

FIGURE 20. Source code for Camera Model Properties

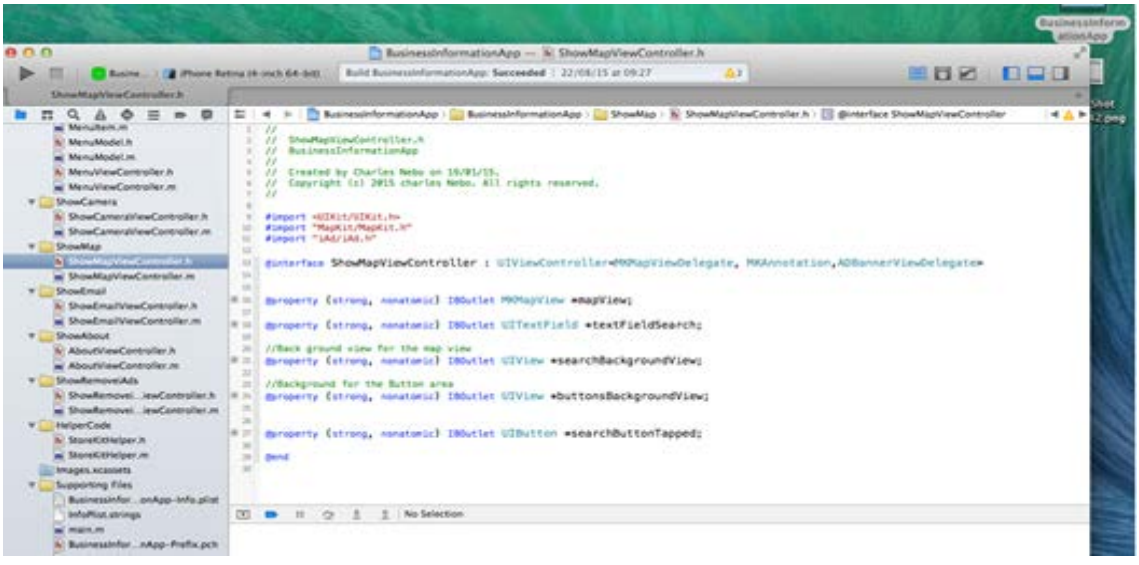

FIGURE 21. Source code for Email Properties

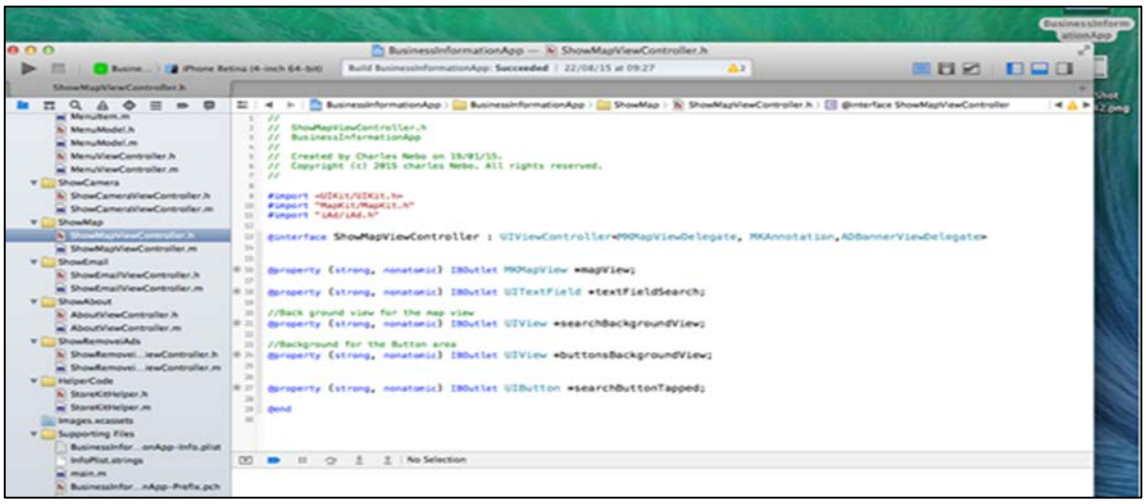

FIGURE 22. Source code for Map model properties

The View Layer

A view is the screen which the user sees when application is launched. There can be many views in iOS applications. The views in BizinfoApp application are the Camera screen, Map screen, Email screen and About screen. Each view responds to user interaction when clicked. But how is the view managed when there are many views so that user interaction is intuitive? A Navigation controller called the SWReveal View Controller was created to manage other View Controller sitting on the stack, so that when the user hits the menu button, the menu item is added on the stack and sit on top of other views. Conversely, when the user clicks the 'BACK BUTTON' the top most view controller is popped off the underlying view controller to make it visible again. In this way, the user sees different views as choices are made. Figure 23 shows Multiple Views in the BizinfoApp Application.

FIGURE 23. Multiple views in BizinfoApp Application

A. Menu View (Table View Creating Process)

A table view is an object which presents data in a list of multiple rows. When a UITableView is dropped on the view of a storyboard it automatically becomes the property of the view controller. But how is the table view going to fill the table with data? The table view calls on some methods on its delegates and asks the delegates to get data for the rows. It also notifies the delegates that the user has selected a row. The UITableView class has two protocols namely the UITableView Delegate and the UITableView Data Source. The UITableView Delegate protocol is to notify the delegates of user event such as row selection. While the UITableView Data Source protocol is to query the delegate for data for display in the tableview. The owner of the two delegate methods is to implement the required methods in the .m file. The owner of the delegate methods is usually the view controller as in this case. The developer used the table view to display the list of menu items.

B. Menu View (Pan Gesture Recognizer creating Process)

A gesture recognizer is an object that is attached to the view so that views respond to certain user actions. There can be multiple gestures namely Tap gesture, Swipe gesture, Pinch gesture, rotate gesture and Pan gesture. After implementing table view so that menu items can be

displayed, the developer discovered that the view was static and therefore unintuitive for the user. A pan gesture recogniser was needed so that the user can pan the view from the left to the right to reveal menu items. Then the user selects one item at a time from the displayed menu list (see figure 24 on page 47).

C. Camera View (UIImage View creating process)

The UIImage View is an object which displays image data. The image objects can be used to represent still pictures taken with camera. The UIImage Picker Controller manages the main view to enable user interaction with Photos. But how does the view Controller know there is photo to get and save in the Camera roll? There was the need to implement the delegate methods in order to enhance user interaction of get picture and save commands in the Camera. So the UINavigation Controller Delegate which is responsible for user interaction was implemented to take care of it. When the Camera button is clicked, a snap shot is taken. Then the user gets the picture with the Get Picture button and subsequently save to camera roll with the save button (see figure 25 on page 47).

D. Map View (MKMap View Creating process)

MKMapView is an object that displays the map interface and map information to the user. The region is the current display on the map view and defines the centre location called latitude and longitude. The MKMapView defines some properties that allow the user to manipulate data directly and still make it available to the map view. These properties include the Span which defines how much of the map is visible using the zoom property; Map types such as Standard, Satellite and Hybrid can be toggled by the user as needed. Consequently, map view can zoom in or zoom out as needed by the user. Annotation is a custom property used as place mark in the map interface. But how is the map view going to reflect the changes in its map interface? MKMapView class supports user interaction that make changes to the map interface by changing the position and zoom level of the map. MKMapView class has some delegates that it calls for loading of the map data and changes in the portion of the map. MKMapView Delegate protocol when defined is responsible for receiving notifications such as changes in the location of the user or region or failure of the device to display current location. The view controller conforms to these methods by implementing them so that changes in the map view or data is activated as the user continues to interact with the map.

The BizinfoApp MapView interface consists of Map view, text Field, and search button. The user can use these tools to search for cities and café hotspots. There are also properties such as the zoom and map Type properties which can be toggled in order to manipulate its contents (see figure 26 on page 45).

E. Email View (Email Client Creating Process)

The email view has only a UIButton. The user clicks the button to open the email client to send email (see figure 27 on page 45).

F. About View

About view show the name of the App, version number, Feedback link, Rate this link and information about the developer (see figure 28 on page 45).

Controller (Application logic Layer)

The controller is the logic layer and glue that binds the model to the view. It is responsible for updating and receiving information from the model and handles user interaction from the view. From the application logic point of view, consideration was made based on various behaviours, actions and events related to model of data and view elements. BizinfoApp has multiple views and logic actions which were provided where necessary to make the app useable. In addition, the user interface has many buttons that trigger actions that enable the app to work well. In the modelling of data, delegate methods were used extensively to enhance user interaction to the application.

5 CASE STUDY: IPHONE APPLICATION DEVELOPMENT IN TAALVISIO OY

5.1 Introduction

Practical training is a compulsory course which students at the Oulu University of Apply Science must complete before graduating from school. It involves applying for a practical training job in a Company of choice relevant to the students' field of study. Sequel to this, I applied and got a practical training place as Software developer at Taalvisio Oy in 2013. Taalvisio Oy is software developing firm which specializes in building iOS applications. It is situated in Oulu region of Finland. My background as student of Software Engineering was a motivating factor for the internship. My development tasks were to create iPhone application for release in the App Store and a self-reliant literature research and implementation of iOS application development and technologies. In this case study, I will discuss my experiences in the context of how iPhone applications are developed in Taalvisio Oy during my internship.

5.2 Company History

Taalvisio Oy is a Software Development Company based in Oulu. It started business operation in 2007; officially registered as a Company that deals in business activity. Today, its business operation encompasses iOS application development and technologies. It is a small sized Company with few employees' such as the managing director, who is the owner of the company, programmers or developers, designer, and analyst working at different sections of the office. There are two categories of Apps which are built in Taalvisio Oy: Clients Apps which are built for Clients to sign off and Apps to be released in the App Store. These apps undergo the same software development process.

It is the company policy to use a software development process which is popular either in the software industry or literature for developing iOS applications. Agile development model was the preferred choice for the company during my short stay as intern in the company. Applying this development process enable Taalvisio Oy to develop robust iPhone applications while leaving behind the complexities that are associated with developing mobile applications in general.

Normally, building application starts when there is a contract between the company and a Client and a compromise to deliver the App at an appointed time is reached; or the company had brainstormed a new idea of an application after intensive market research. The life cycle for developing iOS applications in Taalvisio Oy is as follows: Organisation context, Data Collection, Development Context, Development Process, Prototyping, Testing, Deployment and Maintenance.

5.3 Data Collection

Taalvisio Oy builds apps for Clients' as well as apps to be released in the App Store. Both follow the same method of gathering data. Clients' apps are customized according to the need of the customers while apps to be released in the App Store are for free users. They would carry out some research to make sure they are not building a replica of apps in the App Store. It also helps them to get update about current technology in use and how other software companies would have built the app they are interested in. The company would use observation and interviewing methods for enhancing data collection. Additional data were collected during their project meetings when the developing team and users interact. The objective of data collection is to collect data that is reflective of the application they build.

5.4 Development Context

The developing team members are two programmers, designer, analysts, users and the managing director who is also a software engineer by profession. Meetings are arranged in the Business Kitchen, outside the Office environment of the company, where the developing teams, users, and programmers are in attendance. Business Kitchen is a meeting place where IT professionals and Business incubators for product development meet. It is specially equipped to take care of IT and business related issues. In the business Kitchen, developers and users interact and develop a working prototype of design functionality of the application. Developers would make some changes to the prototype dynamically at the Business Kitchen during conference meeting with users. They would save their documentation in GitHub, a source control that provide access to the development team anytime anywhere. The day usually ends with some recital sessions such as;

1. The review of the day's progress with regard to objective set.
2. The review of what had been achieved and what to do next day in a TODO list format.

3. Documenting the ways in which requirements were met in a log maintained by project Manager.
4. Ad hoc groups would be formed to address design and coding issues.

5.5 Development Process

This is where all the programming happens. Programmers would code different modules of the same prototypes. They would divide the coding into two stages: Coding for functional requirements and User interface UI respectively. The core functional requirements are coded first and get integrated with other modules development. In the second stage, user interface is designed into composites in accordance with iOS User Interface Guide Lines. Designers would make sure the look and feel of all designs are the same with iOS apps to avoid app rejection in the App Store and at the same time comply with Apple rules.

5.6 Prototyping

The development team would do some functionality testing of prototypes to make sure it is in sync with the business requirement objective. The prototypes are tested and sent to the client for feedback. They would use test flight to send the application to Clients. When the feedback was received, necessary changes are implemented. It becomes a vicious cycle as development, prototyping and testing were repeated until the required prototype was reached.

5.7 Testing

The developing team would fix the bugs discovered from the Prototype testing. When all bugs are fixed, more testing with all versions of the iOS are tested using simulator and iPhone device respectively.

5.8 Deployment

After testing was completed and the final feedback was received from the client, the app is ready for deployment. The application is approved and signed-off by the Client if the app is a client app

otherwise the application is uploaded to the App Store for user consumption. For apps that are uploaded in App Store, the following need to be checked for easy access to the App Store:

- The company registers as developer in the Apple developer website and pay the annual fees.
- The company checks the rules and regulations guiding deploying of applications in the App Store.
- The company must be sure the App is bug free because Apple has no patience with buggy applications.

5.9 Maintenance

The maintenance is the final stage of the app development; however, as more and more feedbacks are received from the users' changes are made as bug fixes or improvements. The company provides appropriate security patches, performance improvement upgrades and new user interface at a regular interval in the life-cycle of the app. Thereafter, the app is prepared for advertisement.

6 RESULT

This chapter shows the screen shots of BizinfoApp functional screens. Figure 23 shows BizinfoApp, Figure 24 shows menu screen, Figure 25 shows Camera screen, Figure 26 shows Map screen, Figure 27 shows email screen and the figure 28 shows About Screen.

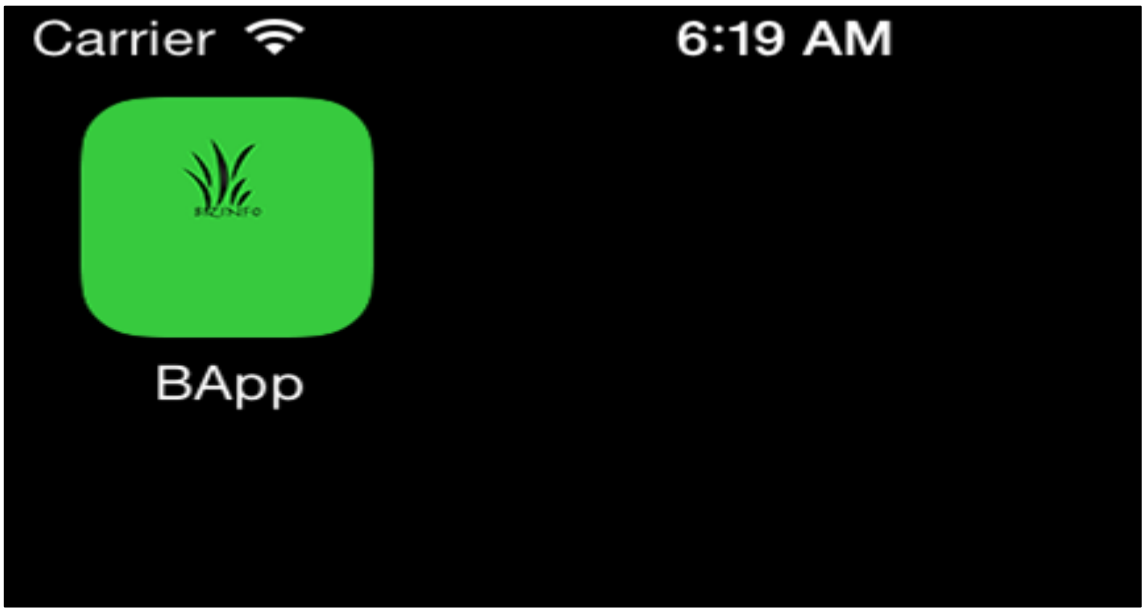

FIGURE 23. BizinfoApp

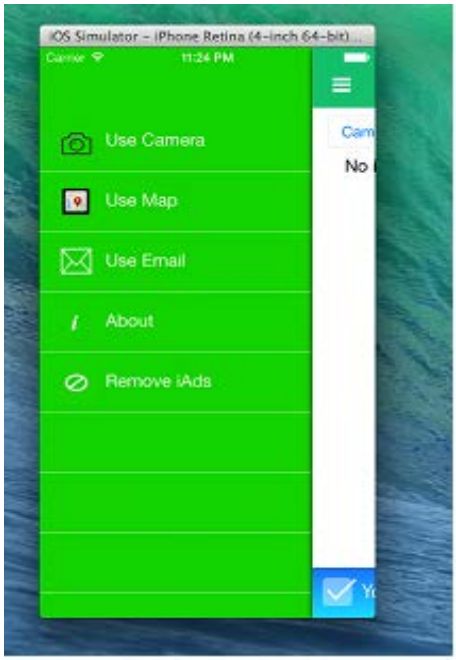

FIGURE 24. BizinfoApp Menu Screen

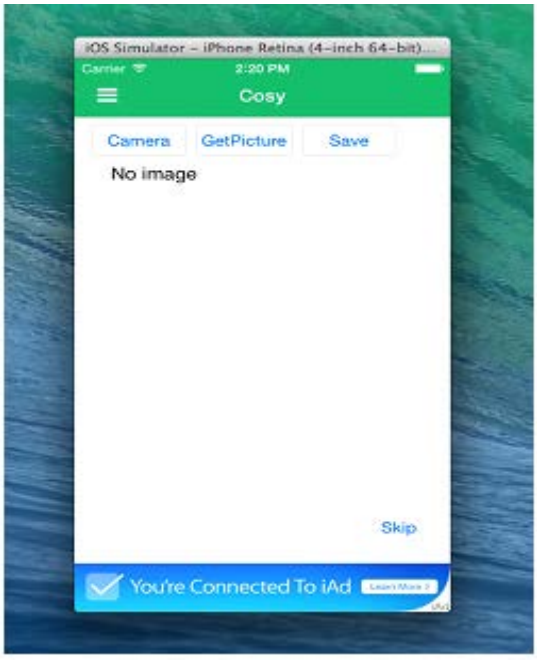

FIGURE 25. BizinfoApp Camera Screen

FIGURE 26. BizinfoApp Map Screen

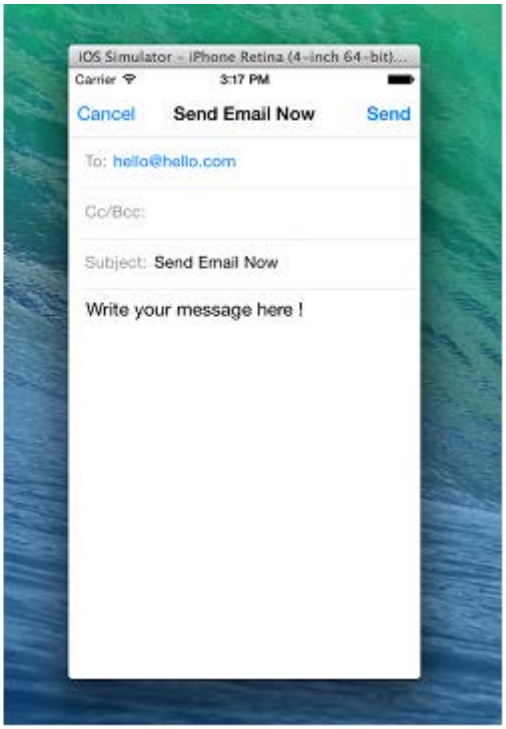

FIGURE 27. BizinfoApp Email Screen

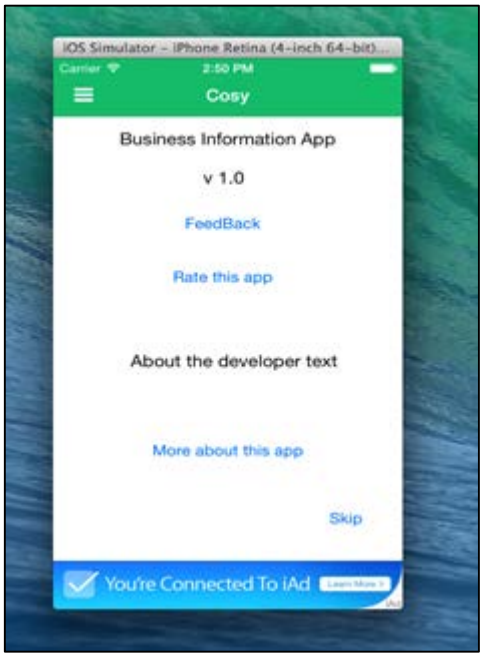

FIGURE 28. BizinfoApp About Screen

7 CONCLUSION

This research project had explored current development study for developing iPhone application where the development environment and technologies were problematic. Some software development process was reviewed and case study with a view to finding appropriate software development Model for building iPhone application. Model-View-Controller Design Pattern was the development Model for building BizinfoApp application for release in the App Store. As mobile applications are gradually becoming part of human life, there exist some development challenges for building robust iPhone applications that will meet an ever increasing user demand.

Regrettably, the practice of developing iPhone application without software development process does little to improve application development in general. Not only does a development process transfer to the application domain, it targets and resolves issues emanating from the iPhone device and evolving technologies. This research project has shown that developing iPhone application using Model-View-Controller (MVC) design pattern improve the quality and robustness of applications.

Findings show that software development process is important for the success of application development. It provides a means to determine quality applications. While the result of this research is not conclusive, this study provides insights into developing iPhone application to be released in the App Store using software development process. It is hoped that this research will increase the awareness and understanding of developing iPhone application.

8 DISCUSSIONS

This thesis has widened my knowledge in the field of software development in a way that bits my imagination. Firstly, it enabled me to engage in a self-reliant extensive literature research and implementation of iPhone application development - a knowledge gained during my practical training experience from a software Development Company in Oulu.

Secondly, the knowledge acquired was used to learn and develop iPhone application (Bizinfo App) for release in App Store using a software development process. During the research process numerous software process methodologies had been reviewed in order to find appropriate development model best suited for developing iPhone application.

Normally a software process model is characterized by stage by stage development pattern such as, planning, requirement, analysis, design, implementation, testing and documentation. Following this pattern to develop iPhone application for release in the App Store culminated in the knowledge and skill gained during the course of my studies.

In the research project, comparison of different software methodologies was carries out based on certain factors namely, iPhone device form factor, constraints in software development environment and the technologies that apply to application development life cycle stages.

Model-View-Controller design pattern was selected based on the aforementioned problems at hand and ability to contain them. MVC is an abstract design pattern that helps to define the structure of data model and its interaction with the rest of the application. In fact, applications designed using MVC design pattern tend to be more re-usable and extensible than other applications and their user interface is more defined. While this is not the only option of for developing iPhone application, the author could have used only the software development kit (SDK) provided by Apple for the development mission and it would work. But the goal was to achieve iOS platform best practice, maintain quality applications and good user experience.

The major work for this thesis was done in the Bizinfo App project design and implementation, where the author needed more time to learn the programming language for developing iPhone application in a private setting because the knowledge acquired from the intern was insufficient.

Furthermore, Wireframing to make the functional specification into the actual iPhone screens, required some thoughtful ideas such as how each screen will achieve the business objective? And, how user elements are arranged in the screen in a way that is intuitive for the user and yet flexible enough to be functional as well? It was easy to see that these ideas needed more skills and tools to make it happen than earlier thought.
Again, learning to use the design composites as a framework to put together designs for the BizinfoApp with different tools took more time than anticipated and thereby caused some delays that affected the time schedule.

Future research possibilities should focus in the area of user experience comprising of design and performance of user interface. The smaller display of the iPhone device and different styles of user interaction have major impact on interaction design, which in turn influences app development. For instance, the iPhone device user interface is based on buttons, touch, and virtual keyboard rather than the familiar desktop application.

REFERENCES

Agile Alliance. 2001. Agile Software Development Manifesto. Date of retrieval 23.4.2015, http://agilemanifesto.org/

Alahuhta, Petteri. 2011. Technologies in Mobile Terminals Enabling Ubiquitous Services. Espoo: VTT Publications 761.

Bilton, N. 2011. Mobile app revenue to reach $38billion by 2015, report predicts. Date of retrieval 14.6.2015, www.bits.blogs.nytimes.com/2011/02/28/

Boehm, B. 1998. A Spiral Model of Software Development and Enhancement. Computer, Volume. 21, 5 (5). May 1998, pp. 61-72.

Boehm, B. & Turner, R. 2003. Balancing Agility and discipline: A Guide for the Perplexed. Addison-Wesley, Boston.

Camponovo G. & Pigneur Y. 2003. Business Model Analysis Applied To Mobile Business. Date of retrieval 18.1.2015, http://www.hec.unil.ch/gcampono/Publications/GC2003ICEIS.pdf

Cox, C. 2008. Essential of UTMS. New York: Cambridge University Press. Date of retrieval 17.8.2015, http://www.cambridge.org/fi/academic/subjects/engineering/wireless communications/essentials-umts?format=HB

Davis, F.D. 1989. Perceived Usefulness, Perceived Ease of Use and User Acceptance Information Technology, MIS Quarterly Vol.13, No. 3 (1989), pp. 319-340

Gruber, H. 2007. The Economics of Mobile Telecommunications. Date of retrieval 20.1.2015, http://www.cambridge.org/9780521843270

iOS Developer Library, 2015. Cocoa Core Competencies. Model-View-Controller. Date of retrieval.14.6.2015. https://developer.apple.com/library/prerelease/ios/documentation/General/Conceptual/DevPedi-CocoaCore/MVC.html

iOS Developer Library. 2015. iOS Technology Overview. Date of retrieval 13.5.2015, https://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTeOverview/iPhoneOSTechnologies/iPhoneOSTechnologies.html#//apple_ref/doc/uid/TP40007898-CH3-SW1

Roma P. & Perrone G. & Valenti F. 2013. An empirical Analysis of Revenue Drivers in the Mobile Business. Date of retrieval 19 5.2015, http://www.pomsmeetings.org/confpapers/043/043-0354.pdf

Salo, Outi. 2006. Enabling Software Process Improvement in Agile Methodology for Mobile Software Development Teams Organisations. Helsinki: VTT.

Statistic portal 2014. Most popular App Store downloads. Date of retrieval 13.2.2015, http://www.statista.com/statistics/279258/distribution-of-worldwide-iphone-app-downloads-by-category

Saif, 2012. Agile Methodology an Analysis to its title. Date of retrieval 18.2.2015, https://ahmsaiful.wordpress.com/2012/10/09/agile-methodology-an-analysis-to-its-title/

Valacich, J.S. George, J.F, Hoffer, J.A, 2004. Essential of Systems Analysis and Design. Upper Saddle River, New Jersey: Pearson.

von Hippel, Eric 1986. Lead Users: A Source of Novel Product Concepts. Management Science, Vol. 32, No. 7, pp. 791-805.

von Hippel, Eric. 2005. Democratizing Innovation. MIT Press, Cambridge, MA, USA.

Wikipedia 2015. Apple's Mobile operating System. Date of retrieval 13.5.2015, https://en.wikipedia.org/wiki/IOS

Zhang, Zheying. 2007. Effective Requirement Development - A Comparison of Requirements Elicitation Techniques. Date of retrieval 17.8.2015, http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.135.185&rank=1