**Software Qualities**

**Team G**

**Jose Rodriguez**

**Productivity**

Productivity is a very important part of the project and to the users. We want to make sure the application actually helps the patients and doctors by making their lives easier while creating effective design qualities in a relatively short amount of time. The team will be using the Titanium Studio, which is based on Eclipse and includes the ability to incorporate Android and iOS development tools. The studio makes it easier for us to have all the tools necessary to test and demo the application. The team will also work alongside with UConn’s Github to maintain and keep version control optimized. This will keep the code organized and updated across all team members who are constantly updating it. The mobile development tools for Android and iOS also provide us with virtual machines, or emulators, that allow us to test PHA as its developed. For those team members who have Android phones or tablets and iOS devices, it is also possible to stream the test code to test the code as you change it. For documentation, the team has also been using and wil continue to use google docs and hangout to simultaneously write on the same document while video chatting. This is an amazing time saver as some members are not able to meet in person very often.

PHA aims to make the management of your daily health activities easier by carrying it everywhere you go with your mobile device. PHA should be fast and easy to use, using and storing all of your information and making it accessible to enter new data at any time as long as you have a mobile network or wifi.

The languages used will be based on javascript and html, as well as xml, and then converted to Android and iOS code without using webviews. Using Titanium gives the incredible advantage of porting to various platforms without rewriting any code. All versions will also be updated at the same time, saving the team and therefore the project a lot of time in developing and maintaining the software. This is discussed below further on the Portability section.

**Portability**

Part of being a productive system is its portability. It is an important aspect of the project as mobile applications are becoming more and more prevalent. The ability to port from one mobile operating system to another will be crucial as more devices become available or new updates are made on existing ones. A huge part of the decision to work with the Titanium Platform was to be able to reach all these devices and make it more accessible for patients and healthcare providers using PHA no matter what device they own. This includes popular mobile operating systems like iOS, Android and even has the ability to function as a web application. These options can reduce costs, especially in other countries where the application has a more needed presence.

Naturally having a centralized platform can also bring negative aspects. Feature available in some devices might not be available in others, so while coding PHA the group needs to keep in mind the potential and scope of the devices we are trying to reach. For example, the way notifications are shown in android and iOS are different, therefore making sure that the application is consistent throughout all platforms needs to be done carefully. Nevertheless, using the same coding environment will help keep the core features be available in all systems and readily available for the future.

**Brittany DePoi**

**Evolvability**

In order to ensure the effectiveness and prolong the life of our software it is vital to make the application very adaptable and evolvable. First, this ability to evolve is necessary due to the nature of the platform—mobile. Mobile devices and their technologies are evolving very rapidly and we want to be sure that our application can keep up. To do this we are writing the code using Titamium. This programing language will allow us to quickly and easily maintain two working applications: one for the android and one for ios. In this way we keep our code up to date on the two major mobile platforms and keep it easily evolvable as changes are made and the application is improved and modified to support future mobile operating systems for deployment.

 The ability to evolve is also vital due to the medical nature of the application. On one hand, as knowledge of specific diseases progress, we want the information requested and recorded by the application to evolve to accommodate this new information. The diseases, in particular, we have chosen to represent are some of the most common, and therefore, have the most research that would make evolvabilty necessary. In addition, we also want to ensure that as the demand for monitoring new diseases increases, we are able to meet that demand with modifications to the current mobile application. Evolvabiltiy in this case will be achieved through a level of abstraction in the sets of the disease information required and how the data is stored and accessed. We will keep standardized interfaces from the mobile application to the data storage system to the doctor’s analysis so that any disease class can be added as long as it adheres to the standard. Also, all of the pertinent information to health records will be stored in one of the healthcare industry’s standard format to ensure that it will be usable and can be incorporated into future programs and applications of that standard. We also want to make the application evolvable with monitoring technology so that, for example, any testing devices have the ability to transmit their data wirelessly we could ease user convenience by incorporating this.

**Understandability**

Patient understandability is vital to the success of the application. The key purpose of the application is to bring the relatively complex personal healthcare monitoring system provided exclusively on the web through Microsoft HealthVault to the patient’s mobile device in an easy to use, helpful way.

 Through our application, convenience is added not only through the mobile platform, but also through our intuitive and simplistic user interfaces. When the user logs into the application, he or she will encounter a main menu screen with clear options. For aesthetics and ease of use, each of the options will be accompanied by an image that illustrates at a glance where the navigation will take them. In addition, the user will be able to choose only the specific diseases that apply to them. This way, the patient has easy access to disease entry information and is not encumbered by irrelevant navigation or screens for diseases. On the other hand, if the patient is monitoring multiple diseases, or just simply overall health and a disease, certain information will be shared. For example, if a patient is interested in monitoring diabetes, he or she will still enter their nutrition/diet information in the food section, but this will be shared with the diabetes specific information entered on a different screen. This allows the user to easily understand where to enter information, as redundancies will be eliminated. We also want to provide easy to understand feedback for the user. If the user understands the feedback then he or she will be more likely to effect change in their lives. This will be done through the use of simple graphs to show progress and encouraging phrases or symbols (such as stars or smiley faces). Finally, the application will have a help option to further explain the functionality of each of the options. User understandability will also be increased by hiding all of the back end functions that the user does not need to understand.

In terms of internal understandability, the software will be very easy to learn and understand. First, as previously mentioned, the source code will be the same for both the Android and IOS. So, not only is there only one set of code to change for evolvability, but there is also only one set of code to understand. The code will also be easy to understand due to the level of abstraction that will be applied in the model view controller design layout. This will allow for the object oriented components of the code to be intuitive and separate for increased ease of understandability.

**Che Chu**

**Reusability**

The Personal Health Assistant will be consisting of multiple reusable components. There are two main reasons for designing those components reusable. First, if we reduce the number of components and let multiple functions utilize same set of the resource , not only it will take less time to rebuild components, but it can also improve the performance by occupying less space in the memory of a smartphone. Unlike personal computers, smartphones usually have far less memory than PCs do. Also, users tend to execute multiple applications, which would cause crash if the memory space required by applications exceeds the limit. Second, this allows us to debug more efficiently. We only need to fix one component reused by many functions instead of fixing many components, written in separate files, that share similar or identical functions. The maintainability will also be improved by using this approach.

In PHA, both decision support algorithms and health information display screen need to retrieve the stored data frequently. To ensure consistency and reusability, we will create an object class to store any type of data instead of storing them in algorithm class or in screen class. Therefore, the health data will be separated from algorithms and screens. The graphical user interface is another reusable component in PHA because many screen share commonalities such as buttons, scrolling feature, chart display, data extraction function, and handling user I/O. The algorithms will be assembled into one object class. If an interface calls an algorithm, the interface created a general algorithm object and then calls the method corresponding to the name of such algorithm.

Since the PHA (both patient and provider versions) use the Microsoft HealthVault as the database to store health records, HealthVault is also a reusable component in regard to this application. Every health data will be retrieved and updated in the database side through JSON. However, each data communication will pass through an intermediate server, which reformats it in secured XML, and then send to MS HealthVault.

**Robustness**

Robustness shall be one of the most important key qualities as medical records expected to be correct all the time. Due to the reason that PHA is connected to HealthVault, which can let the patient upload sensitive health records and medical imagery in addition to personal diet and fitness information, PHA should maintain data integrity. PHA must be able to send and read the data correctly without distortion during transmission.

However, mobile phones are not dedicated to only one function, data integrity may not be easy to achieve. Android and iOS handle phone calls, memory management, wireless communication, and other apps concurrently with limited computing power. Each process or thread will try to acquire as many resources as possible. In addition, the PHA itself has several components running. Therefore, robustness will be one of the toughest issues to resolve.

Phone call or user’s interruption in the middle of data communication could be handled in many ways. One of the solutions is to cancel the transaction and sends the data again after the user resume. This method is most likely to be adopted in PHA because the CDMA version of the iPhone 4 and 4S cannot access cellular data while on an active phone call. Thus the cancellation of data transmission is inevitable. On the other hand, the PHA shall have the ability to cache or journal the data during analytical process in case the process is interrupted unexpectedly.

In addition, a data must be sent to the right PHA client. The Middle-Layer Server handling access policy must be ensured to send the data to the right client each time. An exhaustive unit testing is required for the sake of security. And the Middle-Layer Server should be able to handle large network traffic to not to crash. Therefore, a pressure test is needed to measure its limit.

**Wei Lin**

**User Friendliness**

The user friendliness of the PHA mobile application is an important quality for the system. The user interface should be designed in a way in a self-explanatory way in which novice users should be able to get familiar with the application quickly. The new operating iOS6 and the newer android operating systems for mobile applications offers many good features that would enhance the user friendliness of the application. This includes voice commands that would be able to help the user enter data more easily. For example, the user would be able to enter glucose levels by simply using their voice rather than enter the data by hand. Features like the multi-touch gestures can also be integrated to improve navigation between screens. Many of the new smartphone hardware also includes pedometers and accelerometers, which can be integrated to allow system to enter in data automatically without the user having to enter data by hand. There should be also options for the user to create custom workouts that can be saved so the user can simply select the workout rather than enter numerical data, which can be hard for the user to figure out. The same concept can also be applied in the user diet section of the application where the user can create meals/food that the user often have. The main challenge for the user friendliness for the application is to optimize productivity which involves limiting the time a user spend on imputing data but also have accurate feedback for the user.

**Performance**

The performance of the PHA application is an important factor in software quality. The performance of the PHA can be broken up into two parts. First, the front end of the application will be run using a modern day smartphones running on Apple’s iOS or Google’s Android devices. Both of the technologies offer many features that could be implemented into the application. For the PHA applications, the pedometer and accelerometers can be used to collect movements, which can be used to collect workout data for the user. The multi-touch gestures can also be implemented to increase productivity for the user when navigating the application. The mobile system can also have some challenges. First, the different hardware has different performances in terms of CPU speeds, and cellular data speeds. Different software can also have its challenges. iOS and android have different features that can be difficult to implement into the application. By using Titanium platform, two sets of native code for both iOS and Android can be generated from one set of Titanium code.

The performance of the system in the backend is also important for managing a large amount of data for the user. First, the software has to be able to upload and download data efficiently through Microsoft HealthVault. Due to possible large amount of traffic, a secondary MySQL database can be used to store some temporary data before being pushed onto Microsoft HealthVault. The download and upload speeds can also be affected by the cellular networks which can differ depending on location. To limit the use of slower cellular network speeds, the application should be designed to store some essential data locally on the device.

**Matthew Swircenski**

**Maintainability**

Maintainability is an important aspect for medical software like PHA, as updates and fixes need to be rolled out quickly to minimize any potential damage. Due to being a mobile application, PHA will be available on Apple’s and Google’s app stores. This lets users easily download and install an update to the software. The software can also check on its own for updates, and notify the user when they are available.

**Reliability**

As a medical application, PHA needs to be reliable. Users need to be able to depend on the software to produce consistent and accurate results, so they can make appropriate decisions about their personal health or the health of their patients.

 Reliability can be achieved by using the simple algorithms to analyze data, as there is less potential for bugs to occur. Having well-structured, maintainable code also improves reliability, since it will be easier for the developers to understand how the system works, and thus how to change it without breaking anything.