Sensor Drive Mobile application for health awareness

The SSURE (Software System for User Running Evaluation) app for Android: a design that won't let you down

Patomporn Loungvara

Dissertation 2014

Erasmus Mundus MSc in Dependable Software Systems

NUI MAYNOOTH

Ollscoil na hÉireann Má Nuad

Department of Computer Science

National University of Ireland, Maynooth

Co. Kildare, Ireland

A dissertation submitted in partial fulfilment

of the requirements for the

Erasmus Mundus MSc Dependable Software Systems

Head of Department : Dr Adam Winstanley

Supervisor : Dr. Joseph Timoney

July 2014

Declaration

I hereby certify that this material which I now submit for assessment on the program of study leading to the award of Master of Science in Dependable Software System, is entirely my own work and has not been taken from the work of the others save to the extent that such work has been cited and acknowledge within the text of my work.

Patomporn Loungvara

_____

Acknowledgement

I would like to thank supervisor Dr. Joseph Timoney from the National University of Ireland, Maynooth for all his helps and supports on this project, Eoghan Conway from the National University of Ireland, Maynooth for all his advices about the Android Development, and everyone who volunteers to run in order to collect the data and respond for the survey in this project.

Abstract

There has been a significant increase in the number of mobile applications concerned with health awareness due to the increase in the number of people who are concerned about their health and the raise in the number of people using smartphone/tablet devices. The development of applications related to health and exercise has become popular both in industry and academia. In this project we focus on development of a mobile application that can capture a user's running movement and modify an audio file so that there is a synchronisation between the beats of the music and the kinetic data (cadence or Steps per Minute/SPM) to motivate and guide their exercise. Our approach applies time-frequency analysis to obtain the SPM value by using the *Lomb Periodogram* technique that can effectively process unevenly sampling data, which is a feature of the data captured from the built-in accelerometer sensor on a smartphone/tablet device. In order to process the time-stretched audio file that is adjusted with the running information, the *Phase Vocoder* technique was used to transform the sound to different speed without changing the pitch. Its sophisticated frequency-domain sound processing suits our project's objective. To guide the implementation of these algorithms, several Software Engineering techniques have been used to manage our project. The *Agile Development Lifecycle (SDLC)* technique known as *SCRUM* was used throughout the development process in the design, testing, and implementation phases. This technique allowed us to change the plan if it was necessary, so it suited our project which was dealing with a new technology to be implemented within a short and limited timespan. Finally, we presented our evaluation to determine the accuracy of the results from our approaches and to assess the quality of our application. The results of evaluation showed that our approaches for the functional requirements were effective and gave us accurate response. However the non-functional requirements still needed to be improved and it was found that a new mobile-oriented approach for software metrics is needed if we wanted to achieve our goals fully.

Contents

1. Introduction

1.1 Health Awareness Technology

Considering the number of fitness centres and sport clubs available in modern society, it is certain that people nowadays are more concerned about their health than ever before. In general, good sports clubs offer the chance to have a trainer help to build and sustain an exercise regime. However, the cost of joining a sport club can be quite high, so some people just prefer exercising by themselves. Consequently, jogging has become one of the most popular activities because it does not require specific skills, equipment, or place. Nevertheless, enthusiasts would appreciate any methods by which they are able to evaluate and guide their exercising as it is a strong motivator.

In general, when we talk about health awareness technology, almost everyone might think of a special medical device connected by wires to a sensor which is attached to the body as illustrated in Figure 1.1 and controlled by an expert doctor or scientist. This image of health awareness technology could scare some people who are not familiar with it and make them avoid monitoring their health using this type of approach. However, present technology is far more powerful and portable than this and health awareness solutions have become simpler and more usable for everyday life.

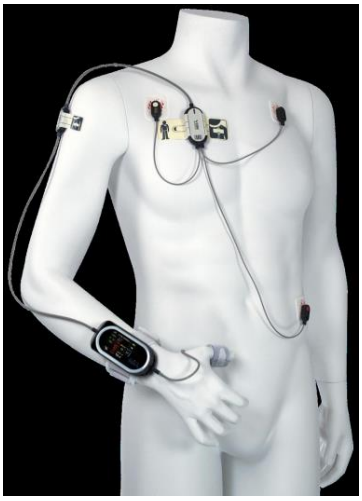

Figure 1.1 Health awareness devices connecting with external sensor [1]

It is indisputable that the development of technology for health awareness relies on powerful computational processing which can handle large amounts of data efficiently to extract information that can be utilised to provide feedback to the user. It also requires a comfortable method which is portable and handy while users will also still demand an acceptable cost for it. The advances in computer capabilities mean that it is now possible to create an autonomous technological solution

that has a low financial cost. It can be seen that many technological products which support guided exercise and health awareness, both hardware-based and software-based, have been released in the past decade, for example a step counter (*Jamie et al., 2009)* and heart rate monitors (Mohaned et al., 2008). Other hardware-based devices that many people might know are pedometers which can provide objective measures of the physical activity (David et al., 2010). One advantage of those devices is that they are objective as they record the number of steps. Consequently, people who use these devices can be motivated to increase their activity level by monitoring their daily steps (David et al., 2010). Some devices are presented as a combination of hardware-based and software-based devices. One of the successful companies producing products concerned about health awareness is Fitbit (Fitbit, 2014). Their products detect a user's activity including steps, distance travelled, calories burned and sleep by using a small device which can be attached to user all the time such as a bracelet or clip to capture the user's movement and sync that data to PCs or smartphone/ tablet to process the statistics and give user the result. This data can be tracked as the user progresses and can motivate user to increase their activity for their health benefits. However, these hardware-based products are still expensive and have limited functionality. The user is required to buy a new device in case they need to update for a new feature.

Software-based products have become more interesting currently, since computing devices are more portable and handy while also being more efficient. It is undeniable that the most remarkable electronic device in this age is a smartphone/tablet which is as powerful as the desktop computer while its size is smaller and suitable for carrying around. Therefore, mobile technology has become the best choice that can support various conditions for health awareness. A number of supportive training applications on mobile devices are now popular to those people who are concerned about their health.

Our project concerns the development of health awareness software on a mobile device, especially to monitor running or jogging information, in a manner which should be accurate and robust. We aim to search for various knowledge and techniques which can be utilized to assess the quality of software. The main question is that whether the application which will satisfy both its functional and non-functional requirements can be developed robustly under the development process using software engineering knowledge. If this is successful, it will improve our knowledge of mobile development for health awareness applications which is becoming such a popular topic now.

1.2 Mobile Technology for Health Awareness

As we know from the previous section that the capacities of mobile devices are now powerful enough to deal with health awareness application. By using off-the-shelf sensors, for instance the accelerometer sensor which is a basic built-in sensor in a mobile device, combined with a high performance of processor in smartphone/tablet device, it is possible to design an application that will monitor a user's activity (Nicholas et al., 2011). It is not difficult to capture movement data from built-in sensor and process these relevant data to find the valuable information about the running performance. Finally, this running information can be displayed to the user as a feedback in formats that can help them to appraise their performance.

Furthermore, running information can also be processed and used to drive a biofeedback system that can motivates and guides a user's exercise in real-time. For example, the data from the sensors can be analysed and special features from it are used to adapt the tempo of songs on a music playlist that the user is listening to while exercising (Bart et al., 2010). The goal of this approach is to synchronise the physical movements with the tempo of the music, resulting in users being able to listening to music while their exercise is detected and guided appropriately. This affects to user's running speed which should respond directly to the tempo of song.

Besides the functionality of a smartphone/tablet that we have talked, there is other abilities leading it becomes the best choice to develop health awareness application in our project. Considering the development of mobile applications, there are two famous platforms for mobile application; iOS (Apple Inc., 2013) and Android (*James et al., 2011*). These two platforms are powerful and have a high population of users. They both support a ubiquitous physical activity measurement due to their built-in accelerometer sensors. However, we will focus at an Android platform in this project. There are three main reasons to choose Android as the platform for our development over the iOS platform. First, the Android platform can operate on various manufacturers' products and devices while iOS platform can only be operated on Apple devices, for example, iPhone, iPad, etc. This limits the number of users to only Apple customers. The second reason is that Android devices are generally cheaper than iOS devices, which makes the distribution of Android's users wider than iOS's user. Lastly, the Android platform is open source and provides a community to support any problem during development. This is of benefit for a new Android developer which could face some problems while developing the program. Besides these reasons, the Android platform provides variety of other abilities. We will describe these in the next section.

1.3 Android Smartphone/Tablet Devices' Abilities

Smartphones/tablets currently are as good now as powerful computers; they have been developed to have a diversity of functionalities with a high performance. This means that it should be straightforward for a developer to create an application which solves our problem. From the large capabilities provided in a smartphone/tablet, we can consider some specific features which support our solution and identify how we can utilise them effectively in our project. For this thesis, we will focus on the Android platform which is an operating system designed primarily for touchscreen mobile devices. Android is popular mobile platform for both users and developer due to its numerous abilities. We will present some particular abilities which support our project as in the list below.

- Portability

It is obvious that the size of smartphone/tablet is small and reasonably robust meaning that it is comfortable to be carried around while the built-in accelerometer sensor is sensitive enough to capture user movement. This means that the smartphone/tablet is well suited for capturing running movement data. This is in comparison to a normal laptop which is less suited for capturing such data due to its larger size and more restricted portability.

- Processor

A smartphone/tablet is presently composed of a high-speed processor. This confers the ability to process data in real-time.

- Built-in sensors

Built-in sensors are a common feature on a smartphone/tablet. There are many choices of sensor that can be fitted to the device depending on its model, but the basic selections item in new devices are normally accelerometers sensors (Maxim et al., 2012). Fortunately, those built-in sensors enable the device itself to capture movement data directly and this simplifies data collection. This is more convenient than getting the data from an external sensor which in general will be more difficult and more complicated.

- Android SDK

Developing a mobile application is sometimes laborious, because the conditions between the development environment and the device platform are different. Having a good Software Development Kit (SDK) helps developers to create and manage the developed application easier. The Android SDK (*James et al., 2011*) is useful in which it is simple and powerful, requires no licensing fee, supports cross-platform development, and provides good documentation and important

libraries. Moreover, there is a developer community where they share experiences and help each other to fix any problems on the platform.

- Interactive Interface

An attractive and understandable Interface is the key that can persuade a user to engage with the application. Nowadays, most of the smartphone/tablets are designed to be an easy-to-use touch screen device with good graphics capability. Hence, it facilitates many design possibilities that requires a user-friendly interface that a user can enjoy working with.

In addition, although we are focusing on the built-in sensor to capture the movement data, we also have the option to interact with an external sensor. For that reason, we need to ensure that such a sensor can be connected easily. Luckily, an Android smartphone/tablet has a great ability to connect with a network or other gadgets using Bluetooth (Apple, 2004) or Ant+ (Joseph, 2014).

According to the list of an Android device's abilities given above, we are satisfied that the smartphone/tablet's abilities entirely support our solution's plan. Therefore, Android is a good mobile application platform for us to start with. Next, we will discuss about the significant data which is relevant to our project.

1.4 Running Information

Now we know that mobile technology is capable of being used for the capture of movement and it suits to being employed for our health awareness application, we are going to consider the health information which can benefit the application's users. Jogging or running is a popular form of physical activity. It offers many health benefits, such as helping to build strong bones, strengthen muscles, and help maintain a healthy weight. (Kara D. 2013) Users may have a different goal to achieve from their exercise, but they do appreciate information that can guide their running, for instance, the cadence (the number of steps a runner takes in a set amount of time), number of strides, their pace, and the distance covered. From observation, efficient runners can achieve at least 180 SPM (Strides per Minute) (Chantelle W. 2011). However, setting a goal of 180 SPM number for beginners might lead to the risk of injury. As a consequence of this, measuring the cadence becomes more significant when evaluating a jogging activity. In the variety of available mobile apps that are related to running training, information such as cadence and the pace are interesting because these two values can be captured and analysed from the built-in sensors, that is, the accelerometer sensor. In general, there are many acceptable algorithms to analyse and filter the captured data, and obtain from it the desired information. Most of them are the threshold-based algorithms. After analysis

process, those two values can be displayed or interacted with by users of the application as shown in Figure 1.2.

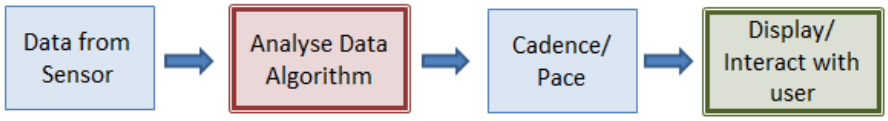

Figure 1.2 Data Processing Steps in available mobile apps.

While the simplest method is to set the goal for the cadence and activate an alarm when the captured speed reaches the set rate (*Jonathan I. 2014)*, the most effective choice is to adjust the tempo of songs to the running speed (beats per minute) (Alanna G. 2014). This will provide a clear auditory cue to the runner that should occur without interrupting their running activity. An ambitious goal would be to synchronise the physical movements with the rhythm of the music to encourage the feet to match the cadence. Our project approach will focus on processing data from sensor by frequency-based algorithm which can gives us running information about cadence. This value can be displayed on screen as feedback and also processed with a song that responds to user (see Figure 1.3). Developing this response to music would be of benefit by users to keep their speed constant and inspire them to be patient with their practice to reach their long term goals that could be specified in beats per minute set by the music. The integration of music and kinematics through technology has become a popular research topic due to the availability of new technological platforms that can support novel developments. This also then has an impact on new applications for Health and Well-being. We will discuss some works that are associated with our project in Chapter 2.

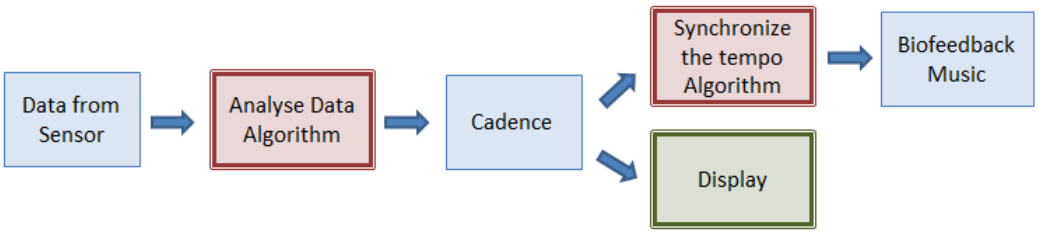

Figure 1.3 Data Processing Steps in our project.

Apart from this, there is additional information which is valuable for health-related goals that sometimes are associated with running, for example, maintaining a particular heart-rate or counting the number of calories burned (Andong et al., 2012). These features would require measurement using sensors that are external to the mobile device. Designing such sensors would be difficult and it is outside the scope of this current work. Hence, we will focus on core elements to this application;

the information that we can measure directly such as the time taken and distance travelled; such information can be extracted by analysis of the data from the phone's in-built sensors, that is, the cadences or steps per minute. This information can be displayed to the user graphically and is also used to drive an auditory signal though adjusting the tempo of a piece of music to be in synchrony with the user's running speed.

1.5 Software Quality in Health Awareness Application

A definition of software quality is that the degree of conformance to specific functional requirements and specific software quality standards and Good Software Engineering Practice (GSEP) (Daniel G. 2004). Besides the functionalities that our application could have, we should pay attention to other concerns about non-functionality which might affect to the quality of this application. In general, there are many software quality factors that have been suggested over the years. They are categorized to the McCall factor model, consisting of 11 factors grouped into three categories. Nevertheless, the alternative factors models specify some more factors which could also affect the quality of software. All factors are presented as follows. (Daniel G. 2004).

- **Product operation factors:** Correctness, Reliability, Efficiency, Integrity, Usability.
- **Product revision factors:** Maintainability, Flexibility, Testability.
- **Product transition factor:** Portability, Reusability, Interoperability.
- **Factors of the alternative models:** Verifiability, Expandability, Safety, Manageability, Survivability.

However, a different application requires different factors to evaluate their quality of software. The relevant software quality factors depend on the goal and requirements. Dealing with a health awareness application, there are some factors that are more relevant to its quality; Correctness and Efficiency become our main focus. An accurate result and a high level of performance are crucial. For such applications, developers always have to keep in mind that users will always need an application that will always work correctly, that is, it should be dependable, otherwise they will be frustrated and simply not use it, or it may give the user the incorrect information which may diminish the benefits of exercise, or in the worst case, the users may get an injury from exercising too much (Banerjee et al., 2012). Another point to consider is the data collected may be noisy because the user movements will not be smooth (Totura N. 2012), (Ayub et al., 2012). The system has to be robust, connecting some hardware sensors to the body, and then collecting the data may not always be reliable. Additionally, the interaction between the application and the internal sensors on the device should be consistent, especially in terms of the sampling time. To avoid this issue, in this project, we

will not include the data capture by an external sensor and just focus on the data from the device's built-in sensor. Lastly, the interface to the application should be easy for users to understand and interact with in order to provide convenience for them while they are using it. In addition, it should be attractive so that they enjoy working with it.

How can we measure the quality of software to determine whether the software for this application solving a problem or has a high performance? Software metrics is the essential technique which can measure characteristics of the software, for example, we can get some sense of whether the requirements are consistent and complete, and whether the design is of high quality (Norman et al., 1997). The specific metrics should be chosen for a particular project depending on that project itself. Therefore, Software Metrics is the only technique which is used to assess the quality of software.

Since Software Metrics are essential to good software quality, we need a strong knowledge of it to determine competent the most suitable metrics by which we measure the attributes of our health awareness application whether its result is accurate and whether these attributes demonstrate what we intended to achieve with our application such as, whether the system is robust, and whether the interface has a good design. To make this clearer, in Chapter 2, we will also provide information about software metrics and their measurement. Then, the detail about defining the metrics for this project which will be used to evaluate the software quality will be described in Chapter 6.

1.6 Summary

The dissertation is organized as follows: Chapter 2 gives an overview of the related work to our mobile health application combined with a discussion on the metrics that were used for the measurement of these applications and their results. Next, in Chapter 3 we present the theoretical background on which we built our system. We will detail the mathematical principles behind the algorithms and discuss their operation that is described in detail in the solution for data analysis in Chapter 4 and solution for software dependability in Chapter 5. Finally we present our evaluation results in Chapter 6 and we give our conclusion in Chapter 7 where we will also discuss some ideas for future work.

2. Related Work

In this chapter we critically analyse other systems and papers that have researched related problems to our domain or have been especially influential to our solution choice. Section 2.1 talks about different algorithms for the step detection which is used to analyse the input signal data, especially from an accelerometer sensor, to obtain the running information. We will also describe their testing experiments and the metrics they used to evaluate their systems. In Section 2.2, we discuss algorithms that have been proposed to adjust the tempo of music and some systems that attempt to sync music with a user's movement. We also critically analyse their measurement approach as it might be useful for us for evaluating our own project. After studying the previous work we then suggest how their methods can be extended and improved upon, which will lead us to present our proposed solution in Chapter 4 and Chapter 5.

2.1 Step Detection

Nowadays, technology can play a key role in guiding exercise and health awareness. Due to the capability of current computers and sensors, it is possible to create an application which can detect and monitor some basic human activities such as walking and running. This had led to Step Detection becoming a more interesting research area. A variety of techniques using step detection algorithms have been published. Most of them use accelerometers to assess the physical activities and act as the reference devices.

In health awareness applications, the performance of accelerometry-based step detection algorithms is more popular than other approaches. One paper (Michael et al., 2008) aims to compare the performance of four freely accessible step detection algorithms using data from the accelerometer: Pan-Tompkins, Dual-Axial, Wolf, and Autocorrelation, all in a non-laboratory setting. They set up samples of healthy and mobility-impaired subjects to distinguish the factors which affect the performance of these algorithms. They calculated the error rates from comparing the results against the real number of steps as counted using video recordings. The results of this experiment showed that the threshold-based algorithms (Dual Axis and Wolf) perform better for the sample of healthy persons, but that they work poorly on the data of mobility-impaired elderly subjects. In contrast, the peak detection approaches (Pan-Tompkins and Autocorrelation) shows acceptable results for both groups of subjects. However, in this paper they mentioned other step detection performance algorithms in the literature. In particular, the algorithms based on finding the Fourier transform of accelerometer signal can achieve an accuracy of up to 100% in a laboratory setting. It is

noted that the disadvantage of this algorithm is that the data need to be pre-processed. (Mathie, 2003)

Another journal paper by (Idin K. et al., 2013) presents the RRACE algorithm to detect cadence from running data using a Lomb-Scargle periodogram approach to deal with the problem of non-uniformly sampled data. This solves the problem with using the Fourier transform approach which is designed for uniformly sampled data only. It performs a spectral analysis on a four-second window of sampled 3-axis accelerometer data. The study was validated on the Android-based platform. Data was collected from the built-in accelerometer sensor on the smartphone/tablet device. The experiment required subjects to walk a known distance, and they monitored the independent variables of window size, body location, and speed of movement. The primary metric was the "Error Ratio", defined as the ratio between the step frequency measurement produced by RRACE and a manually measured step count. The results of this experiment lead to the conclusion that the interaction effects of both body location/movement speed and of body location/window size on the Error Ratio are significant. It performs best with an Arm, Bag, Belt, or Front pocket location for the device. The best window length of RRACE is 4 seconds and it is most sensitive to very slow speeds.

Due to the abilities of smartphones, especially running the Android platform, and the dramatic increase in the number of users, step detection by a mobile application has been considered in a number of recent works. One of these (Melis et al., 2012) aims to compare the accuracy of step detection between an algorithm running on the Android and a commercial pedometer. The algorithm which is used in this research relies on the detecting of peaks in the data produced by accelerometer sensor. The test required the subjects to walk and run at different specific steps per minute and number of steps. First, the subjects were required to wear clothing that restricted the movement of the phone while it is inside the subject's pocket. The test was performed using the smartphone and the commercial pedometer at the same time. Then, another test setup was designed where the devices were placed in a loose pocket of the subject. Comparing the results from the smartphone and commercial pedometer in both experiments leads to a smartphone-based step monitoring algorithms. It performed better than a commercial pedometer in terms of step count.

2.2 Synchronization with Music

Music and, more specifically rhythm, is a very natural way to motivate or synchronise the regular repeated movements of a person or group of people. Due to the increase in usage of portable music players during exercise, a number of research papers have considered their role in improving the development of new personal exercise aids. Synchronizing users' motion with the

period of audio is considered to act as a source of motivation during exercise which can guide users to run at the expected speed.

One of well-known technique to control alter the tempo of a piece of music is the Phase Vocoder which can be used to process an audio signal to speed it up or down. A paper (Jason et al., 2009) presented a system for the real-time automated tempo synchronization of an audio track to a runner's pace which uses the Phase Vocoder to adaptively adjust the music tempo. The system is designed to use two-dimensions of an accelerometer signal to track the step and estimate the step frequency. A sliding-window buffer was used for the real-time data analysis. Once a buffer is filled, the input signal will be processing using the autocorrelation technique to find possible beat periods. These values are then sent to the Phase Vocoder implementation to create a time scaling rate which affects the output audio tempo, without modifying the pitch. To evaluate a system, the performance was recorded over time along with the link between the steps per minute (SPM) and beats per minute (BPM). The subjective evaluation results demonstrated that the audio transformations sounds as expected, with minimal disagreeable artifacts.

Another work (Bart et al., 2010) focused on entrainment in the synchronization of a user's walking with music. The framework of D-Jogger is introduced to use to create applications that make use of the body movement of a user and matches the tempo of the music (BPM) with the walking tempo of the user, switching songs when appropriate in real-time. The 3-axis accelerometer signal is captured and interpolated at a sample rate of 200 Hz with a second-order polynomial function. Then, the resulting signal is filtered using a low pass filter to reduce high frequency noise. Autocorrelation is applied and frequencies in the range of 30 to 120 SPM are only considered. At the same time a BPM-Aware Playlist is used to add or remove songs from the playlist and the playback speed of those songs will be changed to synchronize with the walking speed using the Phase Vocoder. D-Jogger provides several ways to visualize human entrainment to music, each having both advantages and disadvantages.

- BPM-SPM Plots is a plot where BPM and SPM values are plotted over time. This shows very little information about the entrainment but it is useful for validating the correct functioning of D-Jogger.
- Synchronization plots show the time between each step and the closest beat by plotting the synchronization value for different values of SPM and BPM. The value is captured between BPM and inversely proportional to the BPM or Anti-phase. If the synchronization value is close to 0, it means that the beat and the step occurred almost simultaneously. This plots' disadvantage is that they are hard to compare to one another.

- Phase plots show the transformation from the synchronization value to the phase angle where it is plotted in a circle that has been divided into 4 sectors. If the phase plots in the In-phase sector, it means the beat and step synchronize, and vice versa for the Anti-phase sector. This makes is easier to compare different sets of synchronization data.
- Phase histogram plots are adapted from Phase plots by creating a histogram of the phase angle. This gives an overview of the amount of time a user spends in a certain phase angle, giving an indication of the user's entrainment. The advantages of this representation are that it is easily comparable with other histograms and can be represented in a table for easier statistical analysis.

D-Jogger has carried out a pilot experiment with a total of 33 participants. They filled in a survey that asked about their personal jogging preferences and their experience after the experiment. The whole experiment was recorded and the synchronization between SPM and BPM was measured. The results show that the majority of steps were in sync with the music and that the users reported in the survey that they felt more motivated when in sync with the music. In summary, D-jogger focused on entrainment to synchronize music and walking as rhythmical processes. They presented a time-stretch method and in addition, their representation of entrainment by using four different types of graph was clear and comparable. However, D-jogger did not show the accuracy of the SPM measurement process and its accuracy.

2.3 Summary

In this chapter we have identified and critically analysed related systems that perform step detection and music synchronization. We discuss some research in terms of its technique, experimental approaches, and its results. In our project we propose to extend the work on the frequency domain signal analysis algorithm (that is, the Lomb Periodgram in Section 2.1) to detect user running speed and the audio transformation algorithm (Phase Vocoder in Section 2.2) to adjust tempo of music to the movement. We will also extend their evaluation and metrics so that we can more thoroughly assess the accuracy of our project's functionality. An important project goal is to ensure that we create a dependable application that can achieve both the functional and non-functional requirements. We first present the background knowledge of our system in Chapter 3 and then give our proposed solution in Chapters 4 and 5.

3. Background

In this chapter we formally describe the different concepts, terminology and algorithms that are used in our proposed solution that will be presented in Chapter 4 and Chapter 5 and in the evaluation in Chapter 6. First of all, Section 3.1 discusses how to measure running performance using the Lomb Periodogram that will be used to analysis the incoming sensor data. Following that the Phase-Vocoder algorithm will be applied to the music track to adjust its tempo in sync with any change in the running speed. Secondly, Section 3.2 considers the software engineering techniques that are useful to this project, they will be applied in the different phases of the development; the Agile Software Development Lifecycle (SDLC) technique is involve with the design, testing and implementation. Next, in Section 3.3 we will talk about the interactive design and user interface which is important when developing a mobile application. Lastly, Section 3.4 presents the Software Metrics which are used to evaluate the quality of software project.

3.1 Measuring Running Performance

As we have mentioned before, cadence is the number of steps (or strides) a runner takes in a set amount of time. It is often expressed in steps per minute or strides per minute. Other names for cadence include step rate, step frequency, and turnover (Rotaa et al., 2011). Cadence is also one of the components that determine the running speed (*Jonathan Ide-Don, 2014).* In our project, the data captured from the *a*ccelerometer sensor should be regular in line with the rhythm of the running. Time-Frequency analysis is a body of techniques and methods used for characterizing and manipulating signals whose statistics/frequencies vary over time (Boualem B., 2003). It should be a source of useful tools that would allow the incoming data to be analysed in term of step frequency or cadence. This Time-Frequency analysis should also be beneficial when we try to synchronize this cadence to the speed of a rhythmic piece of audio (measured in beats per minute). By using special algorithms, we can transform between the time-domain signal and the frequency-domain spectrum. This processing can be applied to both the sensors and audio signals.

The fast Fourier transform (FFT) (William et al., 1992) is an algorithm to compute the discrete Fourier transform (DFT), it has revolutionized the fields of mathematics, science and engineering by converting data based on time (or space) to frequency and vice versa. This transformation provides a significant tool for spectral analysis and is the basis for variety of signal processing algorithms. The FFT works effectively when it has been dealing exclusively with evenly sampled data. Unfortunately, the data captured from a mobile phone sensor has a time-varying sampling interval as a result from

monitoring sensor events which acquires raw sensor data whenever the sensor detects a change. It means that the captured sensor data, which we are dealing with, is unevenly sampled data. For this situation, the FFT cannot work alone to transform the signal. However, this can be overcome as there is a specific transform method available that was designed to handle the uneven samples problem.

The Lomb Periodogram (aka Lomb-Scargle, Gauss-Vanicek or Least-Squares spectrum), which was developed by Lomb, based in part on earlier work by Barning (Barning F., 1963) and Vanicek (Vaníček P., 1971), and elaborated by Scargle (Scargle J., 1982), is an essentially method of estimating a frequency spectrum for unevenly sampled data. The Lomb method evaluates data, only at the time points when it was measured. Suppose that there are *N* data points $h_i \equiv h(t_i), i = 1, \dots, N$. First, the mean and variance of data is found by the usual formulas (William et al., 1992),

$$\overline{h} \equiv \frac{1}{N}\sum_{1}^{N} h_i \tag{3.1}$$

$$\sigma^2 \equiv \frac{1}{N-1}\sum_{1}^{N} (h_i - \overline{h})^2 \tag{3.2}$$

Now, the Lomb normalized periodogram (spectral power as a function of angular frequency$\omega \equiv 2\pi f \ > 0$) is defined by (William et. al. 1992)

$$P_N(\omega) \equiv \frac{1}{2\sigma^2}\left\{\frac{\left[\sum_j (h_j - \overline{h})\cos\omega(t_j - \tau)\right]^2}{\sum_j \cos 2\omega(t_j - \tau)} + \frac{\left[\sum_j (h_j - \overline{h})\sin\omega(t_j - \tau)\right]^2}{\sum_j \sin 2\omega(t_j - \tau)}\right\} \tag{3.3}$$

Here τ is defined by the relation

$$\tan(2\omega\tau) = \frac{\sum_j \sin 2\omega t_j}{\sum_j \cos 2\omega t_j} \tag{3.4}$$

The fact from these equations shows that the data is weighted on a “per point” basis instead of on a “per time interval” basis in the FFT, leading to the ability to evaluate uneven sampling.

To implement the normalized periodogram, as given in Equations 3.3 and 3.4 might result in an inefficient algorithm. In Equation 3.3 and 3.4, each combination of frequency and data points calls to a trigonometric function which results in an operation count that can easily reach several hundred times N^2. The Fast Computation of the Lomb periodogram (Fast Lomb Periodogram) (William et al., 1992) was presented to speed up the calculation of Equation 3.3 and 3.4 which have an operation count of order $N \log N$. This method uses the FFT, but it is in no sense an FFT periodogram of the data. It is an improved evaluation of Equation 3.3 and 3.4.

The trigonometric sums that occur in Equations 3.3 and 3.4 can be reduced to four simpler sums. Defining

$$S_h \equiv \sum_{j=1}^{N} \left(h_j - \overline{h}\right) \sin\left(\omega t_j\right) \tag{3.5a}$$

$$C_h \equiv \sum_{j=1}^{N} \left(h_j - \overline{h}\right) \cos\left(\omega t_j\right) \tag{3.5b}$$

$$S_2 \equiv \sum_{j=1}^{N} \sin\left(2\omega t_j\right) \tag{3.6a}$$

$$C_2 \equiv \sum_{j=1}^{N} \cos\left(2\omega t_j\right) \tag{3.6b}$$

Then using trigonometric identities,

$$\sum_{j=1}^{N} (h_j - \overline{h}) \cos \omega (t_j - \tau) = \; C_h \cos \omega\tau + S_h \sin \omega\tau \tag{3.7a}$$

$$\sum_{j=1}^{N} (h_j - \overline{h}) \sin \omega (t_j - \tau) = \; S_h \cos \omega\tau - C_h \sin \omega\tau \tag{3.7b}$$

$$\sum_{j=1}^{N} \cos 2\omega\left(t_j - \tau\right) = \frac{N}{2} + \frac{1}{2} C_2 \cos(2\omega\tau) + \frac{1}{2} S_2 \sin(2\omega\tau) \tag{3.7c}$$

$$\sum_{j=1}^{N} \sin 2\omega\left(t_j - \tau\right) = \frac{N}{2} + \frac{1}{2} C_2 \cos(2\omega\tau) - \frac{1}{2} S_2 \sin(2\omega\tau) \tag{3.7d}$$

Ideally, the evaluation of the new Equations could be done using the FFT which would reduce the computational cost. However, due to the data being unevenly spaced, it cannot be evaluated by using the FFT directly. However, if this data can be made to be uniformly spaced, it would be possible to apply the FFT. This can be achieved by interpolation, or rather as in the published algorithm a reverse interpolation called extirpolation (William et al., 1992). It is the opposite of classical interpolation which uses several function values on a regular mesh to construct an accurate approximation at an arbitrary point. In contrast, extirpolation replaces a function value at an arbitrary point by several function values on a regular mesh. To do so, it makes sums over the mesh that is an accurate approximation to sums over the original arbitrary point. The accuracy of the extirpolation depends only on the fineness of the mesh (William et al., 1992).

Taking this into account, a general outline of the fast Lomb evaluation algorithm can be described as follows:

I. A user-selected mesh size should be checked to ensure it is large enough to accommodate a required oversampling factor that can accommodate several extirpolation points
II. The input data values h_i should be extirpolated onto the mesh.
III. Then, take the FFT to obtain S_h and C_h as in Equations 3.5a and 3.5b.
IV. A second extirpolation, it needed followed be an FFT to obtain S_2 and C_2 in Equations 3.6a and 3.6b.
V. Once these quantities are available, it is possible to then evaluate Equations 3.7a-3.7d, 3.4, and 3.3 respectively to get the periodgram.

We can apply the Lomb Periodogram to our application using a short-time interval to process each set of sensor data. The captured data will be kept in a buffer until the buffer is filled, then the set of data in the buffer will be processed in the Lomb Periodogram algorithm, so that we can find the maximum frequency. This value represents the cadence or SPM that will be used to synchronize with the music in the next step. Figure 3.1 illustrates this procedure as a block diagram.

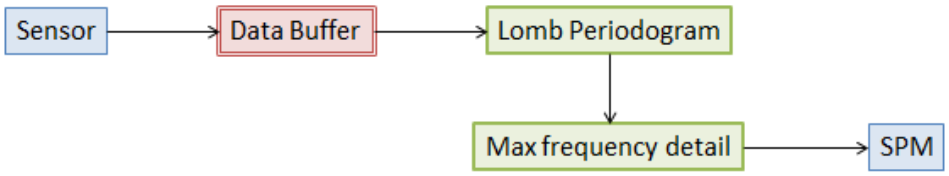

Figure 3.1 Sequence of data in the Lomb Periodogram

After applying the Lomb Periodogram to transform the time-domain signal from the sensor to frequency-based data, we obtain an average cadence value which is used to adjust the speed of the song. By processing the audio signal with these, synchronization between the running speed and the rhythm of song can be established. To complete this, we need a method which can produce a time-based transformation or stretching of sound to modify the tempo of the digital audio signal. In the next section, we will introduce an interesting technique and summarize its functionality briefly.

3.2 Transformations of a Digital Audio Signal

The Phase Vocoder is a well-known technique that uses time-frequency domain transformations to implement a variety of digital audio effects. This algorithm requires three stages; Analysis, Transformation, and Synthesis. The input signal for this method is a time-domain signal which will be

transformed into time-frequency based data matrix in the analysis stage in order to carry out the processing in the Transformation stage. After that, the modified data will be transformed again in the Synthesis stage back to the time-domain signal as the output of this method. In the transformation stage, the signal in Time-frequency domain will be processed to adjust its speed with respect to the result from the Lomb Periodogram as shown in Figure 3.2.

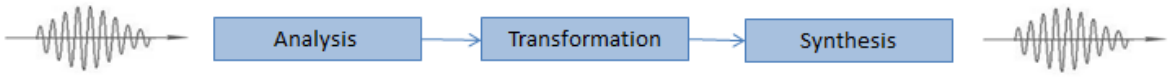

Figure 3.2 Three steps of Phase-vocoder [2]

To work with the Phase Vocoder (Zölzer et al., 2002), it is essentially a short-time Fourier transformation (STFT) performed on the windowed time-domain input signal. The input signal is first multiplied by a sliding window of finite length which is the same as size of spectral frame. Repeated application of this on each new frame yields a succession of windowed signal segments. Then, the FFT will be taken to transform these window signal segments to the spectral domain. From here, the short-time spectra in terms of its magnitude and phase are processed for whichever desired audio effect, for example, to synchronize the speed with the cadence. After that, each modified spectrum goes through an Inverse Fast Fourier Transformation (IFFT) and is again windowed in the time domain. Finally, the operation termed overlap-add is applied to the windowed output segments which yields the output signal and completes the technique. Figure 3.3 illustrates the various stages of the whole Phase Vocoder algorithm.

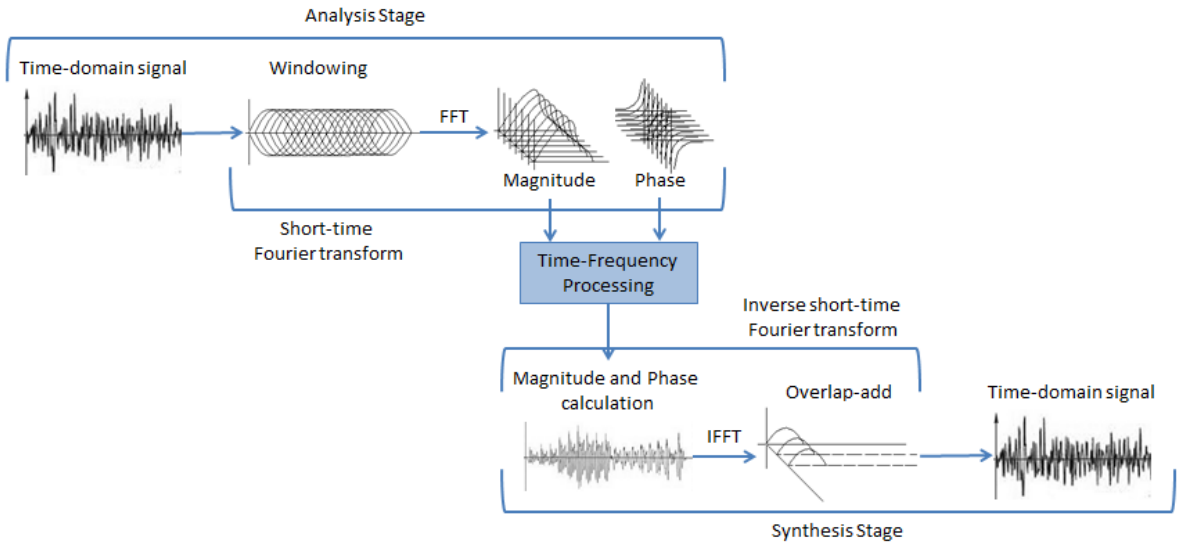

Figure 3.3 Phase Vocoder Process [2]

Considering Figure 3.3 in more detail, the signal is weighted by a finite length window before taking the short-time Fourier transform (STFT). The output of this process is a vector of complex

numbers which is in explicit polar form (magnitude and phase) of each spectral frame. This phase information is necessary for a perfect recovery of a signal without modification. In addition, Measuring of phase between two frames allow the evaluation of 'instantaneous frequencies' which is needed for introducing effects. In Time-Frequency Processing a number of operations may be performed to obtain a multitude of different effects and the tracking of phase changes leads to a finer grain indication of frequency contours in the input signal (Amalia et al., 2000). After we get the modified magnitude and phase of each spectral frame, they will be converted from explicit polar form back to the complex numbers before the inverse short-time Fourier transform is taken in order to obtain the finite length signal. These short-time segments will be weighted by the synthesis window and added by the overlap-add procedure together to acquire the time-domain output signal.

The basic concept of using the Phase Vocoder algorithm for time-stretching is that the signal is decomposed into frames that have a particular window length and the hopping from one frame to the next is given by the *hop size*. The *hop size* for the input signal is related to the *hop size* for output signal by the time-stretch factor. These indicate how far the output signal will be shifted from input in each window. Figure 3.4 shows the case where the input and output *hop size* are the same. Adding the overlapping frames together produces the output signal.

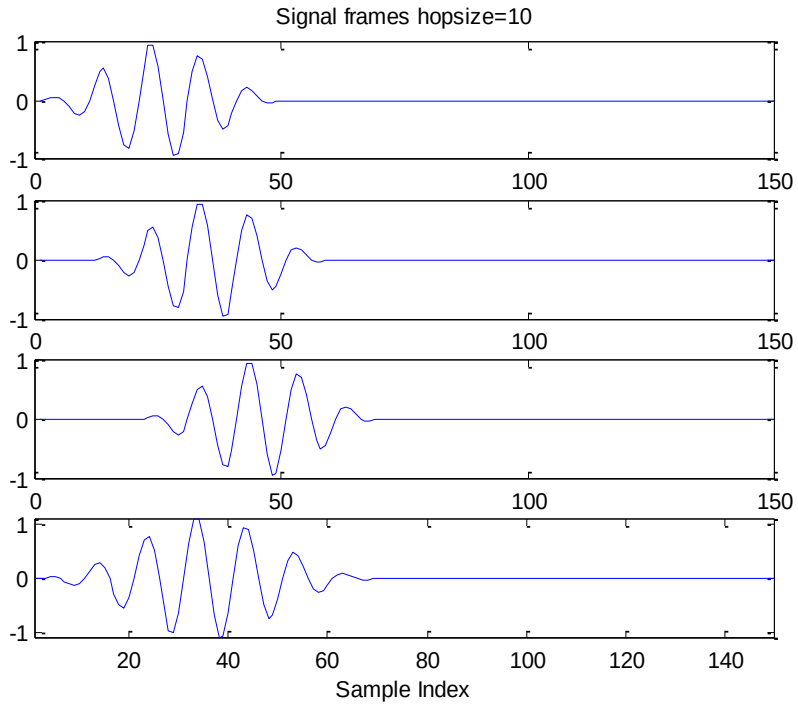

Figure 3.4 Input signal in each hop

If the output *hop size* is different the output signal can be stretched as the frames will overlap and add at a different point. However, if the frames add at a different point in the waveform then

the shape of the output can change and it will affect the output sound. To stop this happening the phase of the frequency components in each frame should be changed so that they add together properly. This is the idea behind the Phase Vocoder and it is illustrated in Figure 3.5

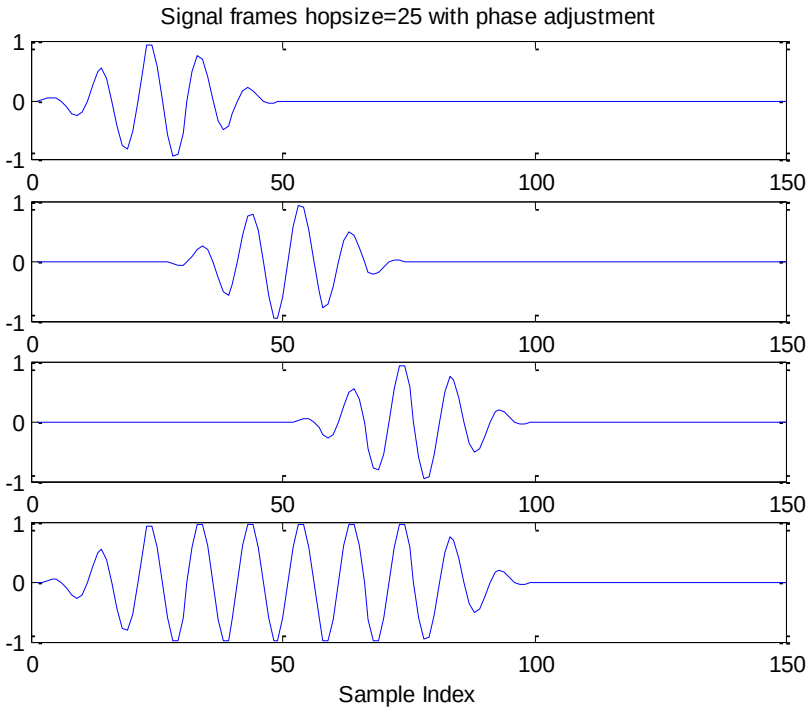

Figure 3.5 Output signal in each hope

The transformation of the digital audio signal is another important process that supports the main functionality requirement in our project. The Phase Vocoder is a very useful algorithm that we can use to work with sounds. Its sophisticated frequency-domain sound processing suits our project's objective.

3.3 Software Engineering

Due to the explosion in the development of large software systems in the past decades, the adoption of a software engineering outlook has become an essential requirement in any software production process for the control of the quality of the software before its delivery to users. Software engineering is actually an engineering discipline which is concerned with all aspects of software production. (Ian S, 2001) In general, software engineering espouses a systematic and organised approach to planning and development as this is often the most effective way to produce

high-quality software. (Ian S, 2001) Software engineering offers a number of methods for defining and managing a software production process. These methods consider every phase of the software development but different methods emphasise these aspects in their own way. The selection of a method directly affects the quality of the final application (Paul et al., 2012). We will now specify some software engineering considerations that are relevant to this development. Firstly, the design will be discussed.

3.3.1 The Design of Software

Software design is the process of implementing software solutions to one or more sets of problems. It affects the performance, robustness, distributability and maintainability of the system among other things. Well-designed software means a good quality of software which is manifested in the successful rating of users' satisfaction. A design is motivated by both the functional and non-functional requirements. Both of these come from the specification. However the functional requirements detail what this software system should do and are realised in the code itself, while the non-functional requirements specify the performance features that can be observed using particular metrics. The previous Section 3.1 explained the algorithms that were brought together to make up the application. These algorithms answer the functional requirements. The non-functional requirements for the application are user-oriented and concentrated on giving a good experience and keeping user interest. These requirements should be expressed in a number of quality attributes that can be measured, otherwise, they cannot be assessed and do not fit within the planning of software engineering (Ian S. 2001). For this health awareness application the following requirements, as outlined in the list, were chosen to be the most important to ensure that it will have a lasting positive effect on users.

- Usability

Usability is a measure of how easy it is to use a product to perform prescribed tasks. Usability in software design represents an approach that puts the user, rather than the system, at the centre of the process, which incorporates user concerns and advocacy from the beginning of the design process and dictates that the needs of the user should be foremost in any design decisions. (Microsoft Corporation, 2000) Interaction design and User Interface becomes more significant in this requirement. The most visible aspect is usability testing when users work and interact with the product interface and share their views and concerns with the designers and developers. The best reason to perform it is to make the application to be a zero-training application. It is important, especially for mobile applications, which users like to interact with straight away based on their experience.

- Efficiency

Efficiency is the ability of the software to do the required processing on the least amount of hardware. Efficiency is relevant to the software performance. It is generally measured in terms of quantity, quality, coverage, timeliness or readiness, within the target or software standard. Software should be designed to localise critical operations within a small number of sub-systems to reduce component communications (Ian S. 2001).

- Dependability

The original definition of dependability is in the ability to deliver services that can be trusted justifiably. Dependability is an integrating concept that consists of the following attributes. (Arash R., 2010)

- Reliability: continuity of correct service. It is the ability of a system to continue operating in the expected way over time. Reliability is measured as the probability that a system will not fail and that it will perform its intended function for a specified time interval. (Microsoft Corporation, 2009)
- Safety: absence of catastrophic consequences on the user(s) and the environment. It is consistent with the potential harm that could result from the loss, inaccuracy, alteration, unavailability, or misuse of the data and resources that it uses, controls, and protects. Software should be used with most critical assets protected in the innermost layers and with a high level of security validation applied to these layers. (Ian S. 2001)
- Integrity: absence of improper system alterations. It defines the consistency and coherence of the overall design. This includes the way that components or modules are designed, as well as factors such as coding style and variable naming. A coherent system is easier to maintain because it is obvious to know what is consistent with the overall design. (Microsoft Corporation, 2009)
- Maintainability: ability to undergo modifications and repairs. It is the ease with which a product can be maintained in order to isolate defects or their cause, correct defects or their cause, prevent unexpected breakdowns, and etc. In some cases, maintainability involves a system of continuous improvement - learning from the past in order to improve the ability to maintain systems. Software should be designed using self-contained components that may readily be changed. Producers of data should be separated from consumers and shared data structures should be avoided. (Ian S. 2001)
- Availability: readiness for correct service. It means the data can be easily accessed any time. Software should be designed to include redundant components so that it is possible to replace and update component without stopping the system. (Ian S. 2001)

These requirements bring about to dependability, describing the ability of a system or component to function under stated conditions for a specified period of time. As we mentioned before, dependability is strongly required in our health awareness application

3.3.2 Agile Software Development Lifecycle (SDLC)

The Agile Software Development Lifecycle (SDLC) is a process for managing complex software projects and the emphasis is on building releasable software within short time periods. (Stephen et al., 2012) It encourages rapid and flexible response to change and promotes adaptive planning, evolutionary development and delivery, and a time-boxed iterative approach. We will specify some characteristics of the Agile development which are relevant to our project.

- Iterative, incremental and evolutionary

Agile methods break tasks into small increments with minimal planning and do not directly involve long-term planning. Iterations are short time frames and involve a cross-functional team working. At the end of the iteration, the release of production will be demonstrated to stakeholders. This minimizes overall risk and allows the project to adapt to changes quickly.

- Quality focus

Specific tools and techniques, such as continuous integration, automated unit testing, pair programming, test-driven development, design patterns, domain-driven design, code refactoring, and other techniques are often used to improve quality and enhance project agility.

There are many well-known agile software development methods and/or process frameworks include. The technique known as SCRUM (Schwaber Ken, 2004) which is for managing software projects and product or application development will be used in this project. A key principle of SCRUM is its recognition that during a project the requirements can change any time, and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, SCRUM adopts an empirical approach—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly and respond to emerging requirements.

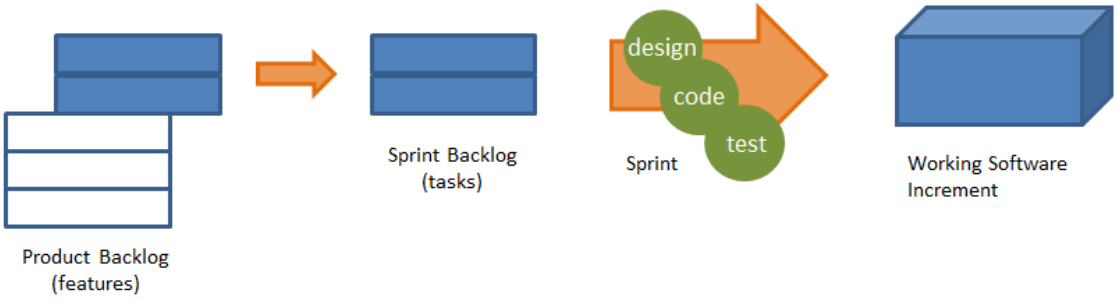

Figure 3.6 SCRUM Process [3]

By this method, the development plan will be organised into an iterative and incremental development. It starts with the Product Backlog which is a prioritized list of all product features from the backbone to the optional features. Features correspond to the elements of the product backlog where it is tackled in iteration and each item is further broken down into subtasks that are implemented. This is called a Sprint. During each Sprint, a list of tasks, called the Sprint Backlog, is maintained. Software development using SCRUM technique allows multiple teams to take on product increments in parallel. (Brown et. al. 2012) Then, working a software increment is released at the end of each sprint. (Figure 3.6)

Although this project is an individual piece, using SCRUM may benefit in that the project is split into product features that are developed within short time periods for release where time limits mean it is preferable to produce the app one feature at a time rather than altogether, just in case we run out of time, it allows for contingency planning. Also as we are dealing with new technologies, we need to adapt quickly if the specifications have to change.

3.3.3 Software Testing

Testing in the software process is the key activities to ensure the quality of a software product. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Proper software testing procedures can reduce the risks associated with software development. Poor quality leads to more failures, increased development cost, and delays in staining product stability (Brown et al., 2012).

There are a number of difficulties that can cause software failure. This includes collecting the user requirements correctly, specifying correctly the required software behaviour, correctly design the software, correctly implementing the software, and correctly modifying software. One of engineering approaches to develop a correct system is a form of checking that the output of each step of the development (User requirements, Software Specification, Software Design, and Implementation) is tested to see that it meets its specifications (verification) and meets the uses needs (validation). (Brown et al., 2012)

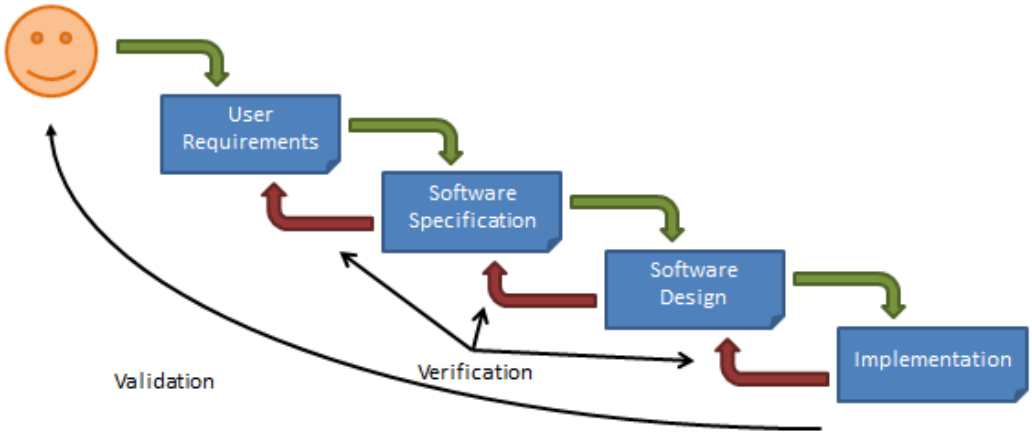

Figure 3.7 Development Process [3]

In the Development Process, there are three key characteristics of software, which are the user requirements, functional specification and a number of modules that are integrated together to form the final system (Figure 3.7). We need four key test activities to verify each of these; Unit testing, Integration testing, System testing, and Acceptance testing.

- Unit Testing verifies an individual unit of software to make sure that it works correctly. Test data is selected to ensure that the unit satisfies its specification.
- Integration Testing tests two or more units to make sure that they interoperate correctly. Test data is selected to exercise the interactions between the integrated units.
- System Testing tests the entire software system as a whole to make sure that it works correctly, and subsequently to make sure it meets the user's needs, or solves the user's problem. Test data is selected to ensure the system satisfies the specification.
- Acceptance Testing tests the entire software system as a whole to make sure it meets the use's need, or solves the user's problem. Test data is selected to ensure the software system meets the users need or solves their problem.

Testing in incremental development in Agile is different from other development models. The incremental model begins with a simple implementation of a part of the software system, resulting in testing becoming an important part of the incremental model and is carried out at the end of each iteration (Brown et al., 2012). This means that testing begins earlier in the development process and that there is more of it overall. This approach gives good results as the testing and coding can work in parallel from the start of project.

3.3.4 Implementation

Implementation is also a critical part of software development. Good planning of the implementation is required for a successful product. It needs a detailed listing of activities, costs, expected difficulties and schedules that are required to achieve the objectives of the strategic plans. It is based around the future-state map and should comprise the objectives, to-do lists and other devices that will help develop the process (Wenso Software Solutions, 2013).

In SCRUM, we break down the required features into small enough pieces and arrange a user story map to outline the big picture of the product. On top are the big stories/activities that are the essentials or backbone of the software. These backbone items cannot be prioritized one over another. The software will be built incrementally, story by story, choosing them from the story map left to right, and top to bottom. There is a slow and steady movement across the backbone, and then down through the priorities of each rib. This is a small implementation of the system that performs a degree of end-to-end functionality. (Figure 3.8) We call this implementation a Walking Skeleton since the most important tasks are from the backbone items (Scrumsense, 2011). Because we are using the Walking Skeleton rather than a strict architectural design, it suits to our project where the requirements can change at any time resulting from having to deal with a new technology and tight time constraints.

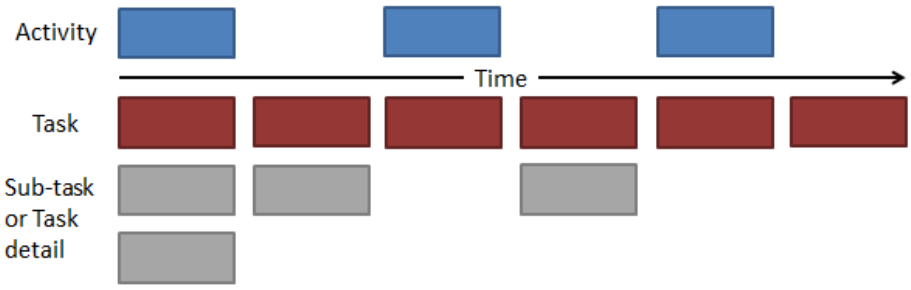

Figure 3.8 Walking Skeleton diagram

3.4 Interaction Design & User Interfaces

By interaction design, we mean “designing an interactive product to support the way people communicate and interact in their everyday and working lives” (Rogers et. al. 2007) which leads interaction design to focus on the users’ experience and to enhance the way they work. It is different from software engineering because interaction design is more concerned about people and their interactions with each other while software engineering is interested in issues to do with realizing

the project, such as cost, duration, and structure. A good design is important to make users appreciate the application.

There are four basic activities involved with the process of interaction design. (Rogers et. al. 2007)

I. Identifying needs and establishing requirements for the user experience.
II. Developing alternative designs that meet those requirements.
III. Building interactive versions of the designs so that they can be communicated and assessed.
IV. Evaluating what is being built throughout the process and the user experience it offers.

These activities are intended to inform one another and to be repeated. Eliciting responses from potential users about what they think and feel about what has been designed can help explicate the nature of the user experience that the product evokes. Broadly speaking to users, there are two type of design: Conceptual and Physical, from which the design will emerge iteratively. The conceptual model will capture about what the product will do and how it will behave, while the latter is concerned with details of the design such as screen and menu structures. For users to evaluate the design of an interactive product effectively, designers must produce an interactive version of their ideas. The activities concerned with building this interactive version are called Prototyping and Construction.

A Prototype which is a limited representation of a design that allows users to interact with it and explore its suitability (Rogers et al., 2007) becomes a useful aid for users and stakeholder to evaluate and discuss about the design. It is often necessary to try out ideas by building prototypes and iterating through several versions. And the more iterations, the better the final product will be. Prototypes are important because they answer questions and support designers in choosing between alternatives. There are different kinds of prototype depending on the purpose. It is used to explore the ideas in early stages of development and never intended to be kept and integrated into the final product.

After we have designed our product and created the prototype with the core functionality, how would we find out whether the product would appeal to users and if they will use it? Evaluating what has been built is very much at heart of interaction design. Its focus is on ensuring that the product is usable. The wide diversity of interactive products gives rise to a range of features that evaluators must be able to evaluate. There are three main approaches, those are: (1) usability testing; (2) field studies; and (3) analytical, evaluation to evaluate the interaction design have several methods associated with it such as observing users, and user surveys and questionnaires. For example,

usability testing and field studies both involve observing users and asking users but the conditions under which they are used, and the ways in which they are used, are different. (Rogers et al., 2007)

Interaction design is another technique which makes the application becomes more effective. Rather than just implement the functionalities that fulfil users' need, a good interface design can indicate whether user will continue to user our products or not.

3.5 Software Metrics

We have presented some knowledge about testing and evaluation in previous section. Now we will describe in more detail about software metrics which is an important technique for evaluating our application. This technique is necessary to assess the status of projects products, processes, and resource to control our projects. Software metrics is a term that embraces many activities, all of which involves some degree of software measurement (Norman et al., 1997). Firstly, we will consider the data collection activity. It is clear that the quality of any measurement program depends on careful data collection. But in fact, collecting data is not easy, especially when data must be collected across a diverse set of projects. Thus, we have to ensure that measures are defined unambiguously, that collection is consistent and complete, and that data integrity is not at risk. It is acknowledged that metrics data collection must be planned and executed in a careful and sensitive manner. (Norman et al., 1997)

After collecting data, we have to model specific metrics in order to be used to measure and evaluate software quality. These kinds of metrics depend on the characteristics of the product and the goals of the evaluation. The basic set of metrics should be of a limited number in order to maximize the visibility of each measurement, and consist of objective and/or subjective measurement (Möller et al., 1993). This makes it easier to explain the metrics program and avoids creating opposition among those that have to implement it to prevent the build-up of problems such as more paperwork that keeps people off the real job. (Figure 3.9) (Möller et al., 1993)

- Objective Metrics

Objective metrics are easily quantified and measured (Möller et al., 1993). Examples of objective metrics are: Program size, Effort, Schedule, and Number of Faults. Program size is used to derive meaningful quality and productivity indicators. This measurement can be made by counting lines of code or using function points. Effort measurement can be counted as staff-days or project cost, while Schedule can be measured by elapsed time for a phase or phases. And the number of Faults is a primary indicator of product quality, which can be counted at various phases of software development life-cycle including after initial delivery.

These basic metrics can become the basic for a hierarchy of additional metrics, some of which may be calculated. One of the advantages of objective metrics is that they can often be collected more cost effectively using software tools. The metrics definition and collection mean should be described within a written Metrics Plan.

- Subjective Metrics

Subjective metrics attempt to track less quantifiable data such as quality attitudes, for example the measurement of customer satisfaction. Data for a subjective metric would often be collected through interviews or surveys. Subjective data could be captured as response classes defined by reference points on a scale.

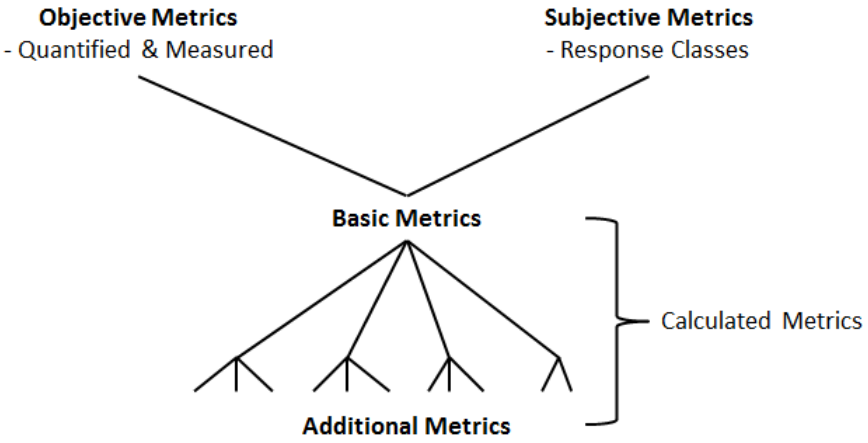

Figure 3.9 Summary various types of metrics [4]

After the goals of metrics have been identified, the initial basic set of metrics can be defined. The metrics will be used to measure performance with respect to the goals. There are some characteristics of good metrics for the purpose of the selection of metrics that are most applicable to specific corporate development environments. These characteristics of metrics can be grouped into organisational characteristics and technical characteristics. (Möller et al., 1993)

Organisation characteristics of metrics apply to the application of the metrics within a corporate organisation environment. These are the good characteristics of metrics that we should be concerned with. These are important for their application to the software process and project management: Highly visible, Consistently applied, Management interest and support, Organisational acceptance, Politics, Responsibility and control, Historical data available, Product development process correspondence, Process Opportunity targets, and Patience.

The Technical characteristics of metrics apply to the definition of the metric itself. It contains a limited Number of metrics whose attributes include: Easily calculated, Readily available data, Precisely defined, Tools support, Experimentation, and Standard.

For this project, the standard that we will focus is the ISO/IEC 25010:2011 (replacing ISO/IEC 9126-1:2001). This standard focuses on one main point, a quality-in-use model composed of five characteristics that relate to the outcome of interaction when a product is used in a particular context of use, and a product quality model composed of eight characteristics that relate to the static properties of the software and dynamic properties of the computer system. (ISO, 2011)

For our project, we will define both objective and subjective metrics to measure the quality of software. In selecting objective metrics, we will focus on measuring the accuracy of the result from the algorithms that we use. Considering the subjective metrics, we will find the metrics that measure the efficiency and usability of software. User surveys and questionnaires will be tools to check the user's feedback in this metrics.

3.6 Summary

In this chapter we presented different terminology and background concepts that we will refer to in the following chapters as we describe our proposed solution. In the beginning, we talked about the analysis of the measured running data using the Lomb Periodogram. Then, we introduced a well-known technique named the Phase Vocoder which is useful to modify a digital audio signal. By this technique, we can adjust the tempo of the song with to keep it in line with the rate of running. Next, we described some of Software engineering techniques and some knowledge on Interaction design and User interfaces that we will use in the development process for this project. Lastly, we described the Software metrics, types of metric and their good characteristics in order to consider their application for the evaluation of this project. The following chapters will describe our prototype, analysis, evaluation, and results.

4. Solution for Data Analysis

As described earlier in Section 3.1, we need specific methods to convert the signal data from the time-domain to the frequency domain. In this chapter we will first detail (Section 4.1) a case study data collection method and how the experiment can effect to the analysed results. Then, we will present the method that we use to extract the cadence value from the sensor data in Section 4.2 and the method that synchronizes the rhythm of music with the cadence in Section 4.3. Finally, we will present how to implement the analysis process as an app on the Android device apps in Section 4.4.

4.1 Case Study Data Collection

In order to test the accuracy of our software for SPM analysis, we need a valuable case study data to be analysed using our developed method along with context data that records what the user is doing at the time. There are a various factors that might bring about an incorrect result from analysis process. Some conditions must be taken into account when we collect the sample data.

First of all, the Android device used to develop this project was a Google Nexus 7, which has a quad-core Qualcomm Snapdragon™ S4 Pro processor and 2GB of RAM (Google, 2013). The tablets were released on 27th June 2012 and run Android version 4.2. It supports many kinds of built-in sensor such as accelerometer, gyroscope, and gravity sensors. To capture running movement, we use the data from the accelerometer sensor which can measure the effect of movement. The ability of accelerometers to detect the kinematics associated with running as they move in sympathy to the heel contact is useful as they provide a signal from which the step frequency can be measured (Brendan et al., 2005).

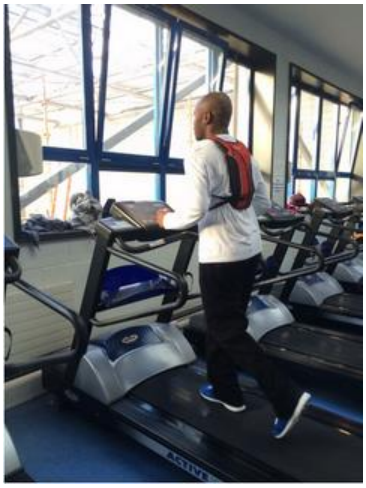

Figure 4.1 Experiment when subject runs on running machine

Eight subjects volunteered to participate in the study. In order to find a valuable case study data, each subject was required to run under different conditions; outdoor or on the running machine, for a running period within 5 minutes and longer. These different conditions present different results from which we can try to analyse the data and select the best case study. Note that in these experiments the body location for the device was controlled. Due to the size of device, it is not possible to attach it to the subject's arm or leg. Every subject will carry a backpack containing the device. This backpack must be tight to their body to reduce noise in the data from swaying of the backpack with respect to the movement. Ideally, it will just move in the one direction in response to the activity of running.

Another issue that must be addressed is that the data sampling from the built-in sensor on these Android devices is not strictly uniform. The time between the capture of each piece of data is not the same as the device will hold off recording data while it is performing some other operation on the phone. Thus, the sampling times are non-uniform. However, we found that constancy of time sampling data is significant, it affects the results considerably when we transform data from the time-domain to the frequency-domain. Thus, we have designed the interface to the mobile app so that it does not contain any unnecessary graphics or processing in order to ease any lags that might be caused by such a thing. We apply *SENSOR_DELAY_NORMAL* mode (*James et al., 2011*), which will refresh the sensor event with a frequency of about 45 Hz for data collection data apps. This mode is suitable for both short-time and long-time data collection. After we start to collect data, this app will insert the three-dimensional sensor data into the internal database whenever the sensor event is detected. Then, the data in database will be written to a CSV file when we stop and export the data.

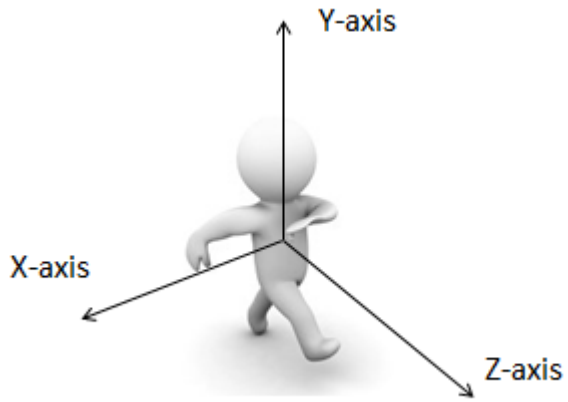

Figure 4.2 Three dimensions represented while running

From the experiments, we consider the longitudinal signal (Y-axis) to be the most indicative of the movement while the signals from moving forward/backward (Z-axis) or turning left/right (X-axis) in the case of outdoor running are assumed to be noise. We obtain significant data with less

noise overall and greater consistency from our subject who runs for a long period (about 15 minutes) on the treadmill machine. (Figure 4.3 and 4.4) This data is assigned to be the case study data when we try to analyse the sensor data in terms of its frequency content. The detail about the analysis will be described in next Section.

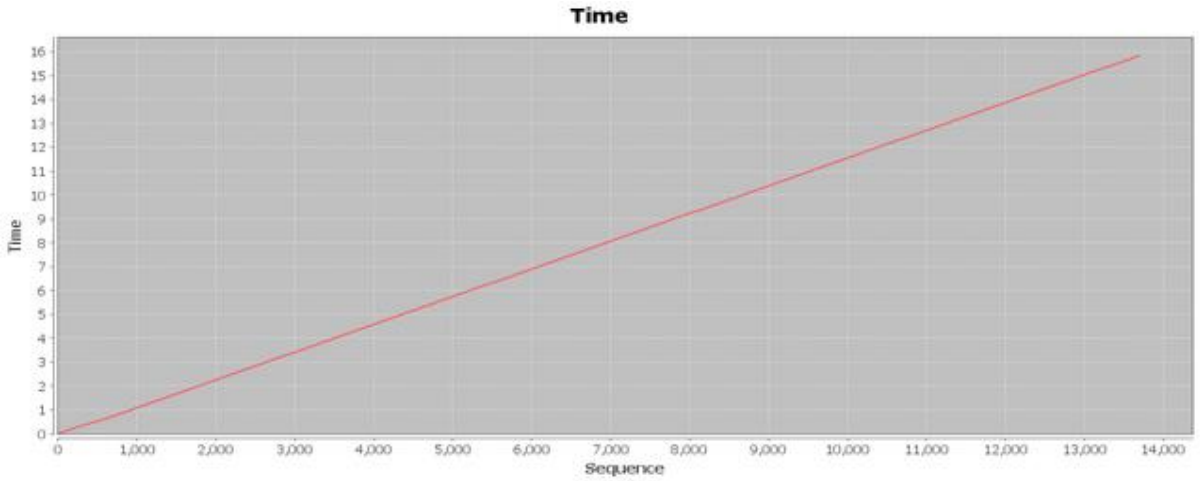

Figure 4.3 Graph shows that time sampling is approximately constant

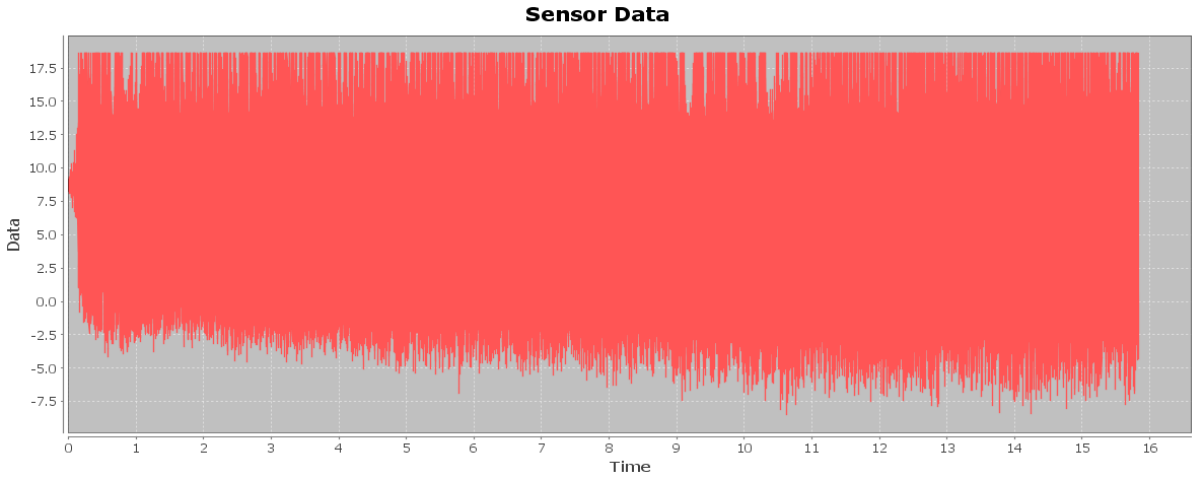

Figure 4.4 Graph shows incoming data from the longitudinal accelerometer sensor

4.2 Analysis the Frequency of Running Data

As we described in Chapter 3, the Cadence is the most significant information that we need to extract from the accelerometer data. Time-Frequency analysis is the most suitable approach to detecting cadence and the implementation of the Lomb periodogram is the most suitable algorithm for our unevenly sampled data.

Open source implementations for the Lomb periodogram are available in various languages such as C++ or MatLab (Hunt et al., 2001). However, we have to implement it in the Java language. Fortunately, the FFT method has already been implemented and tested in Java and is available as a

library (Jtransforms Library, 2011), so we only need to implement the Lomb periodogram itself. In term of the implementation, we use the Eclipse IDE, Version Kepler Service Release 2 on JavaSE 1.7 and follow the sample code from the *'Numerical Recipes'* (William et al., 1992) book and compare the output with a MatLab implementation (Christos S., 2008). The Lomb periodogram Java source code is shown as a table 4.1.

```
public class FastLomb {
        final private int MACC = 10;
        private int nt, nf, ndim;
        private double hifac, ofac, T, Tmin, mean, var, alpha, maxindex;
        private double[] wk1, wk2, mx, x, time, P, f;

        public FastLomb(double[] _x, double[] _time) {
                this.x = _x;
                this.time = _time;
                this.nt = x.length;
                this.ofac = 4;
                this.hifac = 1;
        }

        public void setOfac(double _ofac) { this.ofac = _ofac; }
        public void setHifac(double _hifac) { this.hifac = _hifac; }
        public double getAlpha() { return alpha; }
        public double[] getF() { return f; }
        public double[] getP() { return P; }
        public double getPmax() { return P[maxindex]; }
        public double getFmax() { return f[maxindex]; }

        public void doLomb() {
                double hypo, hc2wt, hs2wt, cwt, swt, den, cterm, sterm, M, Pmax;
                double[] real1, real2, imag1, imag2;
                double[][] W1, W2;

                init();
                Extirpolation();

                //FFT
                W1 = FFT.doFFT(wk1, ndim);
                W2 = FFT.doFFT(wk2, ndim);

                real1 = new double[nf]; for (int i=0; i<nf; i++) real1[i] = W1[0][i+1];
                imag1 = new double[nf]; for (int i=0; i<nf; i++) imag1[i] = W1[1][i+1];
                real2 = new double[nf]; for (int i=0; i<nf; i++) real2[i] = W2[0][i+1];
                imag2 = new double[nf]; for (int i=0; i<nf; i++) imag2[i] = W2[1][i+1];
                P = new double[nf];
                Pmax = -1;
                for (int j=0; j<nf; j++) {
                        hypo = Math.sqrt(Math.pow(real2[j],2)+Math.pow(imag2[j],2));
```

```
                        hc2wt = 0.5 * real2[j] / hypo;
                        hs2wt = 0.5 * imag2[j] / hypo;
                        cwt = Math.sqrt(0.5+hc2wt);
                        swt = Math.copySign(Math.sqrt(0.5-hc2wt), hs2wt/Math.abs(hs2wt));
                        den = 0.5 * nt + hc2wt * real2[j] + hs2wt * imag2[j];
                        cterm = Math.pow((cwt * real1[j] + swt * imag1[j]), 2) / den;
                        sterm = Math.pow((cwt * imag1[j] - swt * real1[j]), 2) / (nt-den);

                        P[j] = (cterm+sterm) / (2*var);
                        if (P[j]>Pmax) {
                                Pmax = P[j];
                                maxindex = j;
                        }
                }

                M = 2 * nf/ofac;
                alpha = M * Math.exp(-Pmax);
                if (alpha>0.01) alpha = 1 - Math.pow((1 - Math.exp(-Pmax)),M);
        }

        private void init() {
                int nfreq, nfreqt;
                double Tmax;

                nfreqt = (int) (ofac * hifac * nt * MACC);
                nfreq = 1;
                while(nfreq < nfreqt) nfreq *= 2;

                ndim = 2 * nfreq;
                mean = StatUtils.mean(x);
                mx = new double[nt];
                for (int i=0; i<nt; i++) mx[i] = x[i] - mean;
                var = StatUtils.variance(mx);

                Tmin = time[0];
                Tmax = time[nt-1];
                T = Tmax - Tmin;

                nf = (int) (0.5 * ofac * hifac * nt);
                f = new double[nf];
                for (int i=0; i<nf; i++) f[i] = i/(T*ofac);
        }

        private void Extirpolation() {
                double ck, ckk, fac;

                fac = ndim / (T*ofac);
                wk1 = new double[ndim];
                wk2 = new double[ndim];
```

```
                for(int j=0; j<nt; j++) {
                        ck = 1 + Math.floor((time[j]-Tmin)*fac)%ndim;
                        ckk = 1 + (2*(ck-1))%ndim;
                        wk1 = spread(mx[j], wk1, ck, MACC);
                        wk2 = spread(1, wk2, ckk, MACC);
                }
        }

        public double[] spread(double xi, double[] x, double c, int m) {
                int[] nfac = {1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880};
                int nx = x.length;
                if (c == Math.round(c)) {
                        int ic = (int) c;
                        x[ic] = x[ic] + xi;
                } else {
                        int ilo = (int) Math.min(Math.max((int) c-0.5*m+1.0, 1), nx-m+1);
                        int ihi = ilo+m-1;
                        int nden = nfac[m-1];
                        double cfac = c-ilo;
                        for (int i=ilo+1; i<=ihi; i++) {
                                cfac = cfac * (c-i);
                        }
                        x[ihi] = x[ihi] + xi*cfac/(nden*(c-ihi));
                        for (int k=ihi-1; k>=ilo; k--) {
                                nden = (nden/(k+1-ilo))*(k-ihi);
                                x[k] = x[k] + xi*cfac/(nden*(c-k));
                        }
                }
                return x;
        }
    }
```

Table 4.1 FastLomb.java

In Java, we implemented the FastLomb class so it can receive input values from the constructor which has arrays of input data and time. These two parameters are relevant and must have the same length. Then, the doLomb method will be called to process the main function of the Lomb peroidogram. Here, we separate it into three sub-methods; *init*, *Extirpolation* and *spread* which will be called under the loop in Extirpolation. The outputs from the *Extirpolation* method will be taken to by the FFT function before processing in the main loop to calculate the trigonometry functions explained in Chapter 3. This process will give us an array of magnitude values which are related to the frequency array which is calculated from time input data in the *init* method.

From this method, we can analyse the input data from the longitudinal accelerometer sensor and find the overall cadence frequency which corresponds to the SPM the subject has kept while running. This will be varying but we will be able to see significant peak(s) which represents the

cadence. The number and height of those peak(s) depends on how constant the rate at which the subject ran during the data collection time period. A very narrow peak means that the rate was perfectly steady while if the peak is broad around the base it means that the subject varied their SPM a lot during the run.

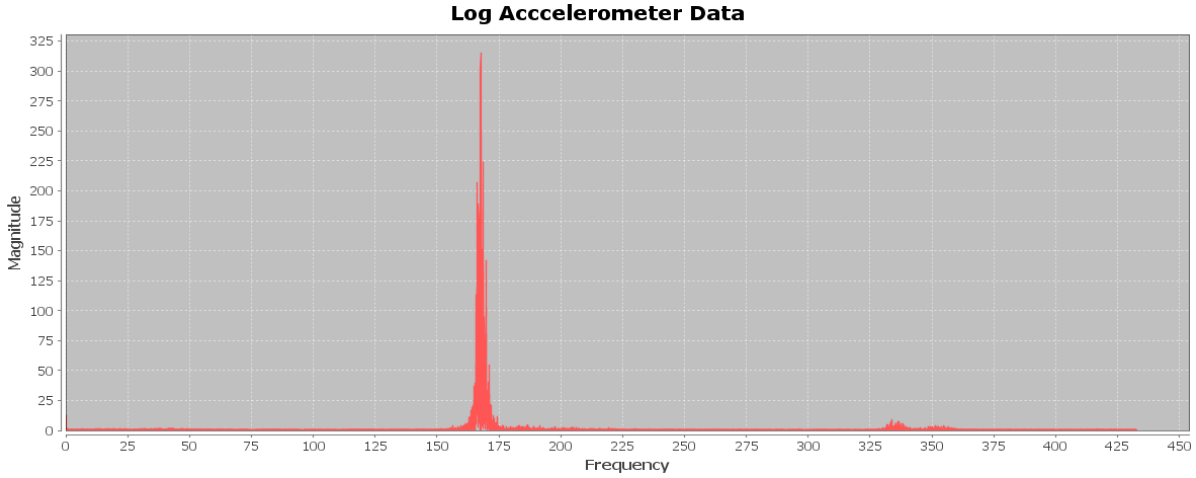

Figure 4.5 Result from Lomb Periodogram shows the maximum magnitude is around 170 SPM

From Figure 4.5, it is obvious that the highest peak magnitude is around the frequency at 170 Hz. We also can see a small peak around the frequency at 330 Hz. However, this peak frequency is not the normal speed that human can reach. We consider this peak to be just a harmonic of the main peak and can be ignored. The most significant peak of the data will be picked as indicating the cadence or SPM that is represented by the frequency at the peak.

According to our goal of app, that is to synchronize the tempo of song with the running speed or cadence, our app must process the data in real-time. We need to find the SPM within short-time intervals, so that we do not capture the whole signal over the total running time but rather after a specific time interval has elapsed, called the *frame size*, we apply the Lomb Periodogram to that data to work out the SPM for that particular time interval only. Thus, we are dividing the incoming data into small frames of a user-specific size, that is *frame size*, and calling the time delay at which every spectral frame is picked up from the signal the *hop size*. The values for this *frame size* and *hop size* can affect the performance by introducing unwanted noise and discontinuity in the resulting spectrum plots. To avoid these problems, we need to consider them carefully.

Some experiments on running data were carried out to see which parameters for the *frame size* and *hop size* produced the clearest results. We found that a frame size of 1024 and *hop size* of 10 were best to for these analysis experiments. This *frame size* corresponds to a time interval of approximately 0.85 seconds. Each spectral plot output from the Lomb Periodogram is then tiled in an ordering by time are displayed together as a depth map image just like the spectrogram, a very

popular time-frequency analysis tool used in many signal processing applications (McClellan et al., 1998).

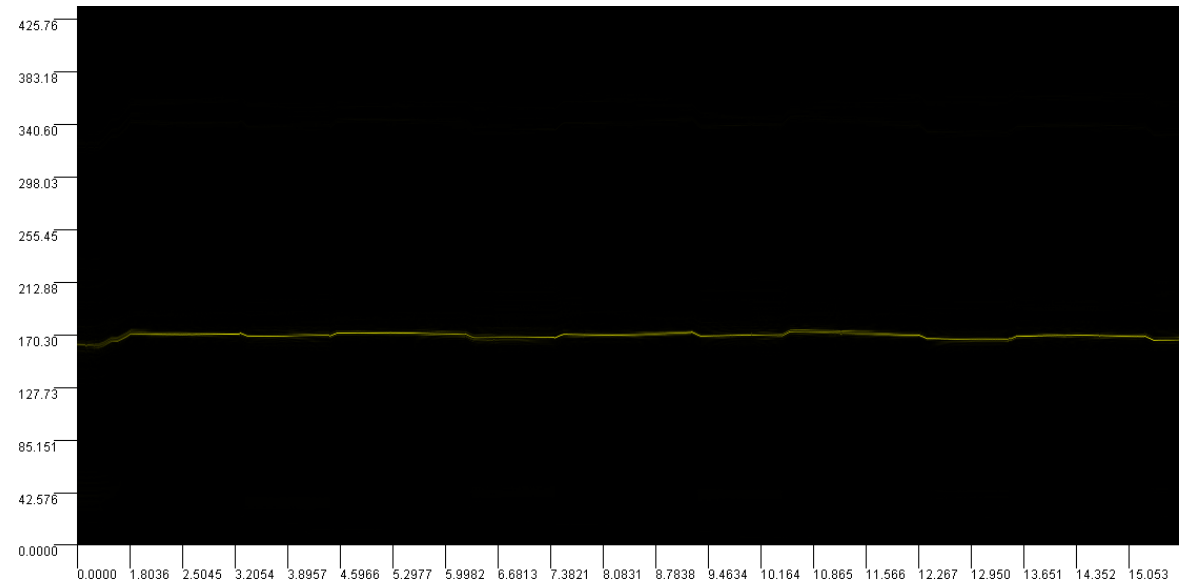

Figure 4.6 Lomb Spectrogram for constant running data

The spectrogram presents the information across time and frequency. Examining each slice of the graph, we can see the peak frequency that has a highest magnitude which represents to the average cadence in each frame. A single selected frame from Lomb Spectrogram is shown in the following plot, Figure 4.7.

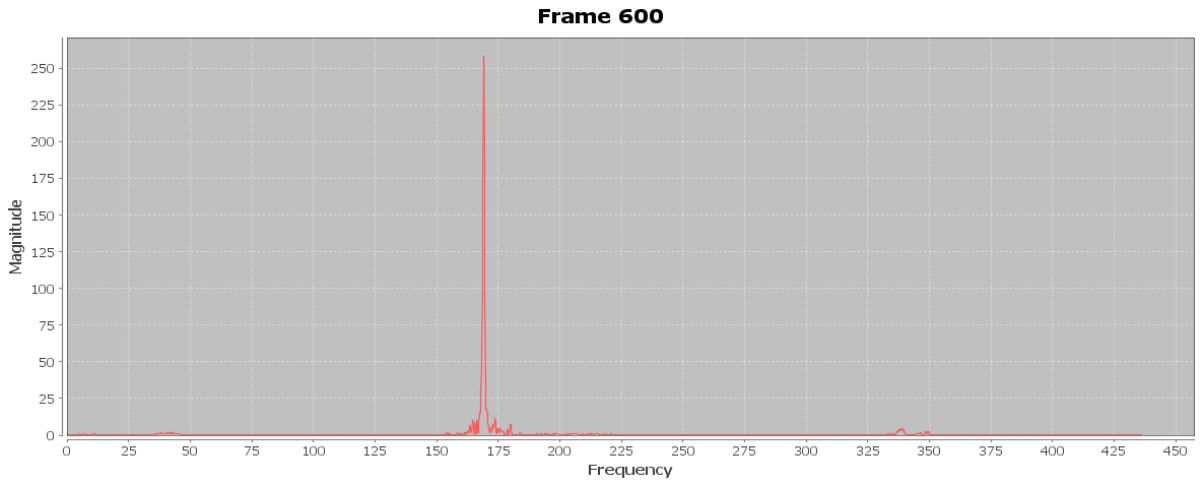

Figure 4.7 Lomb Periodogram Result from frame 600

We can see a clear horizontal line in Figure 4.5. This highlights the fact that the information in each time slice of the Lomb Spectrogram will have a significant peak that corresponds to the cadence of that time. This confirms to the previous results from FastLomb method (Figure 4.5) that we can see a single significant peak from analysing the whole data. It also similar to the frame results in Figure 4.7. Once we find the frequency/SPM at which that peak lies then we have the running speed

by which we can adjust the tempo of the music to provide biofeedback to the runner. This is explained in the next section.

4.3 Adjust the Music with respect to the SPM

In Chapter 3, we have discussed about a transforming digital audio signal using the Phase Vocoder. After we managed to get the value of cadence from actual running data, we need a method to modify an audio signal to adjust the tempo of song. The Phase Vocoder is an effective technique that uses the time-frequency domain to implement its transformation. It also suits our purpose of returning feedback in almost real time.

A book named *DAFX: Digital Audio Effects* (Zölzer et al., 2002) provides information about the implementation of the Phase Vocoder. Regrettably, it supports only the implementation in Matlab. Then, we also need to implement this algorithm in the Java language, in the same manner as we have done for the Lomb Periodogram. Luckily, the implementation of the FFT method in Java that we have mentioned in previous section also provides the Inverse FFT or IFFT method, so we need only be concerned with the implementation details of the Phase Vocoder algorithm itself. We are still in the same environment as when we implemented the Lomb periodogram. The accuracy and correctness of method can be checked by comparing the result with MatLab version. The source code of the Phase Vocoder in Java is shown as follows.

```
public class PVocoder {
        private static int n1 = 200; //can be changed
        private static int WLen = 2048; //can be changed
        private static double[] w1 = utilities.hanningz(WLen);
        private static double[] w2 = w1;

        public static double[] doPV(double[] input, double ratio) {
                int n2, L, Len, pin, pout, pend;
                double max, tstretch_ratio;
                double[] DAFx_in, DAFx_out, inputArray, omega, phi0, psi;

                DAFx_in = input;
                tstretch_ratio = ratio;
                n2 = (int)(ratio * n1);
                L = DAFx_in.length;
                max = utilities.getMaxAbs(DAFx_in);
                for (int i=0; i<L; i++) { DAFx_in[i] *= Math.pow(max, -1); }

                Len = (WLen) + (DAFx_in.length) + (WLen - (L%n1));
                inputArray=new double[Len];
                System.arraycopy(DAFx_in, 0, inputArray, WLen, DAFx_in.length);
```

```
                DAFx_out = new double[WLen + (int)(Math.ceil(L*tstretch_ratio))];
                omega = new double[WLen];
                for (int i=0; i<WLen; i++) { omega[i] = 2 * Math.PI * n1 * i * Math.pow(WLen, 1); }
                phi0 = new double[WLen];
                psi = new double[WLen];
                pin = 0;
                pout = 0;
                pend = Len-WLen;

                while (pin<pend) {
                        double[] grain, r, phi, omega_phi, delta_phi;
                        double[][] f, ft;

                        grain = new double[WLen];
                        for (int i=0; i<WLen; i++) grain[i] = inputArray[pin+i] * w1[i];

                        f = FFT.doFFT(FFT.fftshift(grain), WLen);
                        r = utilities.abs(f);
                        phi = utilities.angle(f);

                        omega_phi = ArrayOperations.SubArray(
                                        ArrayOperations.SubArray(phi, phi0),omega);
                        delta_phi = ArrayOperations.AddArray(omega, utilities.princarg(omega_phi));
                        phi0 = phi;
                        psi = utilities.princarg(
                                ArrayOperations.AddArray(psi,
                                        ArrayOperations.RatioArray(delta_phi, tstretch_ratio)));
                        ft = utilities.PolarToCartesian(r, psi);
                        grain = ArrayOperations.MultArray(FFT.fftshift(FFT.doIFFT(ft, WLen)), w2);

                        for (int i=0; i<WLen; i++) {
                                if (pout+i < DAFx_out.length)
                                        DAFx_out[pout+i] = DAFx_out[pout+i] + grain[i];
                        }
                        pin = pin + n1;
                        pout = pout + n2;
                }
                max = utilities.getMaxAbs(DAFx_out);
                for (int i=WLen; i<DAFx_out.length; i++) { DAFx_out[i] *= Math.pow(max, -1); }

                return DAFx_out; //play sound
        }
}
```

Table 4.2 PVocoder.java

The Phase Vocoder has been implemented in a single static function which calls a number of support methods. There are two constant variables; *n1* and *WLen* which are the interval size and

frame size of data respectively that we use to process the sound. These values can affect the quality of output sound, so that we found that a value for *n1* of 200 and *WLen* of 2048 were best for this experiment. In the function, there is a main loop that works after all basic variables are set and each loop will process the data in each interval and return the output which will be combined into an output array.

This PVocoder class requires a number of support functions, for instance, a number of different Array operations such as adding and shifting are required. These functions are implemented in a straightforward manner that was aimed to render the implementation convenient and make our code look tidy. Moreover, there are some specific functions that we implement necessarily for the Phase Vocoder algorithm. These functions serve a particular purpose to support the processing in our PVocoder class. We will describe the details of those functions in the following list.

- Hanningz function

This function generates a window function which is used to multiply each frame of input audio data. This window has a taper so that the ends of the frame data are close to zero. This has two purposes. Firstly, when recombining the frames of the signal at the end of the processing the window ensures that each frame adds together properly and that the result is smooth. Secondly, the windowing has an effect on the spectrum of the signal by smoothing the discontinuities at the edge of each frame it reduces the presence of artefacts in the spectrum of the frame data. (Figure 4.8) The source code is shown in Table 4.3 below.

```
public static double[] hanningz(int N){
        double[] window=new double[N];
        for(int n=0;n<N;n++){
                window[n]=0.5*(1-Math.cos(2*Math.PI*n*Math.pow(N, -1)));
        }
        return window;
}
```

Table 4.3 Hanningz function

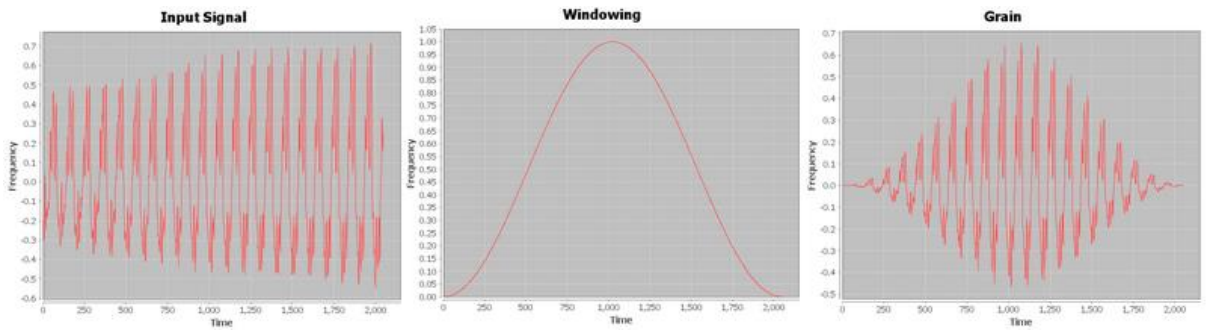

Figure 4.8 Result of weighting signal by windowing scheme from Hanningz function

- fftshift function

When applying the FFT it is assumed that the time origin starts from the left of the window which means that for an impulse located in the middle of the window its frequency components will have alternating phase values of 0 π 0 π ⋯ due to the cosine frequency components of that impulse having values of +1 -1 +1 -1 ⋯ (Amalia D. et al., 2000). In order to have simpler phase relationships which results in a better sounding output for the Phase Vocoder, the fftshit function shifts the signal around its time origin. Table 4.4 shows the implementation code of the fftshift function and the result from taking it is shown in Figure 4.9.

```
public static double[] fftshift(double[] input){
        int N=input.length;
        double[] output=new double[N];
        if ((N%2)==0){
                System.arraycopy(input,N/2,output,0,N/2);
                System.arraycopy(input,0,output,N/2,N/2);
        } else {
                int start=(int) (Math.floor((double)N)/2);
                System.arraycopy(input,0,output,start,start+1);
                System.arraycopy(input,start+1,output,0,start);
        }
        return output;
}
```

Table 4.4 fftshift function

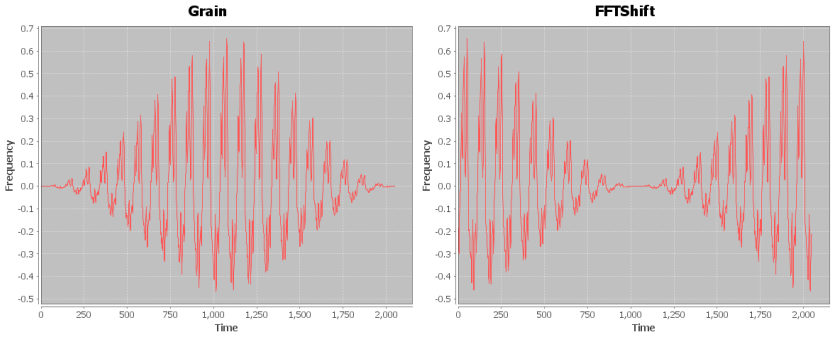

Figure 4.9 Result from taking fftshift function

- abs function

This function is implemented from MatLab function that returns the absolute value or magnitude for each element of complex array (two-dimensions array which contains real part and imagination part values), which means the same as

$$R = \sqrt{X^2 + Y^2} \tag{5.1}$$

where *R* is magnitude, *X* is real part of a complex number, and *Y* is imaginary part of a complex number. For example, if the input value is 4+3i, the result of this function will be 5. The source code for this function is shown in Table 4.5.

```
public static double[] abs (double[][] inp) {
        double [] out = new double[inp[0].length];
        for (int j=0; j<inp[0].length; j++) {
                out[j] = Math.sqrt(Math.pow(inp[0][j],2) + Math.pow(inp[1][j],2));
        }
        return out;
}
```

Table 4.5 abs function

- angle function

This function returns the phase angles, in radians, for each element of a complex array. The angles lie between ±π. This phase angle value can be expressed as

$$p = \tan^{-1}\frac{Y}{X} \tag{5.2}$$

where *p* is phase angle, *X* is real part of a complex number, and *Y* is imaginary part of a complex number. For example, if the input value is 3+4i, the result of this function will be 0.6435011087932844. Furthermore, the proper quadrant around the unit circle is also found so that the sign of the angle is correct. This is achieved using the atan2 function rather than simply atan. The source code for this function is shown in Table 4.6.

```
public static double[] angle (double[][] inp) {
        double [] out = new double[inp[0].length];
        for (int j=0; j<inp[0].length; j++) {
                out[j] = Math.atan2(inp[1][j], inp[0][j]);
        }
        return out;
}
```

Table 4.6 angle function

- PolarToCartesian function

After modifying the phase increment of signal, the signal will be recombined from each element of its magnitude and phase values to the complex array that represent the signal spectrum. The IFFT algorithm expects the input to be in the form of the Real and Imaginary parts that make up the signal spectrum. This requires a change from the polar coordinates of magnitude and phase to the Cartesian form of real and imaginary where the angles lie between ±π. This phase angle value can be expressed as

$$X = R\cos p \tag{5.3}$$

$$Y = R\sin p \tag{5.4}$$

Where *X* is real part of the complex number, *Y* is imaginary part of the complex number, *R* is magnitude, and *p* is phase angle. The example for this function is the inverse of two functions above. If input value magnitude is 5 and phase is 0.6435011087932844, the result of this function will be a complex array of which the real part value is 4 and the imaginary part value is 3. The source code for this function is shown in Table 4.7.

```
public static double[][] PolarToCartesian(double[] r, double[] p) {
        double[][] out = new double[2][r.length];
        for (int i=0; i<r.length; i++) {
                out[0][i] = r[i]*Math.cos(p[i]);
                out[1][i] = r[i]*Math.sin(p[i]);
        }
        return out;
}
```

Table 4.7 PolarToCartesian function

- princarg function

This is a simple function that returns the principal argument of the nominal initial phase of each frame. It is used in the Phase Vocoder when working out the phase of the signal spectrum for the time stretched signal. It was written by *Carlo Drioli* (Brendan et al., 2005). Table 4.8 shows the implementation code of the princarg function and the result from taking it is shown in Figure 4.10.

```
public static double[] princarg(double[] phasein){
        int N=phasein.length;
        double[] phase=new double[N];
        double a,k;
        for (int n=0;n<N;n++){
                a=phasein[n]*Math.pow(2*Math.PI, -1);
                k=Math.round(a);
                phase[n]=phasein[n]-k*2*Math.PI;
        }
        return phase;
}
```

Table 4.8 princarg function

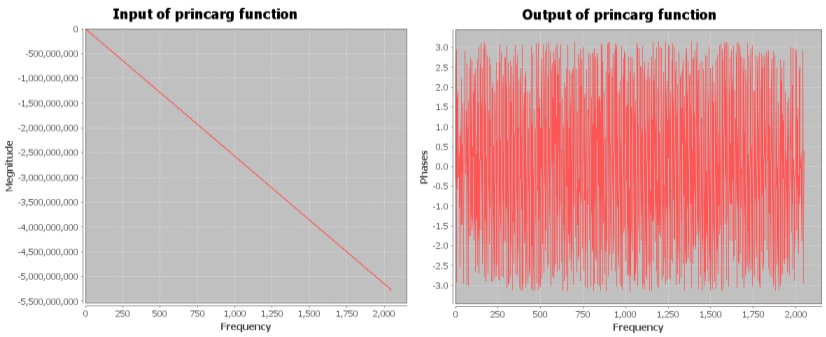

Figure 4.10 Result from taking princarg function

After we know the specific methods for PVocoder class and how they work, we can then focus on the processing in PVocoder. According to the Phase Vocoder algorithm, we can separate the source code into three section plus the initializations. In the initializations section, we prepare the input signal by padding the beginning and the end of input signal with zeros in order to avoid losing the signal structure as the window function first engages with the signal and is then hopped through the signal data. Then, in the analysis stage, we start working on the time-varying spectrum. We first apply the windowing and then take the FFT of each frame which is called a *grain* and the result of the FFT function will be expressed in explicit polar form. In the transformation stage, the phase in each frame is unwrapped first. It is then transformed according to a formula to reflect the desired time stretching factor. This factor changes the phase so that the transformed signal is longer or shorter but the frequency components stay the same. We control the time stretch factor using the SPM value from the Lomb Spectrogram. A deviation greater than the average SPM will shorten the time-stretch while a deviation less than the average SPM will result in a greater stretch factor. Finally, after the signal is modified, it will be recombined to Cartesian form before taking the inverse FFT transformation followed by an overlap-adding of each frame to recombine the frames and get the final output signal. The overall frame size is an important parameter and its size will determine the quality of the output signal. In general, a longer size will give better audio quality as long as the audio information in the signal is not changing too rapidly; otherwise a shorter frame is better. We will show how to consider this number again in the evaluation section. The graphs of results for a selected frame value in each step and input-output signals from PVocoder are presented in Figure 4.11 and 4.12 respectively.

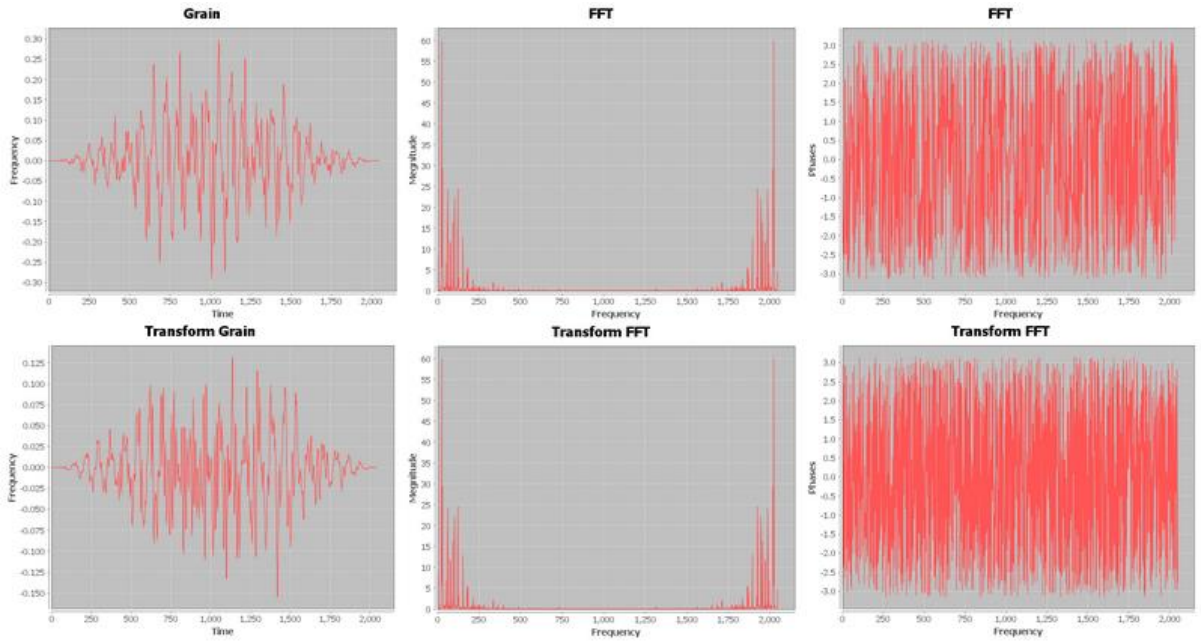

Figure 4.11 the results of selected frame from each step in PVocoder

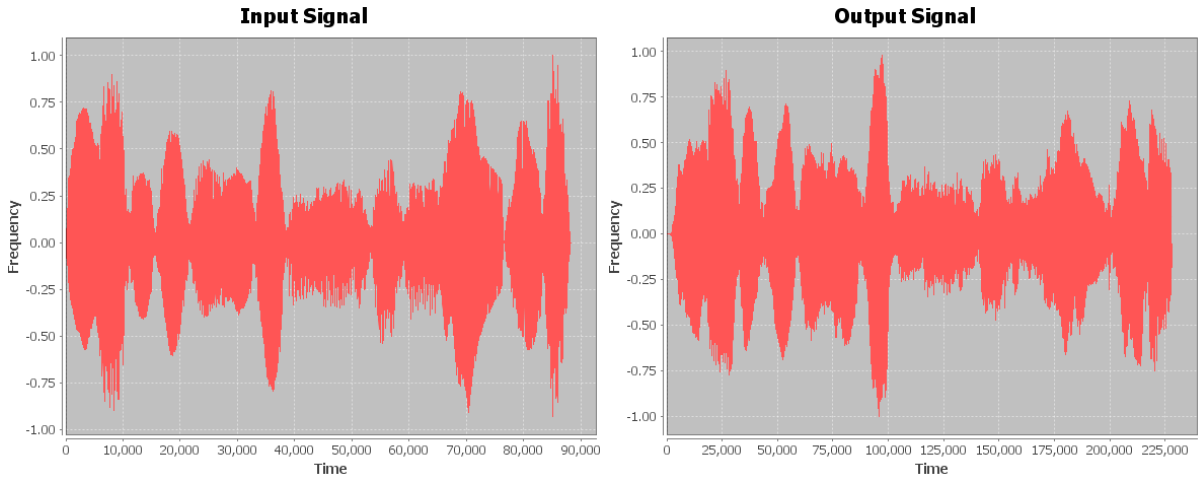

Figure 4.12 Input and the time stretched signals from PVocoder

In Figure 4.11, the upper graphs show a frame of input signal in different form; grain (input data after applying the windowing) and the magnitude and phase of the input signal after taking FFT function, while the lower graphs show a frame of the output signal in a similar manner; grain (output data) and the magnitude and phase of output signal after taking inverse FFT function. And in Figure 4.12, the input audio input signal is shown in the left graph and the right graph represents the final output signal which has been stretched in time.

In this section, we presented the detail of the implementation of the Phase Vocoder technique in Java. The accuracy and correctness of this implementation will be evaluated in chapter 6. The combination between the Lomb spectrogram and the Phase Vocoder leads us to be able to achieve the functional requirement of the application which is to synchronize the running speed of the user with the tempo of a song. In the next section we will describe how we implement these techniques on an Android smartphone/tablet and its real-time behaviour.

4.4 Implementation on the Android Platform

After we implemented the Lomb Periodogram and the Phase Vocoder in Java, we then created an external library and import it into a new Android project named "*SSURE_app*", in order to simplify the project and make it look tidy. In this project, we designed its structure using the concept of threading in Android applications, i.e. thread and thread handler (), to decrease the workload in the main process (UI Thread) and allow the program to use the Android Services to perform application tasks in the background.

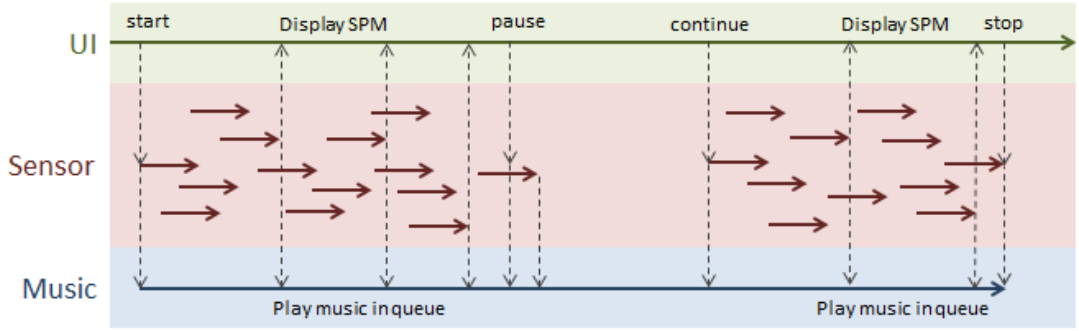

Figure 4.13 Input and the time stretched signals from PVocoder

When the app starts, the UI Thread will run and keep working until it exits. Once the start button in the app is pressed, the Music Thread will be run and play a song in a queue. In case there is no song added to the queue the default song or previous audio sound will be played until a selected file is added. This thread will begin processing when a sensor event changes and save the input data into a temporary database as a buffer until it is filled. Only then, will an SPM value be computed using the FastLomb class from the external library and the value displayed by sending message to the UI thread via its Handler. After SPM is analysed, a time-stretched audio file will be selected and added to a queue of song in Music thread. These processes will be run until the pause button or stop button is pressed. The different action between these two buttons is that pressing pause button will just stop the start of a new Sensor thread but Music thread will keep running without playing any sound, while pressing stop button will kill all threads. (Figure 4.13)

Ideally, the time-stretched audio file must be processed from the Phase Vocoder class in real-time. Unfortunately, this process takes long time although we tried to implement it to process the audio file in a short-time interval and work in parallel. It still takes approximately 20 seconds to process which is an unacceptable response for the purposes of our app. Due to the limited time for work we changed the idea for the implementation to process so that the time-stretched audio file will be carried out by the Phase Vocoder offline and stored within the list of audio files. Then, we can use the select song algorithms to select any of these pre-processed audio file, in which BPM is synchronized with input SPM, to play as it response to a change in the SPM. (Figure 4.14)

$$ratio = \frac{SPM}{BPM} \tag{4.1}$$

In the ideal process, the main purpose of the select song algorithms would be to select a new song with a different BPM when the ratio of SPM over BPM (Equation 4.1) is less than 0.9 or more than 1.1 (different more than 10%). However, in implemented process, we retain the idea of selecting a new song when the difference is more than 10% and select the time-stretched audio file

when the difference is within 10%. The list of songs and selected conditions will be shown in Table 4.9.

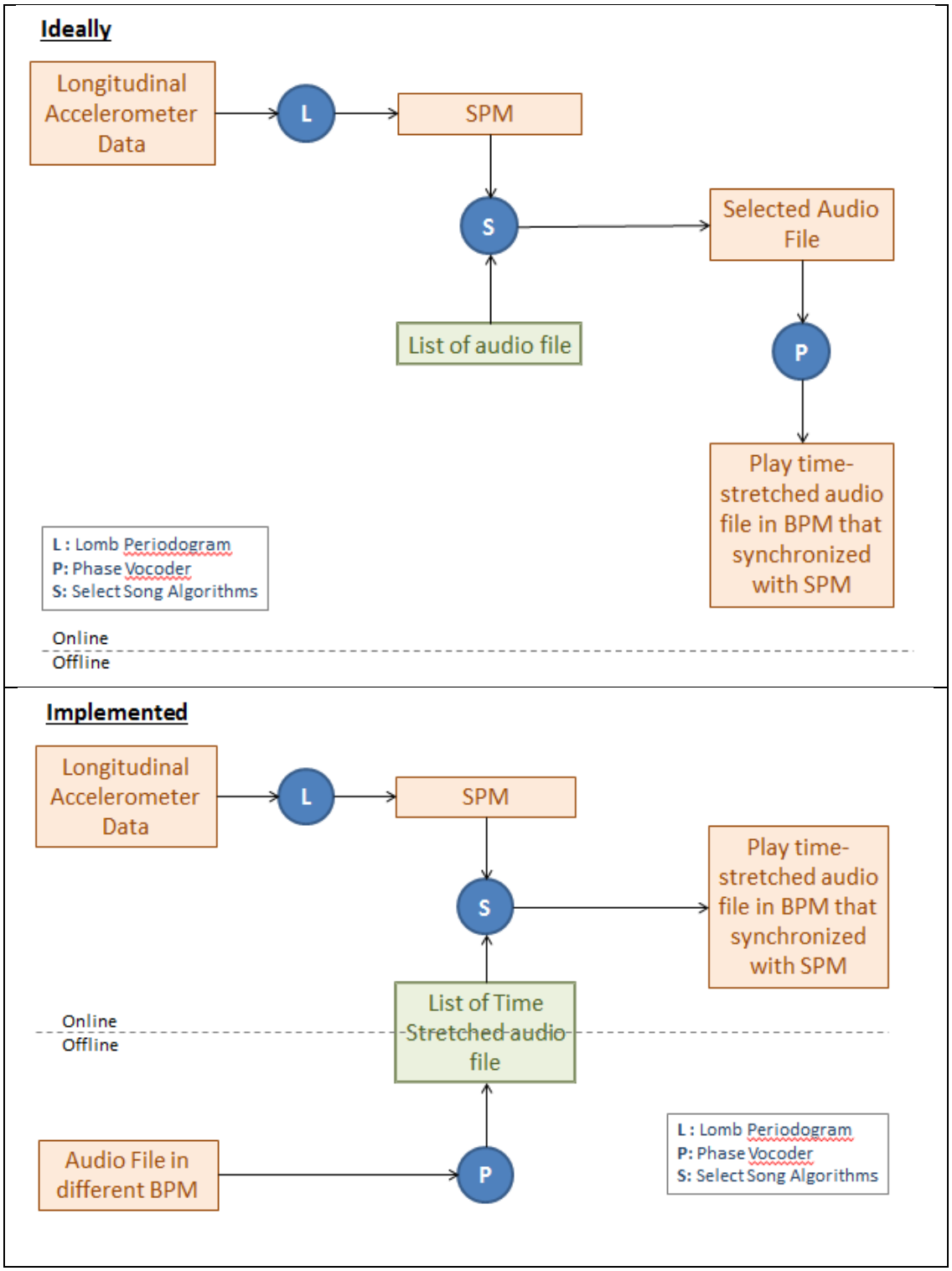

Figure 4.14 Ideal process and implemented process

The SPM value that we got from the Lomb Periodogram can be varying. Some values can be too low or too high in frequency to be considered as the result of human activity, so we consider those

values to be due to either noise from the device, the data collection method, or the way in which the subjects are running. This unsteadiness in the data leads us to implement a filter that can select data whose value is between 50 to 200 SPM to represent the cadence.

Song name	Select new song
Contemporary Music (60 BPM)	50 <= SPM < 70
Chillout Lounge (80 BPM)	70 <= SPM < 90
Bullseye (100 BPM)	90 <= SPM < 110
Deep House (120 BPM)	110 <= SPM < 130
Fitness Beat (140 BPM)	130 <= SPM < 150
Pump Up (160 BPM)	150 <= SPM < 170
Tequila (180 BPM)	170 <= SPM <200

Table 4.9 Song List and new song selecting condition

Time-Stretched Ratio	SPM/BPM ratio
0.9	< 0.9
0.95	0.9 – 0.95
Normal	0.95 – 1.05
1.05	1.05 – 1.1
1.1	> 1.1

Table 4.10 time-stretched audio file selecting condition

In the Android implementation, we attempt to synchronize the music with the cadence, so that we need to set up the parameters for the *frame size* and *hop size* for each frame of SPM processing to produce the clearest results corresponds to a time interval of approximately 3-4 seconds (as the average duration of audio file), we found that a *frame size* of 512 and *hop size* of 240 with *SENSOR_DELAY_GAME* mode (*James et al., 2011*), which will refresh the sensor event with frequency about 120 Hz, were best for this condition.

From this approach, the *SSURE_app* can run smoothly on the tablet with only a slightly delayed response in the changing between the set of time-stretched audio files. An example of the SPM data processing is shown in three colours; Red represents the unprocessed data, Yellow represents the outbound SPM values, and Green represents the cadence values. While the song is playing, its BPM and synchronized SPM value will be shown under the current SPM label. The screen-shot of SSURE_app is presented as in Figure 4.15.

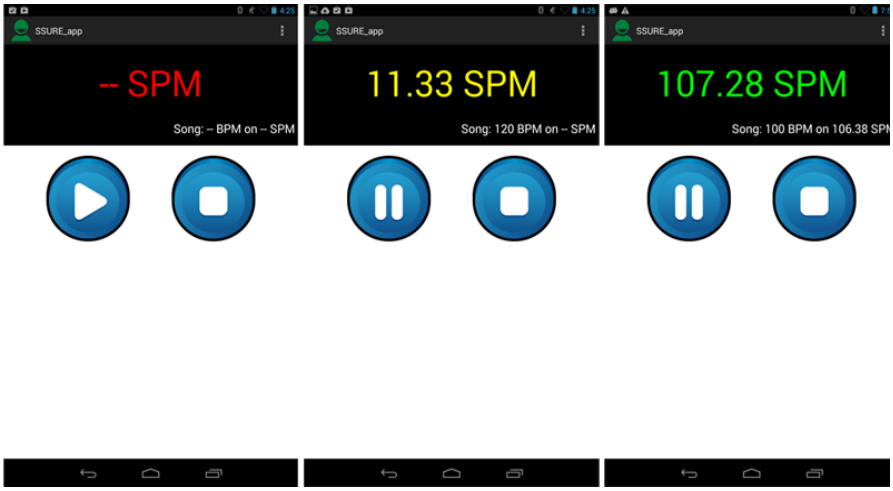

Figure 4.15 Screenshot in three different responses

4.5 Summary

In this chapter, we presented our solution to the functional requirements for our project. Each section described the actual algorithms we are using through a sequence of steps that lead us to get the desired result. However, we have not yet considered the performance and quality of the software. To acquire a good performance while ensuring that the program still works correctly, it requires various techniques and knowledge drawn from the field of software engineering. The solution for these non-functional requirements will be described in next Chapter.

5. Solution for Software Dependability

After we described the solution for analysing the running data in Chapter 4, we now consider the technique that we will use to successfully obtain software dependability. There are a variety of methodologies that we have presented in Chapter 3. In this chapter we will specify the knowledge required at each phase of development that can satisfy our project's non-functional requirements. First, we describe the development plan that we used in this project. This plan must suit our project's conditions in order to gain dependable software as our goal. Next, we present the prototype which displays the user interface and functionality for an initial assessment to motivate the design, and express the concepts behind them. Then, we state our test cases which must be verified to ensure the user's satisfaction. Lastly, we explain our implementation method and its result which can be used to assess the success of our project.

5.1 Development Plan

As we mentioned before usability, efficiency, and dependability are the requirements which were chosen to be the most important to ensure that our application will have a lasting positive effect on users. The technique that we will use to develop our application to obtain those requirements is the Agile Development Life Cycle (SDLC) (Cohen et al., 2004) which breaks the whole project tasks into small increments. The well-known Agile software development methods that we will use for this project is the technique known as SCRUM (Goldstein Ilan, 2013). It suits our project because its key principle is that it recognises that working with new techniques in a project of short and limited duration requires a flexible plan due to unpredictable challenges that might arise.

In general, the software development lifecycle begins with the gathering of user requirements. Luckily, we do not need to gather the user requirements by ourselves because this project is associated with the *Beathealth* project which is a collaborative research project between four different universities from EU countries: France, Belgium, Ireland, and Spain (BeatHealth, 2014). This project aims to help users with the power of beats and rhythm in new technology application that will help people to achieve better health. The *Beathealth* project has already had some development before our project has started. There was a meeting on the 29 October 2013 at which the user requirements were considered and gathered into the form of User stories. From these user stories, we can refer to some of their requirements as the user requirements for our project.

Using the SCRUM method, the development plan is organised as an iterative and incremental development (Brown et al., 2012). We began with designing the Product Backlog as a list of all

product features and specify the target effort (working days) for each increment by selecting the functional requirements which would be reasonable to develop in that interval from the user stories.

From the selected functional requirements, we can generate a product backlog breaks the tasks into 9 sprints with 7 releases, of which 4 of those are internal and the remainder are external releases. The internal releases are only aimed at confirming the operation of the software to the developer team, while those external releases are to deliver an application with certain functionalities to the end-user. In the product backlog, we specify an ID to represent the number of activities and each activity is defined with a target effort for each increment. Each sprint and release is represented in the table below with a blue and green background respectively.

ID	Description Increment #	1	2	3	4	5	6	7
	Effort needed for Release 1 as in the beginning of the sprint	135	113	98	83	53	40	25
1	Study about the language for Android development (Processing and Java)	10	0	0	0	0	0	0
2	Setting the development environment (Android SDK, Processing 2.1, Eclipse 4.3.1 and Add-on)	2	0	0	0	0	0	0
3	Study about the feature for Running Evaluation (Accelerometer Sensor and External Sensor)	5	0	0	0	0	0	0
4	Design User Interface and Function	3	0	0	0	0	0	0
5	Create the prototype of User Interface (using Processing)	2	0	0	0	0	0	0
Sprint 1	*Design User Interface and Function*							
Release 1	**Internal - draft of User Interface & display on tablet**							
6	Talk to sensor	9	9	0	0	0	0	0
7	Fetch data from sensor	5	5	0	0	0	0	0
8	Display raw data as a graph on tablet	1	1	0	0	0	0	0
Sprint 2	*Capture data from internal sensors*							
Release 2	**Internal - Show the captured data from internal sensor as a graph on tablet**							
9	Talk to sensor	9	9	9	0	0	0	0
10	Fetch data from sensor	5	5	5	0	0	0	0
11	Display raw data as a graph on tablet	1	1	1	0	0	0	0
Sprint 3	*Capture data from external sensors*							
Release 3	**Internal - Show the captured data from external sensor as a graph on tablet**							
12	Use signal analysis to work out the rate of running	15	15	15	15	0	0	0
13	Extract signal data to pace, speed and duration	5	5	5	5	0	0	0

ID	Description Increment #	1	2	3	4	5	6	7
	Effort needed for Release 1 as in the beginning of the sprint	135	113	98	83	53	40	25
14	Implement analyse method to tablet	10	10	10	10	0	0	0
Sprint 4	*Process data from sensors*							
Release 4	**Internal - Show the evaluated data on tablet**							
15	Generate interface for data display	10	10	10	10	10	0	0
16	Return data for statistics/performance display to the user	3	3	3	3	3	0	0
Sprint 5	*Running User Interface*							
Release 5	**App that works with the basic data**							
17	Fetch data from GPS	5	5	5	5	5	5	0
18	Include map support	5	5	5	5	5	5	0
19	Create the user interface to see route and tracking	5	5	5	5	5	5	0
Sprint 6	*GPS and navigation Integration*							
Release 6	**App that works with the current tracking on tablet**							
20	Capture the real-time data	5	5	5	5	5	5	5
21	Display and play music from playlist	5	5	5	5	5	5	5
22	Adjust cadence to the songs' tempo	15	15	15	15	15	15	15
Sprint 7	*Tempo Feature*							
Release 7	**App that works with biofeedback on tablet**							
	Effort in the whole backlog	135	113	98	83	53	40	25

Table 5.1 Product Backlog

Accompanying this is a Burndown chart that is used to represent the amount of work left to do over time. This was generated to predict when the work will be completed and, for our project, its purpose is to target the amount of work which can be done in the limited time frame. This chart can be used to monitor progress in each iteration by updating it at the end of each activity and comparing the estimate and actual performance. The Initial version of the Burndown chart linked with the product backlog is shown in the graph below.

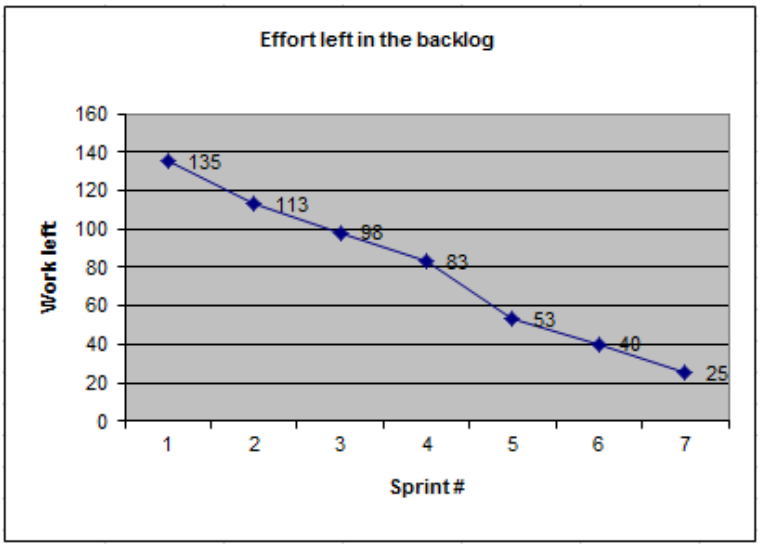

Figure 5.1 Burndown Chart

From the product backlog, we can distinguish between the backbone (Scrumsense, 2011) of features needed for the application and any optional features so that we can prioritize those activities that need to be done before others. Taking this approach, we generated a User Story Map which lays out the essential tasks or backbone of software activities on top and a breakdown into smaller subtasks. Then, we used a 'Traversing the map methodology' to build the application incrementally, story by story, choosing them from the story map left to right, and top to bottom. This technique will build up the application just one feature at a time. The project's user story map is presented as following. Boxes with a blue colour represent the backbone activities and the red coloured boxes are the smaller tasks.

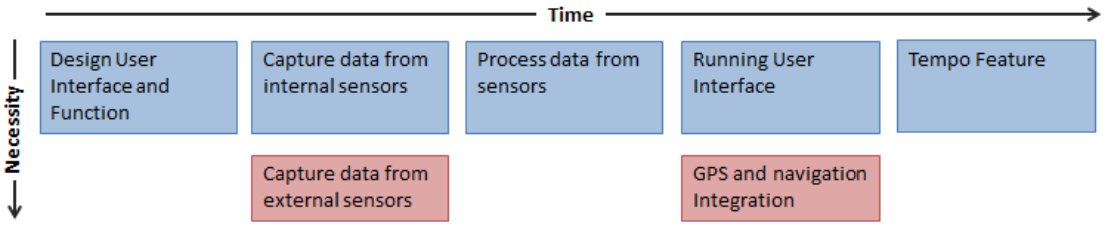

Figure 5.2 User Story Map

If the development follows a good plan, the developer team can control the conditions which might affect to the software quality, predict the problems that might happen during the development process and prepare the solutions to solve them. It also helps us to appraise the project size which correlates with the project schedule. In next section, we will describe the detail of work in the first activity, that is, the design of the User Interface and its functions.

5.2 Prototype

In order to achieve the usability requirements, the first activity that we did after considering the development platform and language was to set up the environment to design the user interface and its functions, and then create a prototype for initial assessment to motivate the design. This prototype can bring about a commitment regarding the product interface and its functionality between the users and developers. As we mentioned before, we gathered the functional requirements from the *Beathealth* project's user stories. Thus, we could ask the project development team to consider our prototype to determine whether it is satisfactory. This prototype is created as a throwaway prototype (Brown et al., 2012); because it would be used only to find the agreement on the design and we would not like to spend too much time developing it.

Figure 5.3 Processing language logo [5]

The Processing Language (Casey et al., 2007), is a programming language and development environment that debuted in 2001, was introduced to build our prototype due to its convenience and its easy-to-understand syntax. It also supports Android devices and has been used to promoted software literacy within the visual arts and literacy within technology. In this phase, we are not concerned about the analysis process. Consequently, the UI is simply designed and does not require implementing complicated algorithms using a powerful language such as Java for Android. Thus, the processing language is most suitable to be used to develop this throwaway prototype.

Due to the fact that the design of interface in prototype should be simple, we decided not to add the 'motion' (Ruiz et al., 2011) which makes the app look smooth and is a sign of interaction. However, an attractive design and appropriate position of menus and buttons are required all the same as they provide the functionalities. After the design is completed, the prototype app is developed using the Processing language in Android mode and presented to our customers, that is, the *Bealheath* development teams. The interface of each screen in the prototype app is shown in Figure 5.4a -5.4c as follows.

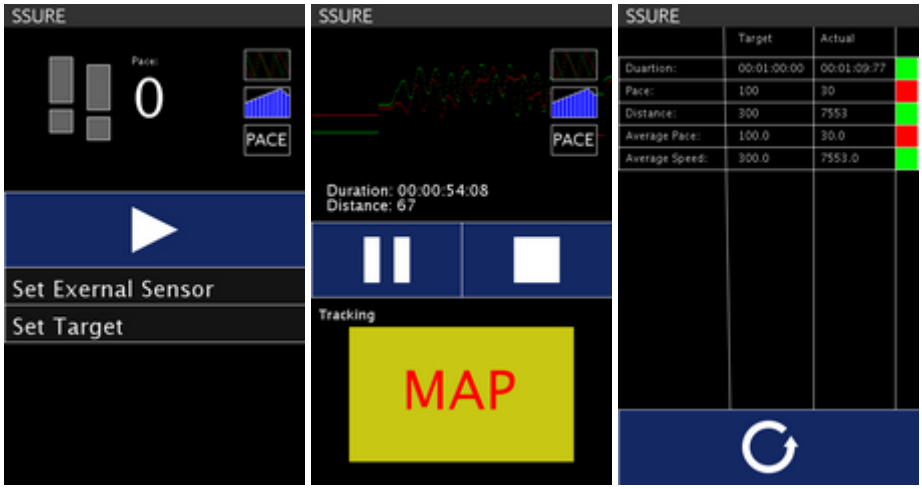

Figure 5.4a, 5.4b, 5.4c Prototype display

From this design of prototype, we will describe the reasoning behind these user interface design and functionality. First of all, we discuss about the start page design (Figure 5.4a). In this page, there is a menu allowing user to set up their target. This target can be duration, distance, average cadence, or how to respond to the music at a specific tempo. This target value will be compared with the actual running information and this will then be displayed in the evaluation page (Figure 5.4c). This menu also allows the user to set an external sensor which might be connected via Bluetooth or ANT+. It is an optional activity that it will not be included to our project due to the limited time available.

On top of the screen, the display mode is shown and can be used to select a desired plotting mode, either a line graph or histogram that displays the pace or cadence. This display will also be shown in the response page (Figure 5.4b) and it must be large and clear enough for the user to be able to see the result while they are running. A big play button is laid in the centre of screen because it leads to the main function of this app. This button needs to be attractive and easy for user to understand how to use the app without training. We use a sideways triangle sign which is generally known as a play function (Figure 5.5).

In the response screen, we keep the display the same as in the start page, but here the running information will be displayed as set in the selected mode. This mode still can be changed by selecting the icons on the right-hand side. This allows the user to see different information without stopping running. Other information that will be displayed in response page is the time taken and the distance travelled. Showing such information allows users to know how long and how far they ran so they can estimate themselves about controlling their speed or to stop running. Two big buttons, which are pause and stop, are laid in the centre of screen for the same reason of play button in start page. We also use a general sign to represent the functionality of each button; two vertical lines sign represents the pause function, and one square represents the stop function (Figure 5.5). The

difference between these two functions is that the pause function can stop analysing data and continue after pressing the play sign again while the stop function will stop the analysis and go to evaluation page which shows the overall result compared with the target. At the bottom of this screen, the map and tracking route will be shown for users to know where they were travelled.

Figure 5.5 General Play, Pause, and Stop signs

The last page is an evaluation page (Figure 5.4c) which shows the actual running information and compares it with the target. This comparison will be summarized and displayed by different colours; green to show that it meets the target, and red to show that it has not. At the bottom of screen, a big button is presented to reset and start the program from the beginning again. This function is represented with a circular arrow and it is the only one button on this screen so it should be easy to understand its meaning. (Figure 5.4c)

This user interface design and functionality was agreed with the *Beathealth* team. Thus, our final app was to be built based on this design to keep the usability and user's satisfaction. Users have played the most important role is this phase. Therefore, assessing the initial design with users should be done at the first phase of project in order keep our development focused and try to prevent any major changes in the design and functionality later on that users may not be happy with if they haven't evaluated them beforehand.

5.3 Test Cases

In our application, we focus on how it responds with an audio track that should be synchronized with the user's movement, so that we define test cases to ensure that program can select the audio of which its BPM has been adjusted to synchronize with input SPM. We created test cases for Unit testing to test the song selection algorithms, using the Black-Box - Boundary Value Analysis technique, and verifying the code coverage by JML on eclipse. (Brown et al., 2013)

For this algorithm, we choose Boundary Value Analysis technique to generate test cases because the specification is defined to return the different output in different input partitions and there is more likelihood of having an error at the boundary of each partition. We derive expected output values from the specification for this song selection algorithm that we have explained in the previous chapter in the implementation on an Android device section. Firstly when we are creating

the tests, we need some analysis to identify the specification-based ranges for the parameters SPM and BPM in our algorithm, then we create test cases and test data as in the following tables. (Table 5.2 and 5.3). Note that nextAfter is the value before the border and nextUp is the value after the border.

Test Analyse- Parameter SPM

Range: [Double.MIN_VALUE..nextAfter]
[50..nextAfter]
[60*0.9..nextAfter][60*0.95..nextAfter][60*1.05..nextAfter][60*1.1..nextAfter]
[70..nextAfter]
[80*0.9..nextAfter][80*0.95..nextAfter][80*1.05..nextAfter][80*1.1..nextAfter]
[90..nextAfter]
[100*0.95..nextAfter][100*1.05..nextAfter]
[110..nextAfter]
[120*0.95..nextAfter][120*1.05..nextAfter]
[130..nextAfter]
[140*0.95..nextAfter][140*1.05..nextAfter]
[150..nextAfter]
[160*0.95..nextAfter][160*1.05..nextAfter]
[170..nextAfter]
[180*0.95..nextAfter][180*1.05..nextAfter][180*1.1..200]
[nextUp..Double.MAX_VALUE]

Test Cases

Case	Parameter	Boundary Value	Test
Inputs:			
1*	SPM	Double.MIN_VALUE	1
2*		(50).nextAfter	2
3		50	3
4		(60*0.9).nextAfter	4
5		60*0.9	5
6		(60*0.95).nextAfter	6
7		60*0.95	7
8		(60*1.05).nextAfter	8
9		60*1.05	9
10		(60*1.1).nextAfter	10
11		60*1.1	11
12		(70).nextAfter	12
13		70	13
14		(80*0.9).nextAfter	14

Case	Parameter	Boundary Value	Test
15		80*0.9	15
16		(80*0.95).nextAfter	16
17		80*0.95	17
18		(80*1.05).nextAfter	18
19		80*1.05	19
20		(80*1.1).nextAfter	20
21		80*1.1	21
22		(90).nextAfter	22
23		90	23
24		(100*0.95).nextAfter	24
25		100*0.95	25
26		(100*1.05).nextAfter	26
27		100*1.05	27
28		(110).nextAfter	28
29		110	29
30		(120*0.95).nextAfter	30
31		120*0.95	31
32		(120*1.05).nextAfter	32
33		120*1.05	33
34		(130).nextAfter	34
35		130	35
36		(140*0.95).nextAfter	36
37		140*0.95	37
38		(140*1.05).nextAfter	38
39		140*1.05	39
40		(150).nextAfter	40
41		150	41
42		(160*0.95).nextAfter	42
43		160*0.95	43
44		(160*1.05).nextAfter	44
45		160*1.05	45
46		(170).nextAfter	46
47		170	47
48		(180*0.95).nextAfter	48
49		180*0.95	49
50		(180*1.05).nextAfter	50

Case	Parameter	Boundary Value	Test
51		180*1.05	51
52		(180*1.1).nextAfter	52
53		180*1.1	53
54		200	54
55		(200).nextUp	55
56		Double.MIN_VALUE	56
Outputs:			
57*	Return Value	null	1
58		BPM60_090	3
59		BPM60_095	5
60		BPM60_100	7
61		BPM60_105	9
62		BPM60_110	11
63		BPM80_090	13
64		BPM80_095	15
65		BPM80_100	17
66		BPM80_105	19
67		BPM80_110	21
68		BPM100_090	23
69		BPM100_095	24
70		BPM100_100	25
71		BPM100_105	27
72*		BPM100_110	N/A
73*		BPM120_090	N/A
74		BPM120_095	29
75		BPM120_100	31
76		BPM120_105	33
77*		BPM120_110	N/A
78*		BPM140_090	N/A
79		BPM140_095	35
80		BPM140_100	37
81		BPM140_105	39
82*		BPM140_110	N/A
83*		BPM160_090	N/A
84		BPM160_095	41
85		BPM160_100	43

Case	Parameter	Boundary Value	Test
86		BPM160_105	45
87*		BPM160_110	N/A
88*		BPM180_090	N/A
89		BPM180_095	47
90		BPM180_100	49
91		BPM180_105	51
92		BPM180_110	53

Table 5.2 Test Cases for the Selection Song algorithm

Test Data

ID	Test Cases Covered	Inputs	Expected Output
		SPM	return value
1*	1, 57	Double.MIN_VALUE	null
2*	2	(50).nextAfter	null
3	3,58	50	BPM60_090
4	4	(60*0.9).nextAfter	BPM60_090
5	5,59	60*0.9	BPM60_095
6	6	(60*0.95).nextAfter	BPM60_095
7	7,60	60*0.95	BPM60_100
8	8	(60*1.05).nextAfter	BPM60_100
9	9,61	60*1.05	BPM60_105
10	10	(60*1.1).nextAfter	BPM60_105
11	11,62	60*1.1	BPM60_110
12	12	(70).nextAfter	BPM60_110
13	13,63	70	BPM80_090
14	14	(80*0.9).nextAfter	BPM80_090
15	15,64	80*0.9	BPM80_095
16	16	(80*0.95).nextAfter	BPM80_095
17	17,65	80*0.95	BPM80_100
18	18	(80*1.05).nextAfter	BPM80_100
19	19,66	80*1.05	BPM80_105
20	20	(80*1.1).nextAfter	BPM80_105
21	21,67	80*1.1	BPM80_110
22	22	(90).nextAfter	BPM80_110
23	23,68	90	BPM100_090

ID	Test Cases Covered	Inputs	Expected Output
		SPM	return value
24	24,69	(100*0.95).nextAfter	BPM100_095
25	25,70	100*0.95	BPM100_100
26	26	(100*1.05).nextAfter	BPM100_100
27	27,71	100*1.05	BPM100_105
28	28	(110).nextAfter	BPM100_105
29	29,74	110	BPM120_095
30	30	(120*0.95).nextAfter	BPM120_095
31	31,75	120*0.95	BPM120_100
32	32	(120*1.05).nextAfter	BPM120_100
33	33,76	120*1.05	BPM120_105
34	34	(130).nextAfter	BPM120_105
35	35,79	130	BPM140_095
36	36	(140*0.95).nextAfter	BPM140_095
37	37,80	140*0.95	BPM140_100
38	38	(140*1.05).nextAfter	BPM140_100
39	39,81	140*1.05	BPM140_105
40	40	(150).nextAfter	BPM140_105
41	41,84	150	BPM160_095
42	42	(160*0.95).nextAfter	BPM160_095
43	43,85	160*0.95	BPM160_100
44	44	(160*1.05).nextAfter	BPM160_100
45	45,86	160*1.05	BPM160_105
46	46	(170).nextAfter	BPM160_105
47	47,89	170	BPM180_095
48	48	(180*0.95).nextAfter	BPM180_095
49	49,90	180*0.95	BPM180_100
50	50	(180*1.05).nextAfter	BPM180_100
51	51,91	180*1.05	BPM180_105
52	52	(180*1.1).nextAfter	BPM180_105
53	53,92	180*1.1	BPM180_110
54	54	200	BPM180_110
55*	55	(200).nextUp	null
56*	56	Double.MIN_VALUE	null

Table 5.2 Test Data for the Selection Song algorithm

After we define the test data, it will be implemented in a JML file so we can determine the percentage of code covered. We found that all test results are passed but the code covered percentage is 90.2%. This is not 100% as there is the output data that will never be returned, for example, *BPM120_090*. (Figure 5.6) This result is acceptable for our project and satisfies the specification for this algorithm.

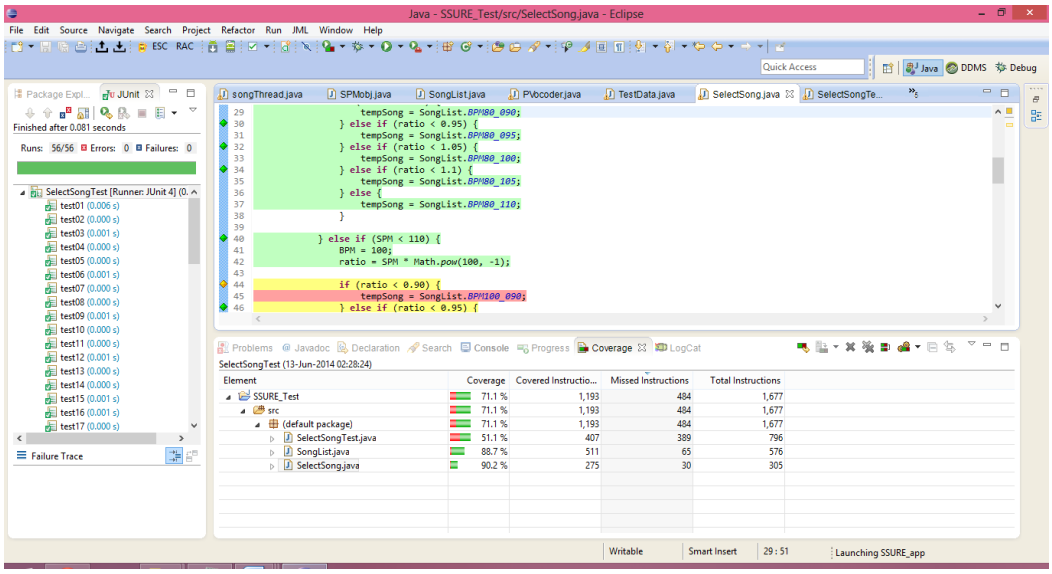

Figure 5.6 JML for Unit Testing and Code Coverage

5.4 Implemented Conditions

After planning for the implementation for each sprint in the product backlog, we found some obstacles, and consequently we could not implement the optional tasks in time. Thus, our first version of program will provide only the main function of the activities in backbone, excluding the GPS and external sensor. However, the program can achieve the main functionality from user's requirement and software quality. Furthermore, as we said in previous chapter that the implementation of the Phase Vocoder on Android platform process is computationally demanding and thus takes too much time to process, we have changed our approach to process the audio sound beforehand so as to fix this problem. By the way, it diminishes the accuracy of the BPM of the responding audio that should be exactly synchronized with SPM analysed from sensor data, but at a slightly different rate it should be too small to be perceptually noticeable.

Due to some changes in the implementation process, the user interface also changed from our first draft. To improve the performance, we removed graphics from the user interface, such as the line graph. The optional functions, like map and external sensor, are also gone as its functions cannot be made to work together with the audio playback at this time. Another difference between our plan

and the implemented program is that we do not have the algorithm to find the pace, so that every function dealing with pace information will be suspended.

Fortunately, using SCRUM to plan our project at the first instance allows us to adapt the plan and select the most important tasks to develop which suits the current situation without making a big impact on our project overall. The quality of software is still in our priority so that we can make sure that our application can work and never let the user down.

5.5 Summary

We have presented the methods and solutions that we used to obtain the non-functional requirements for our project. Each section described the different techniques that we used in the different phases of the development to ensure that the performance and quality of software is the concern of priority throughout the development process. The outcome after we applied this knowledge and methods to our project will be evaluated using another technique named software metrics. In the next chapter, we will present the evaluation results and compare the software quality of our project with the related work to conclude and answer our research question whether software engineering techniques can improve the quality of health awareness application on a mobile phone.

6. Evaluation

In this chapter, we first present the experimental set of metrics for our implementation in which we evaluate the error rate over our solution to make sure that our results are accurate. Then, we present other metrics with which we evaluate the quality of software in terms of the standard and non-functional requirements of our health awareness application. This metrics will be measured with both subjective and objective data. Finally, we will conclude by comparing and contrasting our project with other similar ones.

6.1 Java Implemented Method

When new methods are implemented, an evaluation of the accuracy of those methods is important. It is an indicator of the overall quality of the software. If one piece of the software is not accurate or robust, it can lead to the reliability of the results returned by the application being untrustworthy. Therefore, we need to evaluate every method that we implemented in Java.

First of all, we assess the accuracy of the Java-implemented Lomb Periodogram method by comparing it to the output from the Matlab version that we obtained from the Mathworks website. Our metrics are $\overline{\%Error_M}$ and $\overline{\%Error_F}$, defined as the average of the ratio of the difference between the outputs of the Java and the MatLab versions, to the reference MatLab version, when N is the number of outputs. We consider two outputs which are Magnitude and Frequency. These are denoted in the Java version as M_j for the Magnitude value and F_j for the Frequency value, and in the MatLab version M_m and F_m are the respective quantities.

$$\overline{\%Error_M} = \frac{\sum_N \frac{|M_j - M_m|}{M_m}}{N} \times 100 \tag{6.1}$$

$$\overline{\%Error_F} = \frac{\sum_N \frac{|F_j - F_m|}{F_m}}{N} \times 100 \tag{6.2}$$

Then, we apply these equations to the results of the Lomb Periodogram method using five sets of specific data acquired from the case study data we presented earlier in Chapter 4. The results are shown in the following Table 6.1.

Output	%Error
Magnitude	0.8635
Frequency	0.6574

Table 6.1 %*Error* from Java source code compared with the MatLab version

It can be seen that the average of both $\overline{\%Error_M}$ and $\overline{\%Error_F}$ are not over 1%. From our own observation we saw that this error is a consequence of the differences number rounding occurs in the MatLab version unlike the Java version. However, these errors do not have a noticeable effect on our final result because we are interested in the magnitude of the peak in the Lomb Periodogram only. A small difference between the Java and the Matlab version will have no noticeable perceptual effect when applied to altering the tempo of the music for the feedback, as a small difference in tempo will be too small for a human to notice.

Secondly, we will consider the response time when we find the *frame size* and *hop size* from the short-time transformation process in the Android implementation. As we know that due to the time required to collect the sampled accelerometer data into the buffer, the value of *frame size* and *hop size,* which are processed in the short-time Lomb Periodogram, will determine a response time, that is the time taken before the result can be computed and displayed. We separate this response time into two types; the first response time is when the application starts and the second is the interval time while the application is running. The different between these two types is that the response time when the application starts is the waiting time to fill the first amount of data in the buffer of length *frame size*, while the response time while the application is running is the waiting time to fill the buffer for next *hop size* of data. These interval times should conform to the duration of audio file when it plays and maintain the feeling of interaction with the users.

To find appropriate values for those two parameters, we set up our program to monitor the time when each frame of the Lomb Periodogram starts and finishes. The result will be reported in a Log file that we can track after the program is run. 15 different sets of values were established and statically run through the program. For each set of values, the log information from about 20 frames was recorded to find the average time interval. The results are shown as Table 6.2 below.

Frame size	Hop size	Start time (s)	Interval time (s)
256	30	6.8	1.24
256	60	7.9	1.85
256	120	7.9	3.01
256	180	8.6	4.43
512	30	14.1	1.42
512	60	12.2	2.67

Frame size	Hop size	Start time (s)	Interval time (s)
512	120	16	3.83
512	180	13.3	6.55
512	240	16.1	6.42
1024	30	23.6	1.15
1024	60	24.5	1.24
1024	120	23.7	3.51
1024	180	25.4	4.28
1024	240	30.5	5.23
1024	300	29.3	7.58

Table 6.2 The results of respond time for different values of *frame size* and *hop size*

The average duration of songs of different BPM that we used was 3.58 seconds (the duration of songs are presented in Chapter 4 Section 4). Thus, the best results from our experiment are for a *frame size* 256 and a *hop size* 120 (3.01 seconds), a *frame size* 512 and a *hop size* 120 (3.83 seconds), and a *frame size* 1024 and a *hop size* 120 (3.51 seconds). We know that the more data we process, the more accurate the result is. However, the start time for a *frame size* of 1024 is too slow and might diminish the user feeling of interaction. Therefore, a *frame size* 512 and a *hop size* 120, which has an acceptable start time of 16 seconds, are the best values for our implementation.

In this section, we evaluated the accuracy of the algorithms used in our application. In the next two sections we will describe the metrics to measure the quality of our application in terms of both objective and subject metrics respectively.

6.2 Objective Software Metrics

Measuring the quality of software is an important part of the process to evaluate the potential success of product. Objective software metrics are measurements that can be represented as a number that can be compared to evaluate the quality. In this section, we will describe about defining objective metrics to measure the accuracy and efficiency in our application.

6.2.1 Accuracy in Measuring Running Information

Health awareness applications display information and its accuracy is critical as discussed in the introductory chapter. Thus, the accuracy of result is very important to our project. The SPM value is analysed and processed directly from the data captured from the user's movement. Thus, we need

to find metrics to measure the accuracy of the result from the Lomb Periodogram algorithm that we have implemented.

We set up this experiment where we asked the subject to run for one minute while wearing a backpack with device to collect the data. Their actual steps were counted. Then, we analysed the captured data with the Lomb Periodgram class. The results are shown in the Table 6.3 below.

Actual (Steps Per Minute)	Result from the Lomb Periodogram (Steps Per Minute)	%Error Difference
158	160	1.26
120	128	6.67
68	79	16.17

Table 6.3 The comparison between the number of actual steps and the result from the Lomb Periodogram, and the Error difference

From the results, we calculated the error difference between the measured and counted SPM and found that the average value of the Error difference was about 8%. We note that the error rate increased for lower speeds of running. This is possibly due to the fact that slow movement introduces more noise to the system. Overall, the results are accurate within a 10% error margin.

6.2.2 Efficiency in Movement-Music Synchronization

Efficiency of the software is one of the important qualities that a health awareness application should have. For measuring the efficiency of a product, the main issue is that of response accuracy and getting the response within a significant duration. In our project, we focus on the accuracy of the synchronization of user's movement (SPM) with the Music (BPM) and its response time interval. For this measurement, we define the metrics to monitor the application's response in terms of tis processing of the songs within our application to a different series of inputs. The result can be plotted as a graph representing the SPM input, the output BPM of song and the signal waveform graph of the audio sound with respect to the input time. Here, a series of SPM input values is defined in different four formats: an stepping increase in the SPM, a stepping decrease in the SPM, a gradually increase in the SPM, and a gradually decrease in the SPM. The results of these series are shown in the following figures.

- Step Increase and Decrease in the SPM series – The results in Figure 6.1 show that the BPM of the input sound responds to the stepped increase in SPM as the BPM rises synchronously with it. In a similar way, the stepped decrease sees a relative decrease in the BPM, as shown in Figure 6.2. Meanwhile, there are some delays appearing in between the SPM and BPM in both figures resulting

from the interval time mismatch because of the computation of the Lomb Periodogram for each frame.

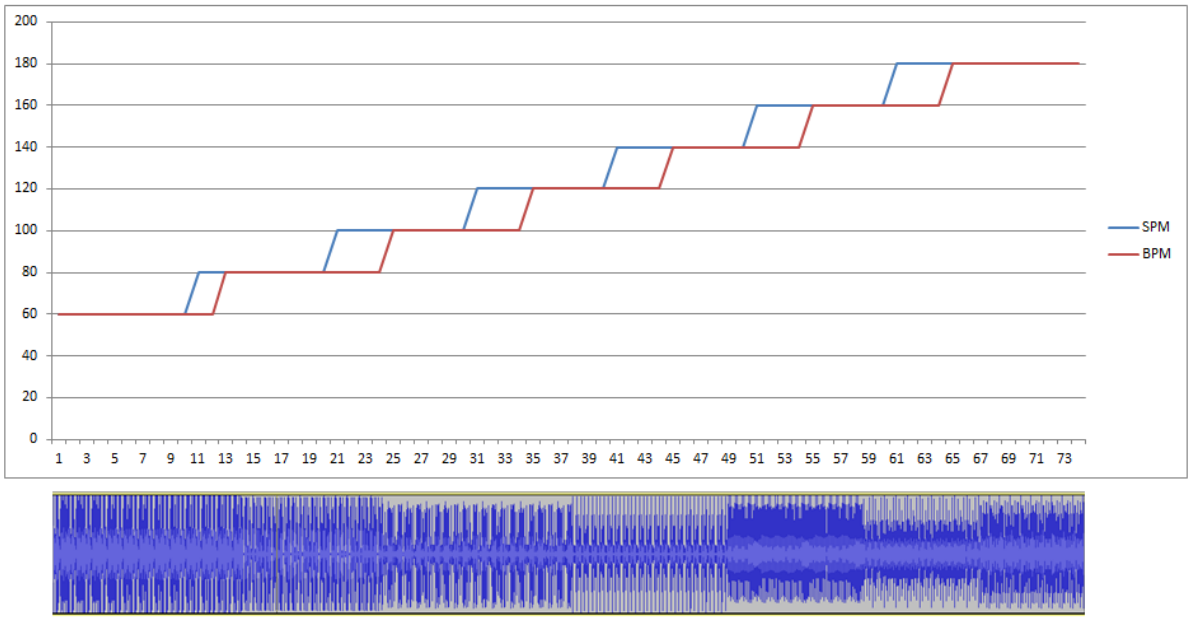

Figure 6.1 The graph result for increase in step series of input format

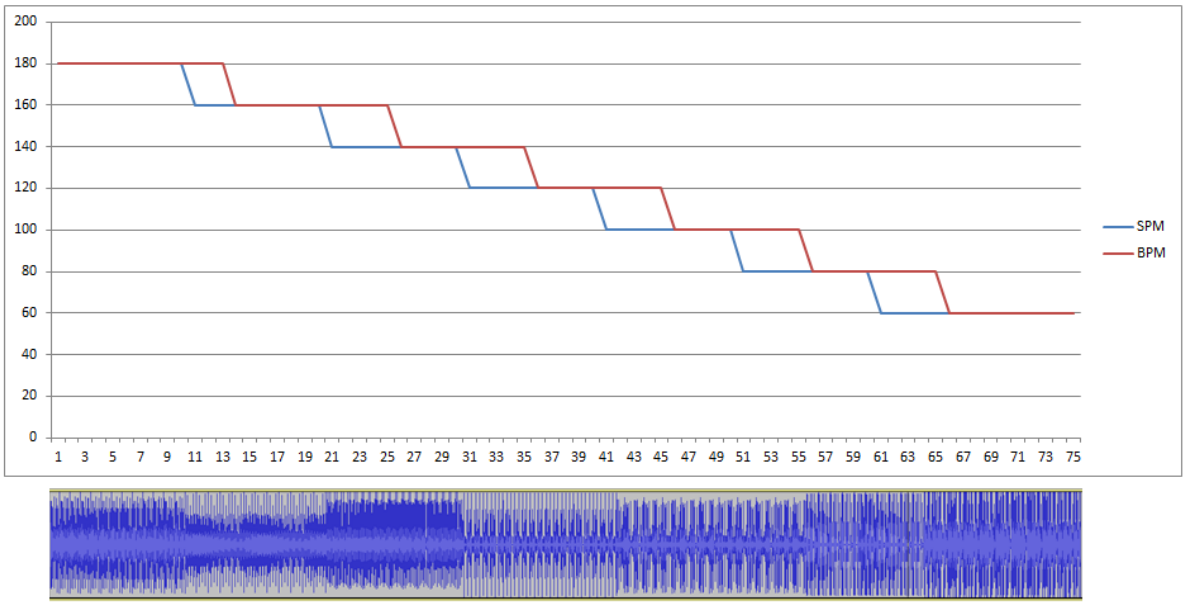

Figure 6.2 The graph result for decrease in step series of input format

- Gradually increase and Gradually decrease the SPM – The results shows that the trend of the BPM of the sound rises synchronously with a gradual increase in the input SPM (Figure 6.3), while the trend from a gradually decrease in the SPM sees a corresponding drop in the BPM (Figure 6.4). However, they are slight fluctuations around the SPM value. In addition, in both sets of their results there are still some delays that appear which result from the different duration between the playing

sound and the interval time of the Lomb Periodogram in each frame. This is the same as for the previous experiments.

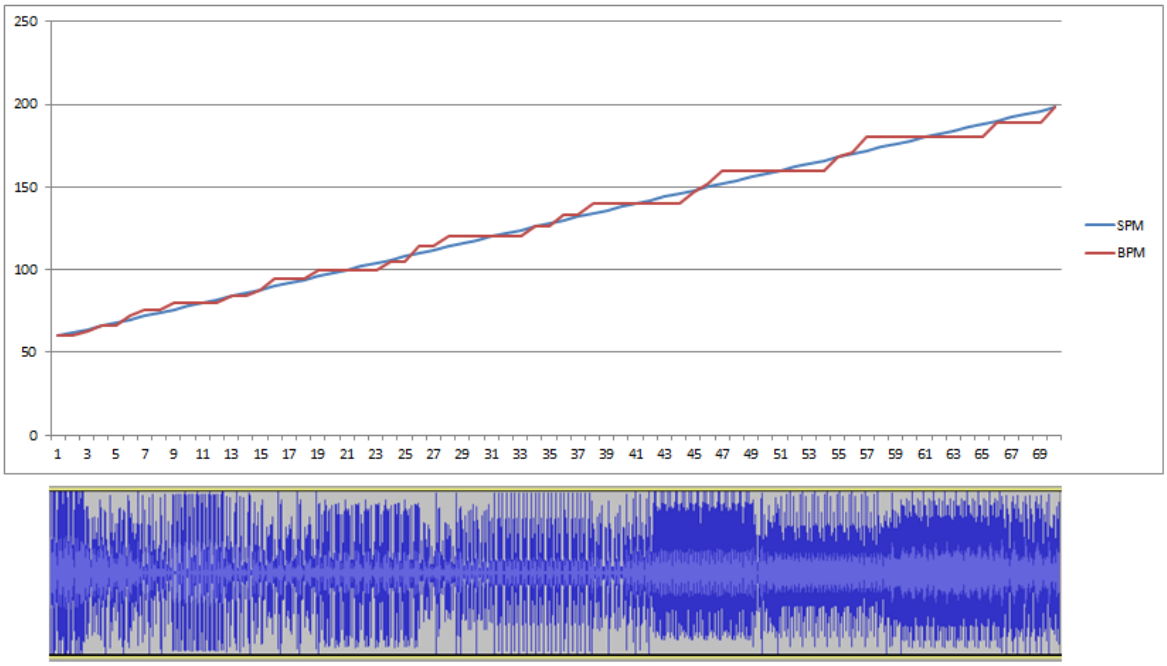

Figure 6.3 The graph result for gradually increase series of input format

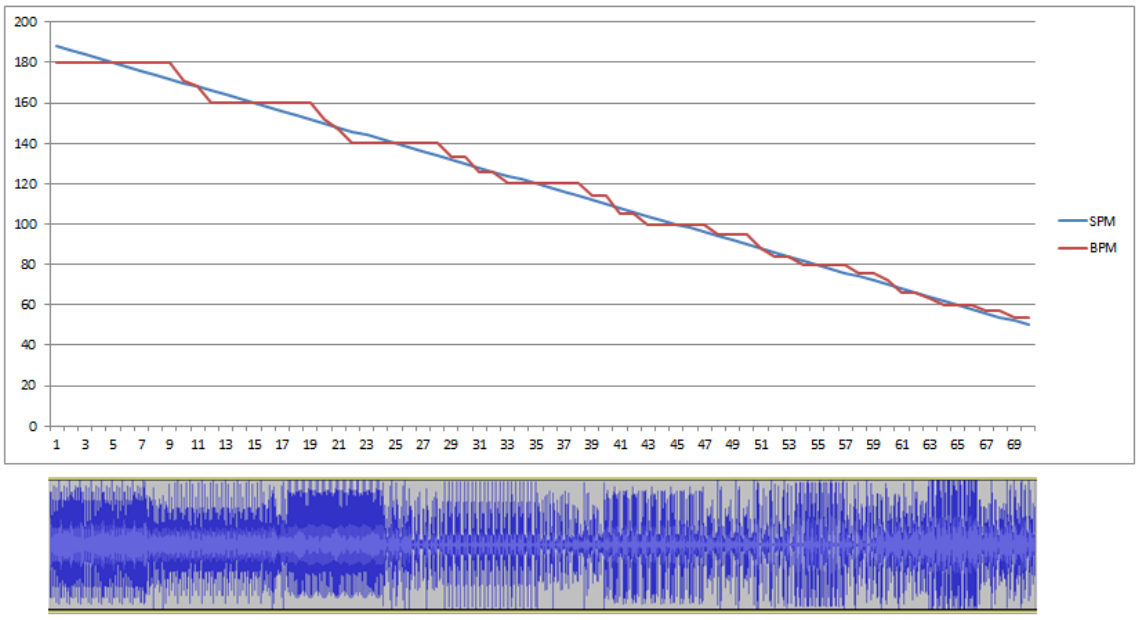

Figure 6.4 The graph result for gradually decrease series of input format

These results show that the synchronization of user's movement (SPM) and Music (BPM) and its response time interval is accurate. Although there are some delays but its trend still responds in the appropriate way. This leads to the conclusion that the application is efficient.

6.3 Subjective Software Metrics

Beside the Objective metrics that we have presented in the previous section, Subjective metrics, which attempt to track less quantifiable data such as quality attributes, have also been used considerably to measure the quality of software. Data for a subjective metrics would often be collected through interviews or surveys and summarized by analysing the response data. In this section, we will describe about defining the metrics to measure the usability and dependability in our application.

6.3.1 Usability

The usability is important in mobile applications, especially on the Android platform as it allows users to download an application easily. It requires those applications to perform in zero-training conditions where users consider using it from their experience only. In order to measure the usability of our application, we developed a survey with 9 questions to get user's feedback after using the application. The questions were based on the quality characteristics which were usability, effectiveness and efficiency. The answers could be ranked over 5 different scores from 1 to 5 and the responder will choose one of those scores for each question. The detail of each question in the survey is presented in the following Table 6.4.

Quality Characteristic	Goal	Guideline	Question
Usability	Attractive	User Interface	Are users happy with interface?
	Simplicity	Ease to learn	Are users familiar with the user interface?
			How easy to understand the application for the first time?
		Ease to use	How easy to use functions in the application?
	Problem detection	Find the unique problem	List the problems that you found while using the application.
Effectiveness	Accuracy	Accurate	Is the application accurate?
		Successful	Does the response sound motivate your running activities?
Availability	Time Taken	Response	How much time is taken by application to respond?
	Features	Touch screen facilities	Does the application provide appropriate menu buttons?

Table 6.4 The graph result for gradually decrease series of input format

There are 4 responders who gave us a feedback on our application by this survey. Although we had a small sample size, it was still possible to carry out a usability test. As a conclusion from an important paper (Turner et al., 2006), the authors said that most usability problems are detected with the first three to five subjects. The average scores that users responded with for each question and the list of problems are shown in Table 6.5 below.

Quality Characteristic	Question	Answer
Usability	Are users happy with the interface?	4.25
Usability	Are users familiar with the user interface?	5
Usability	How easy to understand the application for the first time?	3.5
Usability	How easy to use the functions in the application?	4
Effectiveness	Is the application accurate?	3.75
Effectiveness	Does the response sound motivate your running activities?	4.5
Availability	How much time taken by the application to respond?	4.25
Availability	Does the application provide appropriate menu buttons?	4.75
Usability	List the problem that you found while using the application.	1. Need to provide Help Menu (2) 2. Delayed response (2)

Table 6.4 Summary of the responses to the survey

From the responses, we considered how the average score for the questions in each category represented each quality attribute. Here, when we look at the usability of the application the average score is 4.3125, which we consider as a good response. We note that users' feels satisfied with the user interface and our application is user friendly. Another question relating to the usability characteristic asks users to list the problems that they found while using the application. We actually calculated the possibility of finding a unique problem in the application based on the evaluation by the small sample size as was recommended in (Turner et al., 2006). The calculation will be shown as follows in Table 6.5.

Problem Number				
Subject	**1**	**2**	**Count**	**P**
1	1	0	1	0.500
2	0	1	1	0.500
3	1	0	1	0.500
4	0	1	1	0.500
Count	2	2		
P	0.500	0.500		0.500

Table 6.5 Data from a Survey Usability; Four Subjects, p_{est} = 0.500

Applying the Good-Turing estimating procedure

$$P_{GT-adj} = \frac{0.500}{\left(1+\frac{1}{2}\right)} = 0.333 \quad (6.1)$$

Applying normalization

$$P_{Norm-adj} = \left(0.500 - \frac{1}{4}\right)\left(1 - \frac{1}{4}\right) = 0.188 \quad (6.2)$$

The average of the two estimates is

$$P_{adj} = \frac{1}{2}(0.333 + 0.188) = 0.2605 \quad (6.3)$$

Given p = 0.2605, the estimated proportion of discovered problems would be $1 - (1 - 0.2605)^4$, or 0.701. This number is quite high and represents that there could be some unique problem in the usability. Therefore, it would need to be fixed in some future development. Moreover, in the survey, there are some questions concerning about effectiveness and availability quality characteristics that also must be evaluated. These will be presented in next section.

6.3.2 Dependability

Another aspect of software quality that is important to health awareness applications is dependability. We have mentioned in an earlier chapter that dependability is an integrating concept that consists of several attributes. To measure this dependability, we need to measure those attributes. However, in this evaluation we will focus just on two attributes which are safety and availability.

According to the goals of the questions in the survey, the accuracy questions related to the effectiveness quality characteristic can be interchanged to represent a metric that measures the safety attribute. We also have metrics whose goal is to measure the availability attribute which are given by the time taken and features questions. From the given responses in Table 6.4, the average scores for safety and availability attributes are 4.125 and 4.5 respectively. These numbers show that users have a good response to our application in term of safety and availability. However, although there is satisfaction regarding these two attributes, we still need metrics that measure other

attributes to convince that the application is dependable. We can now just note that our approach achieves some proportion of these significant attributes that might lead us to dependability in software quality.

6.4 Summary

To summarise, we have evaluated several results in our project. We found some techniques that we applied to our project from related work, so that we could evaluate it to ensure the accuracy and the success of our implementation. In addition, we were concerned about the quality of software and we defined the metrics we needed to evaluate the different attributes of quality for the application. In comparison with the related work, our project is more concerned about the usability of the software and its accurate response rather than the improvement of user fitness. A survey was carried out to measure these attributes and the survey results were analysed.

7. Conclusions

7.1 Conclusions

People nowadays are more concerned about their heath and jogging has become one of the most popular activities. They appreciate any methods that can evaluate and guide their exercising. However, they still demand a portable and handy method that has an acceptable cost. As a consequence of the increase in the number of people using smartphone/tablet devices and the extended capabilities of these devices, mobile technology has become the best choice that can support such an application. Our application is associated with the Beathealth project whose purpose is to exploit the pervasive link between rhythm and movement for boosting motor performance, and to enhance health and wellness using a personalised approach (BeatHealth, 2014). In our project, we wanted to develop an application that links between the tempo of song and the movement associated with running. Furthermore, it should be implemented on a mobile device with the aid of the techniques in Software Engineering to ensure that the software satisfies certain quality attributes.

Our proposed solution focused on achieving both the functional and non-functional requirements for this application. In order to capture user's movement, we collected the longitudinal signal data from the built-in accelerometer sensor on smartphone/tablet to analyse the cadence. The *Lomb Periodogram* technique, which is a Time-Frequency analysis tool, was used to find cadence or SPM information in our project to solve the problem of unevenly sampling data obtained from accelerometer sensor using the SensorChanged event. This technique is applied in a short-time interval analysis in order to determine the SPM value in real-time. Then, we introduced the *Phase Vocoder* technique, which is a time-frequency domain transformation that is useful for a number of digital audio effects, to stretch the time of audio file to synchronize the speed of the audio with the SPM value that captures the user's movement.

Besides those functional techniques, we also applied several Software Engineering techniques for our application to ensure the quality of the software. The *Agile Development Lifecycle (SDLC)* technique known as *SCRUM* was used to manage our project through the design, testing and implementation phase. The purpose of applying this technique is for our application to obtain the quality attributes of usability, efficiency and dependability, which were considered to be the non-functional requirements that the application should achieve. To measure these qualities, we needed knowledge of *Software Metrics* to find the measures that can determine by how much the software possesses those qualities attributes.

In elaborating our solution we used the work of (Bart et al., 2010) as guideline (we analysed their work in Section 2.2). Our work differs from their system in the following ways:

- Their system is operated on computer standard PC computer, while our approach was developed for a mobile device.
- We use the Lomb Periodogram technique to analyse the accelerometer data in order to process the cadence or SPM value and we also evaluated the accuracy of its result to ensure that its result is reliable.
- Due to the performance issues, our processing of the audio signal transformations was been done offline.
- We were concerned about the accuracy of response rather than the entrainment which had been their main focus.
- Our project also considered the non-functional requirements and tried to find suitable metrics to measure them.

From our evaluation, we found that by processing the data from the accelerometer sensor with the specific method of the Lomb Periodogram algorithm we could extract the cadence. Further, the results were found to be sufficiently accurate to be used in this health awareness application and this satisfied a critical non-functional requirement. Apart from this, we have implemented the Phase Vocoder algorithm to transform the audio signal so that it works properly to time-stretch an audio file. However, it was found that its performance on a mobile application was not good enough to be able to work in real-time.

In general, we found that dealing with the signal processing in a mobile application development required a large computational capability. This affected our design of the user interface. The use of an early prototype was helpful to have a good design for the user interface and its main functionality was initially assessed to motivate the final design get a commitment from users. However, this had then to be simplified because of the issues with the signal processing algorithms. Using the SCRUM technique to manage the project plan for this mobile application gave us a several benefits. We had a flexible plan that could change and we could prioritize the tasks of work that allowed us to look at the project feature-by-feature and finish the main functionality of the application. We were able to adapt the plan after starting to develop the application that lead us to work on the main task without worrying about the subtasks that were less important to our project outcome. We did not find a set of metrics that were designed to measure the dependability of software on mobile device. Due to the different environment, the dependability metrics for software designed for a standard PC computer could not be directly transformed to measure the same attributes in a mobile application directly. It might be a future topic of research to look for a

proper standard to measure the dependability of mobile applications. This would be important given the dramatic increase in numbers of mobile devices.

Overall, we believe that our result can bring an improvement to the way we understand software quality for mobile application, especially health awareness applications, and that in the future it will help us think about how to create more dependable software systems in this computational area.

7.2 Limitations and Future Work

Although our application can respond the user's movement as it displays the SPM value and play the time-stretched sound that has its BPM synchronized with the analysed SPM value, our first purpose was to development the application that it responds with the synchronized sound processing in real-time. However, it is a limitation of our currently application that it still cannot process the Phase Vocoder algorithm in real-time, so that the time-stretched sound is fixed and might not be completely synchronized with the user's movement. Furthermore, some features, such as tracking the running route or counting the pace are not available in this version. The signal processing limitations lead to a simpler user interface that might not give a strong feeling of interaction with user than could have been.

This is one issue that should be fixed in a future development. An optimization of the Phase Vocoder implementation or algorithm to be able to do its processing in real-time on a mobile device is an important goal to be achieved. Also, more features could be added and the look of the user interface could be improved to enhance the attractiveness of a future version. Moreover, in the evaluation part we found that there is some problem about the delayed response. These delays were caused by the time difference between the SPM processing and duration of audio file. In future work, a key aim should be to find an efficient algorithm that can align the kinetic data and beat without any delay or to keep it at small as possible.

Lastly, finding the metrics to measure the dependability for mobile application is another work that should be done in future. This will yield an improvement in software quality measurement for mobile application development that could also contribute to other similar activities in both academic and industry projects.

References

Apple Computer, (2004), 'Working with Bluetooth Devices', *Apple Computer Inc*, Cupertino, CA, *p.7.*

Andong Z., Marcus C., Yin C., Andreas T., (2012), 'Accurate Caloric Expenditure of Bicyclists using Cellphones', *SenSys '12 Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, New York, USA, *pp.71-84*.

Alanna Greco, (2014), '10 Songs That Are Scientifically Proven to Amp Up Your Workout', *16 Jan 2014*. [Online]
Available at: http://www.bustle.com/articles/11323-10-songs-that-are-scientifically-proven-to-amp-up-your-workout

Amalia D.G., Nicola B., and Daniel A., (2000), 'Traditional (?) Implementations of a phase-vocoder: The tricks of the trade', *COST G-6 Conference on Digital Audio Effects (DAFX-00)*, Verona, Italy.

Apple Inc., (2013), 'iOS Technology Overview', *Apple Inc.*, CA, USA. [Online]
Available at:
https://developer.apple.com/library/ios/documentation/miscellaneous/conceptual/iphoneostechoverview/iOSTechOverview.pdf

Arash, R., Hadi, S., and Mohsen, S., (2010), 'Improving Software Dependability Using System-Level Virtualization: A Survey', *Advanced Information Networking and Applications Workshops (WAINA)*, 2010 IEEE 24th International Conference on.

Ayub, S., Bahraminisaab, A., and Honary, B., (2012), 'A Sensor Fusion Method for Smart phone Orientation Estimation', 13th Annual Post Graduate Symposium on the Convergence of Telecommunications, *Networking and Broadcasting,* Liverpool, UK.

Banerjee, A. and Gupta, S., (2012) 'Your Mobility can be Injurious to Your Health: Analyzing Pervasive Health Monitoring Systems under Dynamic Context Changes', *IEEE Intl. Conf. on Pervasive Computing and Communications,* Lugano, Switzerland.

Barning, F.J.M., (1963), 'Bulletin of the Astronomical Institutes of Netherlands', vol. 17, *pp.22-28*.

Bart M., Leon V. N. and Marc L., (2010) 'D-JOGGER: Syncing Music with Walking', *Proceedings of the Sound and Music Computing (SMC) Conference 2010,* Barcelona, Spain, *pp.451-456*.

BeatHealth, (2014), 'BeatHealth, Health and Wellness on the Beat'. [Online]
Available at: http://www.euromov.eu/beathealth/homepage

Boualem Boashash, (2003), *'Time-Frequency Signal Analysis and Processing, A Comprehensive Reference'*, Protoavis Productions, Seymour, Australia.

Brendan P., Justin C., Daniel J., and Rod B., (2005), 'Use of accelerometers for detecting foot-ground contact time during running', *BioMEMS and Nanotechnology*, Brisbane, Australia.

Brown S., Timoney J., Lysaght T. and Ye D., (2012), 'Software Testing Principles and Practice', *China Machine Press*, China, *pp.2-9,147-151*.

Casey R. and Ben F., (2007), 'Processing: a programming handbook for visual designers and artists', *The MIT Press*, MA, USA.

Chantelle Wilder, (2011), 'Running Cadence: 180 Strides Per Minute is Key', *30 Aug 2011*. [Online]
Available at: http://runnersfeed.com/running-cadence-180-strides-per-minute-is-key/

Christos Saragiotis, (2008), 'Lomb normalized periodogram' [Online]
Available at: http://www.mathworks.com/matlabcentral/fileexchange/22215-lomb-normalized-periodogram

Cohen D., Lindvall M. and Costa P., (2004), 'An introduction to agile methods. In Advances in Computers', New York, USA, *pp. 1–66*.

David R., Bassett Jr., and Dinesh J., (2010), 'Use of pedometers and accelerometers in clinical populations: validity and reliability issues', *Physical Therapy Review*, Vol. 15, No. 3, TN, USA.

Fitbit Inc., (2014), San Francisco, CA, USA. [Online]
Available at: http://www.fitbit.com

Goldstein Ilan, (2013), 'Scrum Shortcuts without Cutting Corners: Agile Tactics, Tools, & Tips', Addison-Wesley Profrssional, Boston, USA.

Google, (2013), 'Nexus 7', CA, USA. [Online]
Available at: http://www.google.com/nexus/7/

Hunt B.R., Lipsman R.L., and Rosenberg J.M., (2001), 'A Guide to MATLAB for Beginners and Experienced Users', *Cambridge University Press*, Cambridge, UK.

Ian Sommervillle, (2001), Sixth Edition, 'Software Engineering', *Pearson Education Limited*, Edinburgh, UK, *pp.6,218*.

Idin K., Oliver S.S., Bryan S., Michelle C., and Karon E.M., (2013), 'RRACE: Robust realtime algorithm for cadence estimation', British Columbia, Canada.

ISO copyright office, (2011), First Edition, 'International Stardard ISO/IEC 25010', Geneva, Switzerland. [Online]
Available at: http://webstore.iec.ch/preview/info_isoiec25010%7Bed1.0%7Den.pdf

Jackson A.H., Marcelo M.W., and Ichiro F.,(2009), 'Real-time Phase Vocoder Manipulation by Runner's Pace', *NIME09,* Pittsburgh, PA, USA.

James S. and Nelson T., (2011), 'The Android Developer's Cookbook, Building Application with the Android SDK', *Pearson Education Inc.*, Baston, USA, *pp1-18,178*.

Jamie M. W. and Nobubuko H., (2009), 'Pedometer and New Technology - Cell Phone & Google Maps What is a Pedometer?', *ARIZON A COOPERATIVE Extension*, Arizona, USA.

Jonathan Ide-Don, (2014), 'CADENCE: HOW YOUR RHYTHM CAN AFFECT YOUR RUNNING', *31 Jan 2014*. [Online]
Available at: http://ptsportswellness.wordpress.com/2014/01/31/cadence-how-your-rhythm-can-affect-your-running/

Joseph Zaloker, (2014), 'ANT/ANT+', *Arrow M2M representative*. [Online]
Available at: https://www.arrownac.com/solutions-applications/machine-to-machine/files/atd-ant.pdf

Jtransform Library, (2011), version 2.4, *the term of the MPL/LGPL/GPL tri-license*. [Online]
Available at: https://sites.google.com/site/piotrwendykier/software/jtransforms

Kara Deschenes (2013), 'Efficient Running Cadence', *3 Apr 2013*. [Online]
Available at: http://womensrunning.competitor.com/2013/03/training-tips/efficient-running-cadence_11060

Maxim G., Elixaveta K., and Alexander O., (2012), 'Research of MEMS Accelerometers Features in Mobile Phone', *Proceedings of the 12th Conference of Open Innovations Association FRUCT*, Oulu, Finland.

McClellen J. H., Schafer R. W., and Yoder M. A., (1998), 'DSP First: a multimedia approach', prentice hall.

Melis O., Jeffry A. P., and Patrick S., (2012), 'Towards the Run and Walk Activity Classification through Step Detection – An Android Application', *34th Annual International Conference of the IEEE EMBS*, California, USA.

Michael M., Mehmet G., Klaus-Hendrik W., Bianying S., and Matthias G., (2008), 'A performance comparison of accelerometry-based step detection algorithms on a large, non-laboratory sample of healthy and mobility-impaired persons', *30th Annual International IEEE EMBS Conference,* British Columbia, Canada.

Microsoft Corporation, (2009), 'Quality Attributes'. [Online]
Available at: http://msdn.microsoft.com/en-us/library/ee658094.aspx

Microsoft Corporation, (2000), 'Usability in Software Design'. [Online]
Available at: http://msdn.microsoft.com/en-us/library/ms997577.aspx

Mohaned F., Mounir B., and Mouldi B., (2008), 'Microcontroller Based Heart Rate Monitor', *The International Arab Journal of Information Technology , Vol.5, No.4*, Zarqa, Jordan.

Möller K.-H. and Paulish D. J., (1993), First Edition, 'Software Metrics, A Practitioner's guide to improved product development', *Chapman & Hall*, London, UK, *pp.40-41*.

Nicholas D. L., Tanzeem, C., Andrew, C., Mashfiqui, M., Mu, L., Xiaochao, Y., Afsaneh, D., Hong, L., Shahid, A. and Ethan, B., (2011) 'BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing', *Proceedings of the 5th International ICST Conference on Pervasive Computing Technologies for Healthcare*, Dublin, Ireland.

Norman E. F. and Shari L. P., (1997), Second Edition, 'Software Metrics A rigorous and Practical Approach'*, PWS Publishing Company*, Boston, USA.

Paul C. and Rory, V. O., (2012), 'The situational factors that affect the software development process: Towards a comprehensive reference framework', *Journal of Information Software and Technology, vol. 54, Issue 5, May 2012. pp.433-44*

Rogers Y., Sharp H. and Preece J., (2007), Second Edition, 'INTERACTION DESIGN beyond human-computer interaction', *John Wiley & Sons Ltd*, West Sussex, UK, *pp8-19,530-572,685-706*.

Rotaa V., Peruccaa L., Simoneb A. and Tesio L., (2011), 'Walk ratio (step length/cadence) as a summary index of neuromotor control of gait: application to multiple sclerosis', *International Journal of Rehabilitation Research*, Milan, Italy.

Ruiz J., Li Y. and Lank E., (2011), 'User-Defined Motion Gestures for Mobile Interaction', *CHI 2011 - Session: Mid-air Pointing & Gestures*, Vancouver, BC, Canada.

Scargle J.D., (1982), 'Astrophysical Journal', vol. 263, *pp.835-853.*

Schwaber Ken, (2004), 'Agile Project Management with Scrum', *Microsoft Press*, USA, *pp. 1-12*.

Scrumsense, (2011), 'What every Product Owner shold know'. [Online]
Available at: http://www.scrumsense.com/wp-content/uploads/2011/02/Product-Owners-Manual.pdf

Totura N., (2012) 'Using Sensors and Location Data for Cutting-edge User Experiences in Mobile Applications', *Intel Corporation: Developer Zone.* [Online]
Available at: http://software.intel.com/en-us/articles/using-sensors-and-location-data-for-cutting-edge-user-experiences-in-mobile-applications

Turner C.W., Lewis J.R. and Nielsen J., (2006), 'Determining Usability Test Sample Size', InternationalExcyclopedia of Ergonomics and Human Factors, vol.3.

Vaniček P., (1971), 'Astrophysics and Space Science', vol. 12, *pp.10-33.*

Wenso Software Solutions, (2013), 'Implement Methodology', Manchester, UK. [Online]
Available at: http://prezi.com/09b8witriltt/implementation-methodology/

William H. P., Saul A. T., William T. V. and Brain P. F. (1992), Second Edition, 'Numerical Recipes', *The Press Syndicate of University of Cambridge*, New York, USA, *pp.530-577*. [Online]
Available at: http://apps.nrbook.com/fortran/index.html

Zölzer U., Arfib A., and Keiler F., and (2002), 'DAFX: Digital Audio Effects', *John Wiley & Sons, Ltd.*, NJ, USA, *pp.237-295*.

Figure References

[1] David Szondy, (2012), 'Wearable ViSi Mobile System lets doctors wirelessly monitor patients', *30 December 2012*. [Online]
Available at: http://www.gizmag.com/visi-mobile-wireless-health-monitoring/25583/

[2] Zölzer U., Arfib A., and Keiler F., and (2002), 'DAFX: Digital Audio Effects', *John Wiley & Sons, Ltd.*, NJ, USA.

[3] Brown S., Timoney J., Lysaght T. and Ye D., (2012), 'Software Testing Principles and Practice', *China Machine Press*, China.

[4] Möller K.-H. and Paulish D. J., (1993), First Edition, 'Software Metrics, A Practitioner's guide to improved product development', *Chapman & Hall*, London, UK.

[5] Wikipedia, 'Processing (programming language)'. [Online]
Available at: http://en.wikipedia.org/wiki/Processing_(programming_language)