Spring 2012

Software Engineering for the Mobile Application Market

Jacob Schwartz
University of New Hampshire - Main Campus

Software Engineering for the Mobile Application Market

BY

Jacob Schwartz

THESIS

Submitted to the University of New Hampshire
In Partial Fulfillment of
The Requirements for the Degree of

Bachelor of Science with University Honors

In

Computer Science

May 2012

Table of Contents

Abstract

Software Engineering for the Mobile Application Market

by

Jacob Schwartz
University of New Hampshire, May, 2012

One of the goals of the current United States government is to lower healthcare costs. One of the solutions is to alter the behavior of the population to be more physically active and to eat healthier. This project will focus on the latter solution by writing applications for the Android and iOS mobile platforms that allow a user to log and view the food they eat, set goals for the food they want to eat, tag the food with nutrition attributes and see how closely they meet the food group foals set forth by the Center for Distance Control and Prevention (CDC) as well as the goals they have set for themselves. The application will be a single-user stand-alone application that does all processing locally on the mobile phone. The process in which the problem statement was formulated will be discussed, along with how that problem was turned into requirement for the app. From those requirements, user interfaces and code were developed, and tested. In the end, two applications were made for different platforms using Software Engineering techniques, which allowed the development process to be as efficient as possible.

I. Introduction

More than one-third of American adults are considered legally obese by the CDC [1]. Oftentimes, obesity is a result of overeating, but it is often difficult to be sure of how many servings of a food group a food contains and even how much a person has already eaten that day. While a person may want to log the foods via pencil and paper, they may not want to carry around these materials with them at all times. The most convenient way to record this data would be to use something that the user already keeps with them. More than 104 million Americans were said to have been owners of smartphones during the last three months leading up to February 2012 [2]. Smartphones have risen in popularity in recent years; mobile apps saw the number of downloads go from 450 million in 2007 to 1 billion in 2008, a 146% growth. A study done by the World Mobile Applications Market, a research firm says that the mobile application marketplace will reach $25 billion by 2015 after the market was valued to be only $6.8 billion in 2010 [3].

Software engineering is the “application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches” [4]. The term ‘Software Engineering’ first appeared at the 1968 NATO Software Engineering Conference. While software development and Software Engineering are used interchangeably, the former is a more general term for programming an application and the latter involves more analysis of the process to find where the developer(s) can improve. By using the same Software Engineering principles to create the same mobile application on different platforms, the applications can be created at a fast rate and more efficiently.

This project was done in collaboration with Elbrys Networks. This company is based in Portsmouth, New Hampshire. They focus on wireless sensors and devices that can collect health based data for a user, such as blood pressure monitors and glucometers. Their first foray into the health data market was a smartphone app called GrokLife. GrokLife was an application for iPhone, Android and also had a user portal that could be accessed through an Internet browser. In addition to including social media features such as events and calendaring, the real focus of the application involved using the internal sensors of the smartphones, namely the accelerometer and the global positioning system (GPS). When the user indicated that they were doing physical activity, these sensors would be started and data would be collected. Raw accelerometer values were then turned into the number of steps taken by the user and the number of calories burned, while the GPS measurements were calculated into the average speed of the user and distance that they travelled during the activity.

After the first versions of the application were released in June of 2011, Elbrys developed a plan for the future of their new mobile app. They wanted to make it a total health application, allowing the user to take readings from wireless sensors external to the phones, such as weight scales or blood pressure monitors that could talk to devices through a wireless protocol. In addition to keeping track of these vitals, they also wanted the user to be able to record the foods user the eats, in order to see a pattern in their eating and correct it, if the pattern showed a negative trend. The application, for simplicity, will be targeted at a single demographic: 40 year old women, 5 foot 6 inches, 120 pounds that partakes in less than 30 minutes of physical activity a day and are simply looking to eat healthier. The system is intended to raise awareness and change behavior. Although the system is based on sound principles provided by the United States government it is for ‘recreational’ purposes and not intended to be scientifically accurate. My senior project consists of making a standalone dietary journal, according to the specifications given to me by Elbrys Networks, which would later be integrated into their GrokLife application.

II. Methodology

Now that there is have a problem to solve, it must be determined what pieces of the problem are most important to solve and how the application will help solve this problem. A *user story* is one more sentences in the language of the end user of the product that explains what the user can expect from the piece of software that the story describes. User stories are a key piece of the Agile software development methodology for defining the functionality of the software. User stories are usually written by a ‘customer representative’, someone who assumes the role as the consumer that would be using the product. The stories are written in the form of: “As a <role>, I want <goal> so that <benefit>” This explains not only the piece of functionality that is to be implemented, but also directs it at a specific user base and explains why the functionality is being implemented and included. One example of a user story is “As a user, I want to be able to write a message to another user that I am friends with” but it also includes functionality like “As an engineer, I want the server to encrypt user credentials so that if the server is attacked, the user’s private data is safe”.

The development of the application did not follow every single detail of the Agile system, as the methodology is directed more toward teams in an attempt to quantify their productivity. The notion of user stories, however, was used to create the requirements for the application. Seventeen user stories were ultimately implemented for the application, after several had to be removed due to implementation complications and hardware restrictions. The chosen user stories and other requirements for the application can be seen in Figure 1.

Figure 1: Selected User Stories

New-User Stories

1. As a new user, I want the system to ask me to define personal eating goals by selecting the food tags for food attributes (see Food Tags below) that I wish on the food that I eat, so that the system can track my personal eating goals.

‘Eating Journal’ and ‘Food Cupboard’

2. As a user, I want the system to maintain a ‘*food cupboard*’ of the foods I have purchased or previously eaten that will be used to define the foods I most commonly eat (unique id, name, associated image, tags, date of last update, nutrition facts if available), so that I can more quickly indicate I am eating them again as well as see what I commonly eat and when.
3. As a user, I want the system to maintain an ‘*eating journal*’ of the foods I have eaten (id of food in food cupboard, date/time eaten, snack/meal indication), so that the system can track all my meals/snacks and help me eat more healthy.

Food Tags

4. As a user, I want the system to support food tags for ‘food groups’ (grains, whole grains, vegetables, fruits, dairy, protein, junk), so that I can tag food with the food groups it provides and see how closely I am eating to the CDC guidelines.
5. As a user, I want the system to support food tags for ‘food attributes’ (low sodium, sugar-free, all-natural, low cholesterol, minimal processing, free-range, vegan, vegetarian, organic, gluten-free), so that I can tag food with these attributes and see how closely I am eating to my personal goals.

Daily Goal

6. As a user, I want the system to track my daily eating goal, so that I have a system that makes me more aware of my eating habits.
7. As a user, I want my eating goal to be based on CDC standards for a 40 year old women who is 5'6" 120 lbs and does less than 30 minutes activity a day (number of foods with these tags: 3 foods tagged grains or whole grains, 3 foods tagged grains, 3 foods tagged vegetable, 2 foods tagged fruit, 3 foods tagged dairy, 3 foods tagged protein), so that the application is targeted at this market.
8. As a user, I expect my daily goal to be set to 0% at 12:00 midnight on a given day, so that each day I start over.
9. As a user, I expect the application to track my food entries for food groups and personal goals and update my % of goal every time I enter a food as 'eaten', so that I can track my goal throughout the day.
10. As a user, I expect there to be a clear algorithm used to calculate my % of goal ((for each meal/snack add one to each food group (from the meal/snacks tags) up to max for each meal - % goal = Sum (all food groups)/Sum (total required per food group))), so that I have confidence in the consistency of the system.
11. As a user, I want to be able to change my personal goal (food attribute tags), so that I can have the application change as I change.

Visualizing Progress

12. As a user, I want to be able to view my eating journal over a 7 day period to see how well I am doing at achieving my goals, so that I may determine whether I am improving or not.

13. As a user, I want to be able to see my % of goal and to be able to find out what I need to eat to reach 100%, so that I can achieve my goal.
14. As a user, when viewing my eating journal I would like items that contributed 0% because they did not meet my personal goal be color coded differently, so I can, at a glance, see how many times I am eating and doing nothing to help me achieve my goal.
15. As a user, when viewing my eating journal I would like items that are tagged 'junk' to be color coded differently, so I can, at a glance, see how many times I am eating junk food.

Extra-Credit (Social Network)

16. As a user, when I enter a food into my 'eating journal' I want to be offered the chance to share what I am eating to my Facebook wall, so that my friends/family may comment and help me eat more healthy.

From the finalized user stories, the marketing team and the developers then worked together to define the behavior of the application. A program called Balsamiq Mockups was used to create the following wireframe screens for the application:

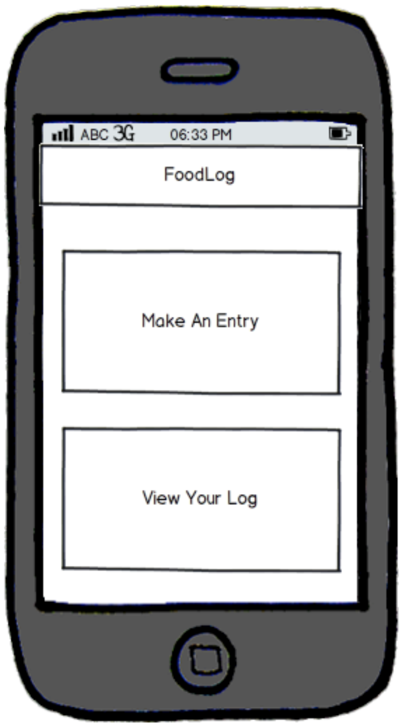

Figure 2: This is the home screen of the application, with buttons to guide a user to the two main features of the app: adding an entry to the log and viewing the log itself.

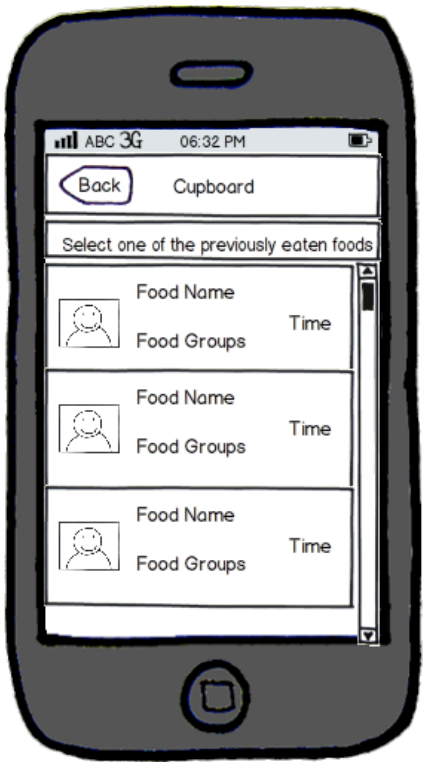

Figure 3. This is the “Food Cupboard” page. It allows the user to re-enter a food they have entered previously. This way, the user can build a list of commonly eaten foods to select from, instead of entering a food multiple times.

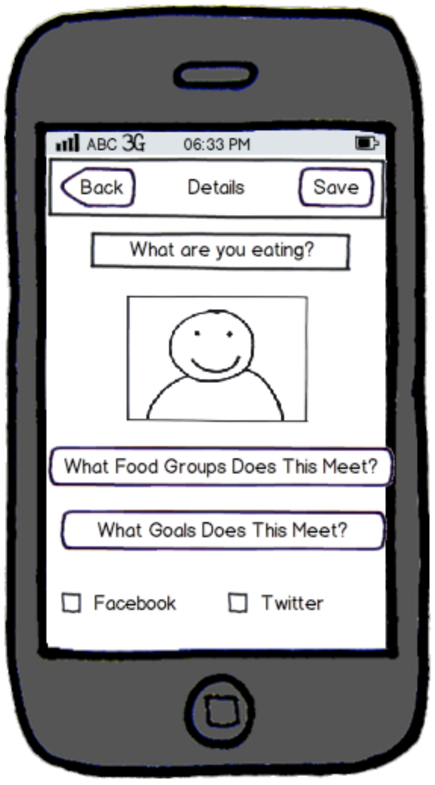

Figure 4: This is the food entry screen. All the details of the user's food must be entered before saving the entry. An entry consists of a name, picture, food groups that food meets and any user-set goals the food meets. The user can also choose to share their entry with their friends on Facebook and their followers on Twitter.

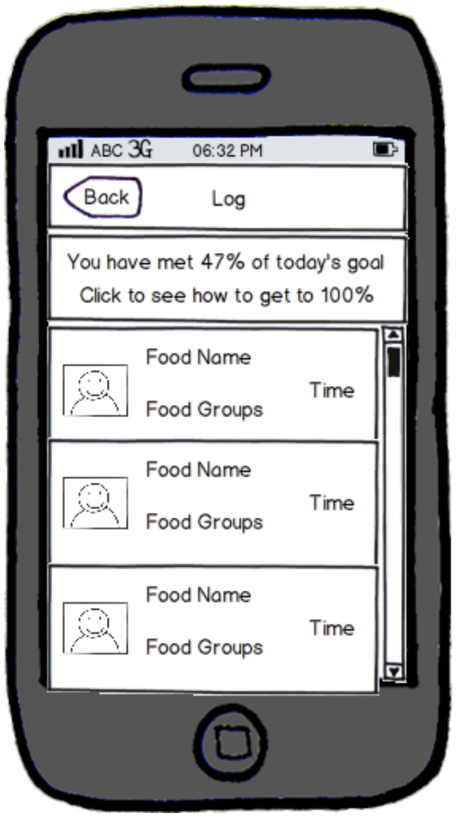

Figure 5: This is the log screen, which shows the user's entries over the past seven days and the progress of the current daily goal. Clicking on the current progress button will show the user what they can do to improve to meet their goal.

For a more complex application, one *screen flow* would describe one piece of functionality and consist of a description of that functionality, followed by all of the screens used to execute that action in the order that the user has to follow to complete the action. In an application with four screens, the flow of how the user does something is obvious a majority of the time, and most of the time requires only one screen to display the action being executed. With the wireframes now complete, actual screens files can be made for the two platforms. Once the screen is set up, implementation can begin to make the screens function.

III. Implementation

The application runs on two different mobile operating systems: Android and iOS. Google and the Open Handset Alliance are currently developing the Android operating system, originally created by Android Inc, which is a consortium of, currently, eighty-four manufacturers that create open standards for mobile devices. Android 1.0 beta was released in November 2007, and Google has put out nine updates since then, each one with a codename based on a dessert. Most recently, Android version 4.0 Ice Cream Sandwich was released in an effort to combine the phone-based Android 2.3.X Gingerbread with the new features of the tablet-based Android 3.X Honeycomb. The project runs on Android 2.2 Froyo and Android 2.3 Gingerbread, which make up a combined 85.3% of the Android devices that are currently active. [5]

The Android operating system actually runs on top of the Linux kernel, with some extensions built by Google in order to save power. Dalvik is run as a virtual machine that the mobile apps will run on. Dalvik code is made from Java bytecode, making the major programming language of the platform Java. Native code can also be written in C/C++ and translated into Dalvik over the Java Native Interface. Google then wrote the frameworks for aspect of the operating system that would not be native to the Linux kernel, such as window and view managers and telephony resources. In addition, standard libraries that developers would be used to were added, including Apache HTTP, SQLite and OpenGL ES [6]. Lastly, Android shipped with several stock applications, include a browser, email client and calendar.

Apple's iOS was released in 2007 on the original iPhone and iPod Touch devices. Originally called iPhone OS, iOS has also moved onto other Apple devices, namely the iPad and AppleTV. Major releases of the operating system have been released yearly in June since the initial iOS 1.0 release. iOS 5 is the most recent release, which included new apps like Messages and Reminders, and included a new Notification system, similar to Android's, and Twitter integrated right into the OS. (Apple) Because Apple controls the devices that are allowed to run iOS, every device that Apple sells at the time is able to run the current version of the iOS. This allows developers to focus their attention on developing their app for the current iteration of the operating system. For this reason, the project was developed for iOS 5.

iOS is derived from Apple's desktop operating system OS X, making both of the operating systems a derivative of Darwin, the open source operating system released by Apple in 2000. There are four layers of abstractions in iOS. At the top lies Cocoa Touch, which, much like its OS X counterpart Cocoa, is the using interface framework for building apps. Below that is the media layer, which handles various hardware pieces that interact with the user. Underneath the media layer lays the Core Services, which provides an application programming interface (API) that connects to low level C and aspects of the last level, the Core OS layer.

The two platforms also have differences at the implementation layer. As mentioned, Android is implemented in mostly Java, whereas in iOS developers use Objective-C, C and C++. The way that the interfaces are designed is also different; extensible markup language (XML) is used to create the screens for Android, while iOS developers use an application called Interface Builder to drag and drop widgets on a canvas in order to structure the user interface. Apple also has strict guidelines on what the user interfaces may look like. Widgets must be used for their designated purpose.

On Android, before 4.0, the developer had free reign over the appearance of their app’s user interface. The devices that Apple makes that run iOS only have a single hardware button for all off-screen navigation. The Android framework provides the ability for all sorts of hardware buttons, back, home, search and menu being the most popular, but also can allow for physical keyboards, trackballs and a camera button.

IV. Testing

Although functionality is tested as it is implemented, it is important to test the system as a whole when the application is complete. Much like writing a paper, when the paper is complete, it is often a good idea to have others read it over, not only to find mistakes, but to also make sure that it makes sense as a whole. The same applies to software: it is important to make sure the application is intuitive. Initially it is best to let those with no prior knowledge use it. This way, the testers can validate assumptions made by the developers about the way the users will interact with the application. The testers are also more likely to find new errors because they will execute different actions than the developers who know the system.

When testing mobile applications, it is important to be able to test on a wide variety of devices and versions, because they change faster than desktop applications. As mentioned above, there are over a hundred different kinds of phones that run the Android operating systems that are supported by the application. These phones are all a little different: they use different cameras, have different screen sizes, and have a varying number of buttons. It is important to make sure that an application works with as many of these as possible as when sent to the Android Play Store, because any device that meets the requirements will be able to download the application, regardless of if it was tested on that device or not. Testing it easier on the Apple side of things, as there is only one kind of device to test for, all with the same hardware. Apple users also tend to update their phones to the latest version of the operating system, where Android users rely on their device manufacturers create an update for their phone.

V. Conclusion

Together, iPhone and Android consisted of 80.3% of the mobile market as of February of 2012 [2]. Having a product that runs on seamlessly on both of those platforms in a similar fashion, while still utilizing the unique properties of each device is crucial to success in the mobile application market. In 2010, Instagram Inc. released it's a social networking photo sharing mobile app in the Apple App Store. Two years later, it finally released its Android app, after adding more features and making the system more robust. Facebook purchased the company, which won several awards for its iOS app, later that week for approximately $1 billion in cash and stock [7]. In the end, using Software Engineering methodologies are a proven way to organize the work a developer must do, and by applying it to mobile applications, it is simple to create the same application on different platforms easily and effectively.

Bibliography

[1] "Adult Obesity Facts." *Overweight and obesity*. Centers for Disease Control and Prevention, 2012. Web. 4 May 2012. <http://www.cdc.gov/obesity/data/adult.html>.

[2] "comScore Reports February 2012 U.S. Mobile Subscriber Market Share." *comscore*. comScore, Inc, 2012. Web. 9 May 2012. <http://www.comscore.com/Press_Events/Press_Releases/2012/4/comScore_Reports_February_2012_U.S._Mobile_Subscriber_Market_Share>.

[3] Perez, S.. *Mobile App Market: $25 Billion by 2015*. N.p., 2011. Web. 9 May 2012. <http://www.readwriteweb.com/mobile/2011/01/mobile-app-market-25-billion-by-2015.php>.

[4] SWEBOK executive editors, Alain Abran, James W. Moore ; editors, Pierre Bourque, Robert Dupuis. (2004). Pierre Bourque and Robert Dupuis. ed. *Guide to the Software Engineering Body of Knowledge - 2004 Version*. IEEE Computer Society. pp. 1–1. ISBN 0-7695-2330-7.

[5] "Platform Versions | Android Developers." *Google*. Google, 2012. Web. 5 May 2012. <http://developer.android.com/resources/dashboard/platform-versions.html>

[6] Bray, T.. "What Android Is." *Ongoing*. N.p., 2010. Web. 5 May 2012. <http://www.tbray.org/ongoing/When/201x/2010/11/14/What-Android-Is>.

[7] Segall, Laurie. *Facebook acquires instagram for $1 billion*. *CNN Money* . N.p., 2012. Web. 9 May 2012. <http://money.cnn.com/2012/04/09/technology/facebook_acquires_instagram/>.