Tempo Player - The music player that fits YOUR tempo

P8 Project

Group SW802F15

Software

Department of Computer Science

Aalborg University

Spring 2015

AALBORG UNIVERSITY
STUDENT REPORT

Department of Computer Science
Selma Lagerløfs Vej 300
9220 Aalborg Ø
Phone (+45) 9940 9940
Fax (+45) 9940 9798
http://www.cs.aau.dk

Title:
Tempo Player - The music player that fits YOUR tempo

Subject:
Mobile Systems

Project period:
02-02-2015 – 27-05-2015

Project group:
SW802F15

Participants:
Alexander Drægert
Christoffer Nduru
Dan Petersen
Kristian Thomsen

Supervisor:
Ivan Aaen

Printings: ?

Pages: ??

Appendices: ?

Total pages: ??

Source code:
https://github.com/SW802F15/SourceCode/tree/???????

Abstract:

This report contains Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras consectetur tristique justo, vel pellentesque augue porttitor ut. Suspendisse potenti. Aliquam ipsum metus, commodo eget eros eu, consectetur auctor turpis. Suspendisse potenti. Sed ut felis elit. Proin eget urna placerat ante rutrum scelerisque ac id neque. Quisque gravida porttitor pulvinar. Fusce feugiat arcu at quam hendrerit, nec ultrices dui fermentum. Nam non nibh ac ligula suscipit dictum. Ut sit amet massa at massa ornare lobortis et ac neque. Duis vulputate mauris sit amet dictum iaculis. Praesent tellus velit, lobortis ac elit in, laoreet efficitur tortor. Etiam blandit, mi ac tincidunt tincidunt, ante sapien efficitur leo, eu efficitur nisi elit condimentum sem. Mauris vulputate neque ut lorem consequat semper. Praesent non tempus tellus.

Preface

Aalborg, March 23, 2015

Alexander Drægert

Christoffer Nduru

Dan Petersen

Kristian M. Thomsen

Contents

Introduction 1

Analysis 2

2.1 Problem Statement

Running is a popular form of exercise, however, it can be a tedious and uninspiring endeavour. To improve the experience, Edworthy and Waring [3] found that *"... participants enjoyed what they were doing [running] more when they were listening to music of any sort when compared to when they were not."*

It was further concluded by Edworthy and Waring [3] that the volume and tempo of the music influenced the running experience. They concluded that the running pace for novice runners, while listening to relatively low-tempo music, was slower than when not listening to music. Additionally listening to high-tempo music resulted in a faster running pace, compared to when not listening to music.

This conclusion is in disagreement with the conclusion of Yamamoto et al. [5] which suggests, that *"... music had no impact on mean power output"*.

As a result, we can not definitively conclude whether music of different tempos will affect the running experience differently. However by adhering to Edworthy and Waring [3]'s conclusion, we can only improve the running experience, since Yamamoto et al. [5] concludes there can be no negative impact, by playing music.

Today many runners use their smartphone as a music player, which can either be placed in their hand, pocket, or on their arm. The sensors in a smartphone enable reading the pace and speed of the runner. From this knowledge the first problem can be stated:

How can we provide music with an appropriate tempo, compared to the current pace, to the runner through a smartphone?

Operating a smartphone while running is difficult, especially if it is placed in the pocket or on the arm of the runner. In order for the runner to operate the smartphone properly, the runner would have to stop running, or focus more than normally, which can disrupt the runner's form, which can lead to injuries and accidents.

From this knowledge the second problem can be stated:

How can a smartphone application be operated without disrupting the runner's form or concentration?

Alexander: BPM error deviation = 7.5 BPM - `http://www.cs.virginia.edu/~stankovic/psfiles/MusicalHeart-SenSys2012-CameraReady.pdf`

Ivan: Write about XP

Ivan: Deliberate on what Yamamoto et al. [5] is about.

Methodology 3

3.1 Extreme Programming

XP (Extreme Programming) is an agile development methodology and this project is developed using the principles of XP. XP follows 12 core practices as decribed by Beck [1, p. 54].

The purpose of the 12 practices is to ease the development of high quality software. This is partly done by ensuring thorough testing, continuous refactoring, and efficient knowledge sharing. XP further encourages improving oneself and the team as a whole.

For the practices to work there are some criteria that need to be met, some of those criteria are:

- The size of an XP team should not exceed ten members.
- Iterations should not exceed four weeks, but two-three weeks are preferable.
- The team should be self-organising and should not be controlled by a boss.
- The team members must be able to embrace change.
- The team should have their workstations placed in the same room.
- All the practices must be followed to utilise the full potential of XP.

The listed criteria are fulfilled by this project.

3.1.1 Approach

According to Beck [1, p. 53], the idea behind these practices is that while one practice in itself can be weak, the other practices cover that weakness, creating a synergy effect between the practices.

While all the practices are followed in this project, some need more thought than others. In the following we will explain we use each practice. It is important to note that these descriptions are for ideal scenarios, and some alterations may be made when applying them.

The Planning Game has been discarded in favour of Planning Poker as described by Grenning [4].

Christoffer: do we follow this completely?

Christoffer: skal omskrives til at have en beskrivelse og rationale bag brug af collaborative estimation i stedet for det traditionelle planning game

The main difference is how conflicting estimations are resolved.

Planning Poker starts with the team discussing the task. This ensures everybody understand the scope and task at hand. This will in-turn reduce disparity of the individual estimates, and make it easier to agree on an estimate. Each member then considers his estimate and keeps it to himself. When all are ready, everybody reveals their estimate at the same time. If there is great disparity between estimates, a discussion is organised. When this discussion is over, everybody estimates the task again. If the conflict still exists, estimation of the task is postponed, the task is split or the lowest estimation is taken according to Grenning [4, p. 1].

The Planning Game starts with the team discussing the task. Then each member considers his estimate and reports this to the team. If a conflict arises, the lowest estimate is chosen, as stated by Beck and Fowler [2, p. 58]. This way of reporting individual estimates may influence the estimation of other team member.

Alexander: Explained p.153 description of individual version of planning game (should be used?). Planning p.58 description of collaborative version of planning game (used in report). We originally chose Poker Planning because we only knew the individual version of Planning Game.

Alexander: Explained p.157, Planning game not necessary four teams of 3-4 developers.

Small releases are ensured by prioritising important tasks and making sure to have a working product for each release. A release should not take more than a few iterations, because the short time frame of the project makes it critical that we discover problems as early as possible, allowing us to discard less important features to have time to fix these problems.

Metaphors *"guide all development with a simple shared story of how the whole system works"* according to [2, p.54]. For example a metaphor for this system could be a pacer or a personal trainer. The problem analysis in Chapter 2 describes the metaphor for this system.

Simple design means that all the code in the program must serve a purpose. This is achieved by always focussing on solving the problem at hand, rather than adding code that might solve a future problem. To make sure the design stays simple constant refactoring and testing is done.

Testing is implemented by writing unit tests before writing any production code. After the tests and production code are written, they are run frequently to ensure that everything is working as it is specified, especially after adding new features or refactoring a piece of code.

Refactoring is done whenever a pair discoveres a piece of code, relevant to the issue they are working on, that could be implemented in a simpler way. Pair programing aids refactoring since two people programming together will be more likely to have the courage to refactor complicated pieces of code. The Testing practice also aids in this since tests can be run after refactoring which lessens the possibility of the code breaking.

This is a relatively short lived project, where some parts of the code may not need to change. Therefore, if a piece of code, that is outside the scope of the current issue a pair is working on, is discovered the pair should add a new issue which can later be estimated, prioritised, and included in a later iteration. This way, if the code is not refactored as part of another issue, it will eventually be anyway.This comes at the cost of spending time that could otherwise have been spent on developing other features, but ensures an overall higher quality code.

Pair programming provides an instant code review, and while it has some drawbacks, it encourages each team member to produce better quality code. With some trivial issues pair programming can seem like a waste of time, but it still strengthens some of the other practices, such as test writing and refactoring according to Beck [1, p. 102], making the extra time investment worth it.

Collective ownership means that the whole team owns all the code. This means that if a bug appears, it is the team's problem, and if refactoring is needed, it should be done by the pair who discovers it rather than the pair who implemented it.

Continuous integration is achieved by manually running all tests and running the application every time a branch is merged into the master branch. Each issue is solved on its own branch and merged into the master when the issue is solved. It is important that the master branch always works, i.e., tests must pass and the application should run without any significant issues.

40-hour week is a practice that emphasises not working overtime. If there is not enough time to solve all issues this should be taken into account when planning for the next iteration instead of working late. People are less productive when they are tired, and while it will sometimes be necessary to work more than other, it is important to notice when productivity goes down and react to this accordingly. For us, the use of this practice is complicated by having to take the time spent on lectures into account, and instead working a specific amount of hours each week, we focus on regularly evaluating our productivity and adjusting according to that.

Christoffer: ↑ This paragraph needs more rewriting???

On-site costumer is adapted a bit, as we serve as our own customer. This does not mean that each person can simple make up a requirement, but rather that the team as a whole should be asked whenever there is a question that would otherwise be directed at the customer. Additionally the team must fulfil the other tasks done by the customer, such as writing user stories and acceptance tests.

Coding standards used are in an informal form. The group has worked together multiple times and as a result have adapted to writing code in a similar way. Differences between team members' code occur, but overall the code produced looks similar. The informal coding standards improve collective ownership since no one can necessarily say "He wrote this code, I can tell. It's his problem!" because the code looks alike. It also aids refactoring because the refactorer is used to the code conforming to a code standard and thus is easier to read and rewrite.

Christoffer: Yderligere tilføjelser? Holder det vand?

Alexander: Should be elaborated upon.

Step Counter 4

In the following sections, the development of our step counter module will be explained.

Christoffer: Intro omskrives?

4.1 Use cases

4.2 Specification

4.2.1 Accelerometer

There were no libraries available for the Android API version 16, so we had to develop a way of determining when a step was taken based on the data from the accelerometer.

4.2.1.1 Test Goal

The tests of the accelerometer was performed to determine behaviour during different movement patterns.

Ivan: Define words like ‘behaviour’ and ‘movement patterns’.

4.2.1.2 Test Setup

For the tests we had:

- Four people
- Two smartphones (one Samsung Galaxy Note II and one Samsung Galaxy S III)

We use four different people to ensure the measurement are general. Although four people does not necessarily represent all people, but it is better than one.

Alexander: Better reasoning ↑

We use two different types of smartphones to ensure consistency.

Alexander: Describe reasoning ↑

4.2.1.3 Test Procedure

In order to determine the behaviour of the accelerometer in different movement patterns, we devised five tests. The data we will gather from each test is measurements of:

- Movement according to X, Y, Z axis from the accelerometer.
- Position according to X, Y, Z axis from the gyroscope.
- Gravity.
- Estimated number of steps.
- Time between measurements.

The first test consisted of each person placing both phones in each pocket and then performing the following tasks.

The test is iterated four times, one for each possible rotation of the phone. For each iteration the following tasks are performed.

1. Walk 150 steps, save the gathered data.
2. Jog 150 steps, save the gathered data.
3. Run 150 steps, save the gathered data.
4. Sprint 150 steps, save the gathered data.

The second test consisted of each person placing both phones on their arms and then performing the same tasks as in the first test.

The third test consisted of each person placing both phones in their hands and then performing the same tasks as in the first test.

The fourth test consisted of both smartphones being placed statically on a level table.

The fifth test consisted of both smartphones being held statically by a person.

4.2.1.4 Test Results

Alexander: Check for correct tense (past-tense vs. present-tense) in entire chapter.

4.3 Implementation

Conclusion 5

5.1 Discussion

5.2 Future Work

5.3 Conclusion

Appendix

Project CD A

The CD found on this page contains the following:

- The source code for
- A compiled version of
- A digital version of the report in PDF format.

Bibliography

[1] Kent Beck. *Extreme Programming Explained: Embrace Change.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000. ISBN 0-201-61641-6.

[2] Kent Beck and Martin Fowler. *Planning Extreme Programming.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 2000. ISBN 0201710919.

[3] Judy Edworthy and Hannah Waring. The effects of music tempo and loudness level on treadmill exercise. *Ergonomics*, 49(15):1597–1610, 2006. doi: 10.1080/00140130600899104. URL `http://dx.doi.org/10.1080/00140130600899104`. PMID: 17090506.

[4] James Grenning. Planning poker. `http://renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf`, 2002. [Online; Accessed 03-03-2015].

[5] T. Yamamoto, T. Ohkuwa, H. Itoh, M. Kitoh, J. Terasawa, T. Tsuda, S. Kitagawa, and Y. Sato. Effects of pre-exercise listening to slow and fast rhythm music on supramaximal cycle performance and selected metabolic variables. *Archives Of Physiology And Biochemistry*, 111(3):211–214, 2003. doi: 10.1076/apab.111.3.211.23464. URL `http://dx.doi.org/10.1076/apab.111.3.211.23464`. PMID: 14972741.

Examples & ToDo B

Alexander: Example of comment/ToDo made by Alexander

Christoffer: Example of comment/ToDo made by Christoffer

Dan: Example of comment/ToDo made by Dan

Kristian: Example of comment/ToDo made by Kristian

Ivan: Example of comment/ToDo made by Ivan

```java
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World");
    }

}
```

Listing B.1: Caption of code snippet

This is how you refer to a source written by Edworthy and Waring [3].

List of Todos

4, 🟩 **Alexander:** BPM error deviation = 7.5 BPM - `http://www.cs.virginia.edu/~stankovic/psfiles/MusicalHeart-SenSys2012-CameraReady.pdf`

4, 🟥 **Ivan:** Write about XP

4, 🟥 **Ivan:** Deliberate on what Yamamoto et al. [5] is about.

6, 🟧 **Christoffer:** do we follow this completely?

6, 🟧 **Christoffer:** skal omskrives til at have en beskrivelse og rationale bag brug af collaborative estimation i stedet for det traditionelle planning game

6, 🟩 **Alexander:** Explained p.153 description of individual version of planning game (should be used?). Planning p.58 description of collaborative version of planning game (used in report). We originally chose Poker Planning because we only knew the individual version of Planning Game.

6, 🟩 **Alexander:** Explained p.157, Planning game not necessary four teams of 3-4 developers.

7, 🟧 **Christoffer:** ↑ This paragraph needs more rewriting???

8, 🟧 **Christoffer:** Yderligere tilføjelser? Holder det vand?

8, 🟩 **Alexander:** Should be elaborated upon.

9, 🟧 **Christoffer:** Intro omskrives?

9, 🟥 **Ivan:** Define words like 'behaviour' and 'movement patterns'.

9, 🟩 **Alexander:** Better reasoning ↑

9, 🟩 **Alexander:** Describe reasoning ↑

10, 🟩 **Alexander:** Check for correct tense (past-tense vs. present-tense) in entire chapter.

19, 🟩 **Alexander:** Example of comment/ToDo made by Alexander

19, 🟧 **Christoffer:** Example of comment/ToDo made by Christoffer

19, 🟦 **Dan:** Example of comment/ToDo made by Dan

19, 🟨 **Kristian:** Example of comment/ToDo made by Kristian

19, 🟥 **Ivan:** Example of comment/ToDo made by Ivan