Dissertation

ADDICTED: ADaptive mobile aDvertIsments based on ConTExtual Data

CONSTANDINOS MAKRIS

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2017

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

ADDICTED: ADaptive mobile aDvertIsments based on ConTExtual Data

Constandinos Makris

Supervisor

Dr. Georgios Samaras

Co-Supervisor

Dr. Andreas Pamboris

The individual thesis submitted for partial fulfillment of the requirements for obtaining the degree of computer science, department of computer science, University of Cyprus

May 2017

Acknowledgements

I would like to express my sincere thanks to Dr. George Samaras, the supervisor of my dissertation, for believing in me and giving me the opportunity to undertake this project. By giving me this opportunity, during the course of the dissertation, I was able to study and acquire knowledge that is essential for my academic and professional career.

I place on record, my sincere thank you to Dr. Andreas Pamboris, the co-supervisor of my dissertation, for all the assistance and guidance he provided me during the whole year of my dissertation and for his overall trust in me.

I am also grateful to Dr. Marios Belk, for his support and assistance during the development of the dissertation project.

Finally, I want to thank my family, for the continuous encouragement, support, attention and patience.

Abstract

The choice between two dominant monetization strategies for mobile apps, i.e. capitalizing on revenue generated from either users or advertisers, is not a straightforward one. While users are initially attracted more to free (ad-supported) apps, paid (ad-free) apps are more likely to persist over time. That happens due to the fact that the overuse of ads reduces user experience and eventually overall user satisfaction.

The purpose of this dissertation is the design and implementation of a system that strikes a balance between the two extremes: ADDICTED re-factors ad-supported apps automatically by adjusting the type and frequency of mobile ad occurrences at runtime based on policies that consider mobile factors – internet consumption and battery life, as well as the human factor - the user's physical and emotional state that derive from raw data from a wearable device.

These policies were implemented in a mobile app and a wearable app that were developed and tested on more than 30 users.

Table of Contents

Chapter 1

Introduction

1.1 Thesis overview

The concern that mobile ads erode user experience, causing low app retention rates, is commonplace among developers. The only alternative solution at the moment is to charge users when using an app, basically forcing them to buy the app. This approach creates a cost-barrier and reduces the number of possible users. Just 0.2% of paid apps in the Android Play Store, in contrast to a significant 20% of free apps, are downloaded more than 10,000 times [1].

Therefore, developers are faced with a critical dilemma: (1) to earn faster income through mobile advertising, with a high probability that this will drop significantly over time, or (2) to earn a more slow but stable income over longer periods of time by requiring users to pay for app downloads and the full ad-free app experience.

In this thesis, we propose ADDICTED a system developed and implemented to tackle this problem by dynamically adapting mobile advertisements in real time based on emotional and physical contextual data gathered from a wearable device. ADDICTED comprises of policies that were created based on the result of our algorithms that detect emotional and physical state i.e. “Stressed” or “Relaxed”, “Moving” or “Not moving”. Those algorithms were developed and implemented based on literature and we achieved an accuracy of 75% and 97.5% respectively.

1.2 Problem Statement

As mentioned before, mobile app developers are faced with a critical dilemma regarding the approach the are going to follow to create their application. There are two ways to do that. The first approach is a free application filled with ads that will be the main source of income for the developer or a paid application with no ads that the main source of income is users buying that app. There are pros and cons with either of the approaches but most of the time developers decide to go with the free app approach which yields a bigger revenue in a shorter period of time. The problem with this is that most developers abuse the use of ads to gain more money. Thus, users are bombarded constantly with ads that decrease user experience and user satisfaction. Eventually users stop using those apps and developers end up with no revenue.

Therefore, the solution to this problem is to come up with an idea that will tackle this problem and strike the balance between the two – user experience and developer revenue. The idea is to create a system that will increase user experience but also increase the revenue of the developer.

1.3 Motivation

Nowadays the users prefer downloading free apps despite the fact that sometimes they will be forced to come across intrusive or unwanted ads that may or may not interfere with the use of the application.

As a user, I sometimes catch my self being annoyed by constant ads while I am using an application especially when that advertisement interferes with the use of the application. E.g. Using a free memo notepad on a mobile device mainly for writing down appointments and suddenly while trying to write down a very important meeting, the application shows a video advertisement that has the user waiting for almost 30 seconds. At that moment 30 seconds seem frustrating enough that may make the user to close the app and note the details of the meeting on a piece of paper. We see that depending on the situation, ads can be frustrating. If the same happened while the user was just reviewing the noted appointments he might not mind at all.

Taking all these into consideration and with the rise of wearable devices, we came up with the idea of finding a way to dynamically change the type format and the frequency of the ad, depending on the physical and emotional state of the user the exact time that he is using the application based on readings that will be collected in the background from the wearable device of the user.

1.4 Scope of the Thesis

The scope of this thesis is to extend and enhance policies that were developed and implemented in "AD-APT: Blurring the Boundary Between Mobile Advertising and User Satisfaction", the first version of ADDICTED. AD-APT policies focused on device factors to adapt in real time the frequency of ads. By taking that into consideration and combining those policies with new policies based on the human factor - the user's physical and emotional state we created a system that includes both the old and new policies to further improve user experience.

Chapter 2

Background

2.1 Mobile Advertising Systems

2.2 Mobile ad formats for Android apps

2.3 Heart Rate Variability – HRV

Introduction

At the beginning of this chapter I will present the Mobile Advertising Systems that exist today. After the brief explanation, I will present the Mobile ad formats that are used in android apps and I will briefly explain their use. Lastly, I will refer and explain the Heart Rate Variability - HRV assessing method.

2.1 Mobile Advertising Systems

Mobile advertising systems today, consist from five *entities* (Figure 1) [14]:

- ***mobile clients*** that use ad control modules such as the AdMob Android module to load ads on mobile devices.
- ***advertisers*** that supply the corresponding ads by submitting ad campaigns to ***ad networks***, which specify a budget, a target number of impressions/clicks and a deadline
- ***ad servers*** that process requests from clients when an ad slot becomes available and track the number of views and clicks per ad
- and ***ad exchanges*** that hold an auction during which ad networks place a bid on behalf of the advertisers

Targeted advertising is key for the companies to maximize their revenue. Ads are assigned to available slots at runtime based on the user location, context of execution and other characteristics of the user.

2.2 Mobile ad formats for Android apps

The most basic ad unit is the common banner ad. The ad size for mobile apps has more-or-less standardized to 320x50 (Figure 2). One can find a few slightly smaller variations of that, but in general banner ads fit within a space of that size.

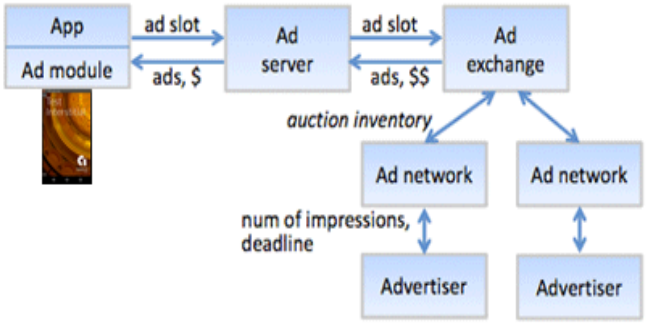

Figure 1.Architecture of a typical mobile ad

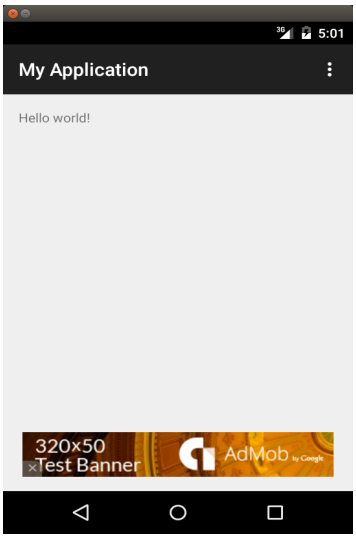

Figure 2. Standard banner ad

This type of ad is under the category of on-screen advertising. With tablets and larger android devices becoming more common, there are other sized ads, such as 300x250 and square ad units. Also, rich media ads are also under the category of on-screen advertising. Rich media ads contain video or other interactive elements and some of them can be configured to pop up from a small banner to a full-screen ad. This can combine elements of on-screen and intermittent advertising formats.

Intermittent ads are another category of ads that might include full-screen ads, videos and a number of ad walls (also known as offer walls or unlockers). There are also pop-ups and other event-driven ad displays. One common intermittent ad is the interstitial ad. Interstitial ads are full-screen ads that cover the interface of their host app (Figure 3).

They're typically displayed at natural transition points in the flow of an app, such as between activities or during the pause between levels in a game. When an app shows an interstitial ad,

the user has the choice to either tap on the ad and continue to its destination or close it and return to the app. The direct call to action and larger size combine to make interstitial ads a

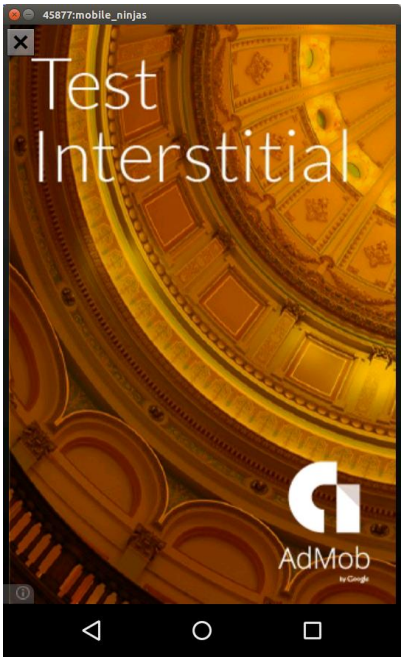

Figure 3. Interstitial

particularly effective form of mobile advertisement. There are a number of different types of interstitial ads: text, image, video, and more.

Another not so frequent type of ad is the out-of-app advertising. Out-of-app advertising takes advantage of having access to the device even when the user is not using your app. Many users do not expect the ads to continue after they stop using your app and this methodology may catch them by surprise. Google recently made changes to the developer policy to prevent some bad practices and prevent both accidental and intentional violations of the good faith of the users. Ad networks that offer these kind of ad formats, generally require a statement in your apps description to let the users know that your app is supported by this kind of advertising.

ADDICTED is focused on the first two categories, on-screen and intermittent ads.

2.3 Heart Rate Variability – HRV

The body reacts to nearly everything happening around it through emotions, observations, thoughts and activity. The brain guides the body by regulating heart and other organs through autonomic nervous system. This physiological variation of heart rate, controlled by

autonomic nervous system, is called Heart Rate Variability (also commonly known as HRV). Measuring heart rate variation reveals wide range of information about your body and health.

Heart rate variability (HRV) is calculated based on variation of time in milliseconds between two heartbeats. HRV varies as you breathe in and out (Figure 4). HRV is a relatively new method for assessing, for example, stress. What makes HRV interesting is the fact that it can reflect changes in stress while other physiological parameters, like blood pressure, are still in normal or accepted ranges.

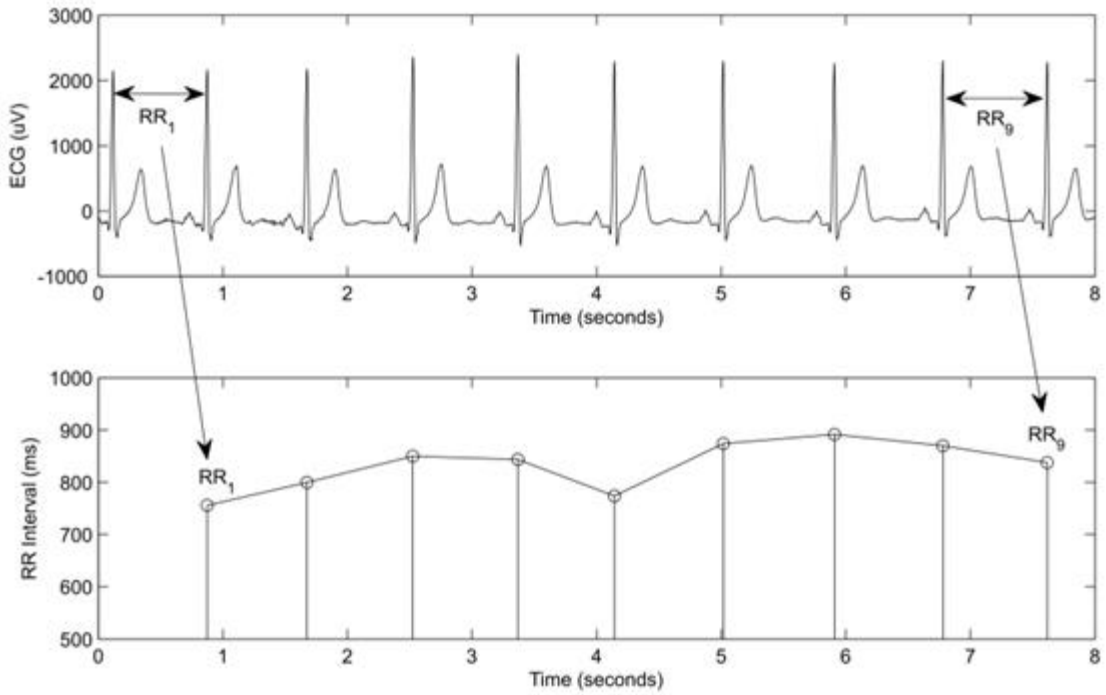

Figure 4. Heart rate variability.

Scientific evidence has linked high heart rate variability to good health and fitness. In contrary, decreased HRV is linked to stress, fatigue or even burnout. Because of this it is important to monitor your HRV during normal everyday life. When measurement of physical activity is combined with HRV it is possible to analyze and evaluate the reason for low or high HRV. That is why HRV is becoming an increasingly popular parameter in the fields of sports and sports science, corporate health, cardiology, ergonomics, diabetes care and relaxation training therapy. HRV is also being widely used on physiological research of autonomic nervous system.

Chapter 3

Literature Review

Introduction

In this chapter I will present the selected publications that we used to back up our new algorithms and I will review those selected publications. Finally, I will explain in pseudocode the structure of the algorithms that are presented in both papers.

To be able to develop our system, we firstly had to develop our algorithms that would determine the physical and emotional state of the user. To do that we had to find publications to base our algorithms on and backup our work. After reviewing a pool of candidate papers, we decided to base our algorithms on two papers, one for the emotional state recognition algorithm(a) and one for the physical state recognition algorithm(b). The two papers that were selected are:

1. **Title: Mathematical Algorithm for Heart Rate Variability Analysis**
   Authors: Alberts Aldersons, Andris Buikis
   University of Latvia Raina bulv. 29, Riga, LV1459 LATVIA
   Published in:
   Proceeding AIASABEBI'11 Proceedings of the 11th WSEAS international conference on Applied informatics and communications, and Proceedings of the 4th WSEAS International conference on Biomedical electronics and biomedical informatics, and Proceedings of the international conference on Computational engineering in systems applications
   Florence, Italy — August 23 - 25, 2011

In this paper, the authors use HRV (Heart Rate Variability) to determine based on an algorithm that they developed, if the subject is stressed or not.

Their algorithm uses HRV to generate positive biofeedback signals (PBFS) or negative biofeedback signals (NBFS). To do that they take a sliding window of four consecutive HRV

values and evaluate the arithmetical difference between two consecutive intervals. If the arithmetical difference between the two consecutive intervals has identical sign with the sign of the arithmetical difference between the previous two consecutive intervals then a positive biofeedback signal(PBFS) is generated. If they have a different sign then a negative biofeedback signal (NBFS) is generated. (Algorithm will be analyzed at the end of the review)

After a period of 20 seconds, i.e. after 20 consecutive HRV readings a "Central Index" (CI) is calculated according to formula:

$$The\ sum\ of\ all\ PBFS\ *\ 100\ /\ (The\ sum\ of\ all\ PBFS\ +\ The\ sum\ of\ all\ NBFS)$$

They also state in the paper that the absolute smallest number of pulse beats needed to calculate the average heart rate and therefore the HRV is 10-15 beats. They also state that after the first calculation of the emotional state it is possible to calculate the next accurate emotional state after 1-6 minutes. Furthermore, it is also stated that this approach is not generating results that describe the average state of minutes or hours, but about 1 second.

Their algorithm is implemented as follows:

The following convention is adopted:

P(n) = the time moment of the current heart beat (fourth, if the calculations carried out at the fourth heart percussion);

P(n-1) = (third, if the calculations carried out at the fourth heart percussion);

P(n-2) = (second, if the calculations carried out at the fourth heart percussion);

P(n-3) = (first, if the calculations carried out at the fourth heart percussion);

T(n) = time interval between P(n) and P(n-1);

T(n-1) = time interval between P(n-1) and P(n- 2);

T(n-2) = time interval between P(n-2) and P(n- 3);

Beginning from 4-th pulse beat and forth, each

pulse beat is granted with following designation

(Parameter V(x)=+ or V(x)=–) according to

following algorithm:

V(n) =" +", if T(n) > T(n-1);

V(n) =" -", if T(n) < T(n-1);

V(n) =" 0", if T(n) = T(n-1);

V(n-1) =" +", if T(n-1) > T(n-2);

V(n-1) =" -", if T(n-1) < T(n-2);

V(n-1)=" 0", if T(n-1) = T(n-2)

On each pulse beat beginning from the fourth

beat, the following calculations are made:

If V(n) =" +" and V(n-1) =" +"

then PBFS (Positive Biofeedback Signal) = PBFS+1

If V(n)="-" and V(n-1) =" -"

Then PBFS=PBFS+1

If V(n)=" +" and V(n-1) =" -"

then NBFS (Negative Biofeedback Signal) = NBFS+1

If V(n)="-" and V(n-1) =" +"

Then NBFS=NBFS+1

If V(n)="0" and V(n-1) =" +"

Then NBFS=NBFS+1

If V(n) =" +" and V(n-1) =" 0"

Then NBFS=NBFS+1

If V(n)=”0” and V(n-1) =” -”

Then NBFS=NBFS+1

If V(n)=”-” and V(n-1) =” 0”

Then NBFS=NBFS+1

If V(n)=”0” and V(n-1) =” 0”

Then NBFS=NBFS+1

For time period of 20 seconds or longer the “Central Index (CI)” is calculated as following:

Central Index (CI) = amount of all PBFS / (all PBFS and NBFS amount) * 100 (percent).

Their algorithm and approach are fairly simple, straightforward and fully explained in the paper. For the above reasons this paper was selected to base our Emotional State Recognition algorithm upon.

2. Title: **Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phones**

Author: Jun Yang
Nokia Research Center Palo Alto, CA 94304, USA

Published in:
Proceeding
IMCE '09 Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics - Beijing, China — October 23 - 23, 2009

In this paper, they perform physical motion recognition using mobile phones with built-in accelerometer sensors.

The accelerometer sensors inside the mobile phones are tri-axial accelerometer sensors that recognize separately the X,Y and Z-axis movements. Same accelerometer sensors are in the wearable devices.

Sensor data processing and smoothing techniques are discussed first to reduce the special noise present in phone-collected accelerometer data.
They also explore orientation-independent features extracted from vertical and horizontal components in acceleration as well as magnitudes of acceleration for six common physical activities.
The activities are:

- Sitting
- Standing
- Walking
- Running
- Driving
- Cycling

They claim that decision tree achieves the best performance among four commonly used static classifiers, while vertical and horizontal features have better recognition accuracy than magnitude features. Also, a well-pruned decision tree with simple time domain features and less over-fitting on the training data can provide a usable model for inferencing a physical activity diary, refined by a similarity match from k-means clustering results and smoothed by an HMM-based Viterbi algorithm.

They collected the data using mobile phones (Nokia N95) with built in tri-axial accelerometers. They collected data for everyday activities that involved normal body movements and range in intensity levels. Their time frame window was 10s and they uploaded the data to a server to process it after the task was finished.
Data Processing algorithm: Each reading of accelerometer sensor consists of 3D accelerations along X-axis, Y -axis and Z-axis according to local coordinate system of current phone orientation. (Figure 5).
They used an algorithm to reduce the accelerometers jittering (newer mobile phones and sensors shouldn't have this problem).

As illustrated in Figure 5, accelerometer readings from the phone of person A sitting are plotted in the subplot. Although the phone is located in a pocket and positive Z-axis values indicate that the phone is almost facing downward, different orientations cause Z readings varying from case to case.

One easy solution to avoid orientation problem is using magnitude of each (X, Y, Z) accelerometer signal. However, this kind of operation will cause acceleration loss 3-d directional information. Related work [11], has shown the signal average on each axis over a reasonable time period can produce a good estimate of the gravity-related component. The authors of the paper take a similar approach here to estimate the gravity component from each segment of (x, y, z) readings. Their approach to remove the z-axis orientation problem will be explained in the next Chapter.

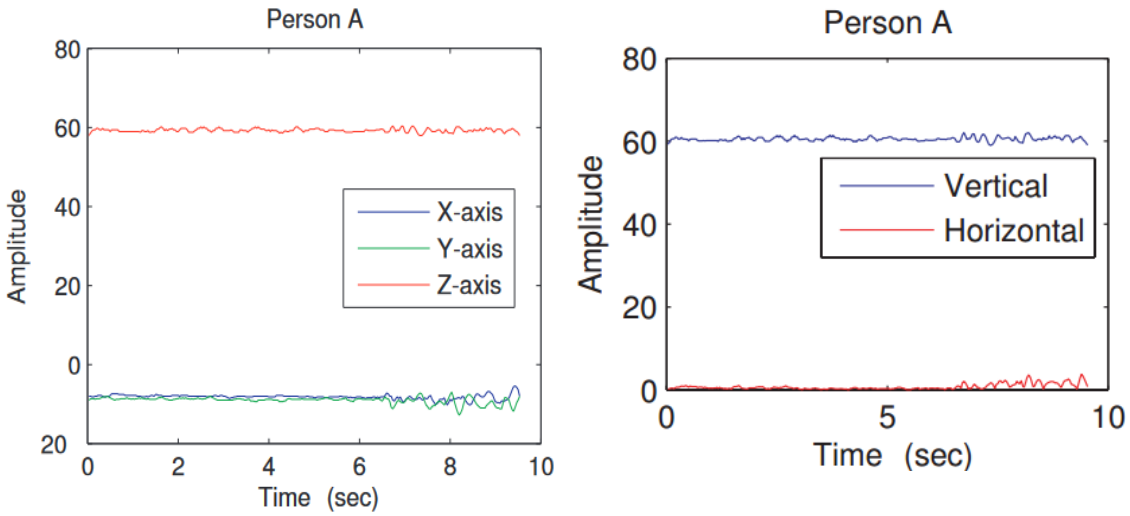

Figure 5. X, Y and Z-Axis readings of accelerometer sensor before and after data manipulation

For each axis, they used the mean Value and the standard deviation e.g. meanH (mean horizontal reading) and stdH (standard deviation Horizontal reading). With these readings and the use of a decision tree with certain thresholds for each measurement, they decided the activity of the subject (Figure 6).

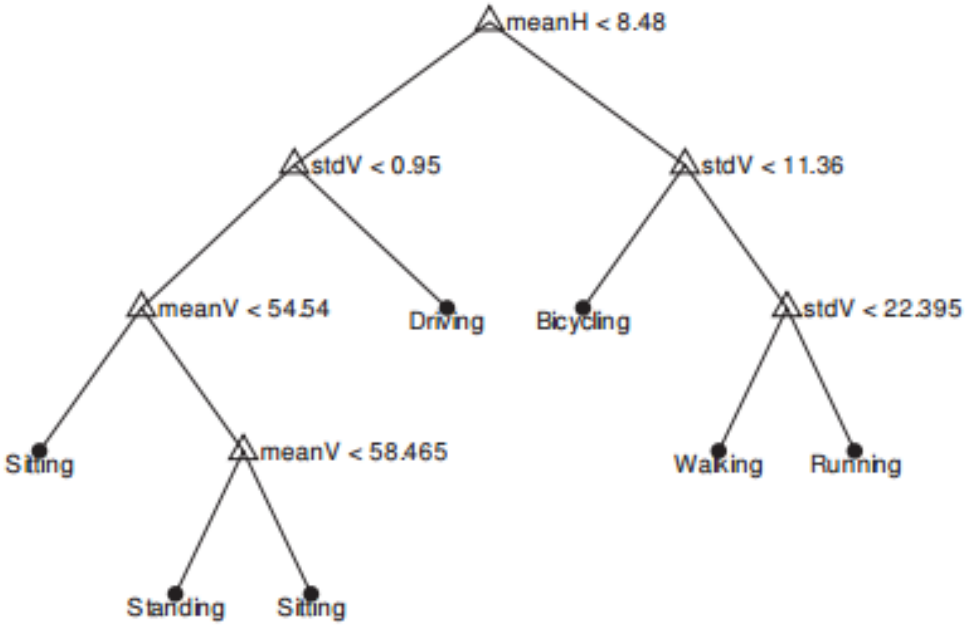

Figure 6. Physical state decision tree.

Chapter 4

Design and Development

Introduction

In this chapter I will present the AD-APT system that was developed and published in IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft'16). Later I will explain the design of both PSR and ESR algorithms of ADDICTED. The structure of each algorithm will be explained and any differences or similarities with the algorithms found in the literature that we used. Also, the parsing procedure of the raw data from the wearable device to the handheld device will be explained using pseudocode. Furthermore, the modified decision trees will be presented for both algorithms as well as the policies that we created will be shown and thoroughly explained. A reference to the devices and tools that we used is made later in this chapter. Also, all the versions and modifications of the testing application are presented with detailed explanations regarding their functionality. Finally, I will present a demo application that shows how the ADDICTED system with its policies will be implemented in an application.

4.1 AD-APT Blurring the Boundary Between Mobile Advertising and User Satisfaction

(Authors: A. Pamboris, G. Antoniou, C. Makris, P. Andreou, and G. Samaras)

AD-APT was the initial idea and approach for our system. AD-APT was developed, implemented and published in IEEE/ACM International Conference on Mobile Software

Engineering and Systems (MOBILESoft'16), ser. Proceedings of the 3rd IEEE/ACM International Conference on Mobile Software Engineering and Systems, vol. 1. Austin, Texas, United States: IEEE, May 2016.

This paper proposes AD-APT, a solution that tries to balance the associated tradeoff by dynamically adjusting the frequency of mobile ads in apps to adapt to changes in runtime conditions such as the device's battery life, the type of network connectivity and limits on network usage.

In particular, AD-APT offers the mechanism for transforming automatically ad-supported Android apps at the bytecode level by wrapping ad-related method calls around conditional statements that adjust the frequency of ad occurrences at runtime. AD-APT also provides a framework for specifying the corresponding adaptation policies such as: display X% of ads when battery drops below Y%.

AD-APT was evaluated using ten real-world Android apps with mobile ads. Our prototype implementation is able to react to changes in runtime conditions by reducing network usage by up to 30× and energy consumption by up to 40%.

Network and Energy Cost of Mobile Ads

Previous work [12, 13, 14, 15, 16] has shown that mobile ads incur significant hidden costs, which amount to approximately a 15% and 97% increase in app energy and network consumption, respectively. The energy overhead is primarily caused by tail time [17], i.e. an artificial delay introduced by network providers when transitioning between radio power states. This delay helps control the responsiveness of apps when new network activity starts by avoiding frequent time-consuming power state transitions during consecutive network operations. However, a longer tail time consumes significantly more power, especially for short transmissions spaced with rest periods, for e.g. when downloading mobile ads. While for Wi-Fi networks, tail time is significantly less compared to 3G and 4G networks, Wi-Fi consumes more energy per second of use, which compensates for the difference [18]. The network overhead depends on the corresponding types of mobile ads that are downloaded at runtime. These include standard and expandable banners, interstitial (full screen) ads with animations, and short video ads. According to a pioneer mobile ad network, these range in size between a few tens of KBs to tens of MBs [19]. Mobile ads can therefore quickly become

a bottleneck for users on metered data plans, considering apps that display tens of mobile ads during a single app session.

Ad Prefetching and Caching Approaches

To reduce the energy cost of mobile ads, previous work has proposed mobile ad prefetching and caching techniques. Based on ad traffic traces, it has been shown that approaches that prefetch ads can yield significant energy savings by avoiding the tail energy overhead [13, 16]. More recent proposals [14] go one step further to align with the real-time nature of modern advertising systems: they predict future ad slot availability and replicate ads across multiple clients using an overbooking model, which aims at meeting ad expiration deadlines. Approaches based on prefetching and caching suffer from the limitation that ads may become stale before they are displayed on the device, which in turn leads to revenue loss and SLA violations. Even the work by Mohan et al. [14], which explicitly tries to address this concern, heavily relies on an accurate prediction model for future ad slot availability, and on large enough ad deadlines to effectively reduce energy consumption.

In contrast, AD-APT assumes nothing about the app workflows or the associated ad deadlines. Furthermore, as opposed to the aforementioned approaches, AD-APT is able to reduce both the energy and network overhead caused by mobile advertising.

System Architecture

Developers first specify the policies that drive the decisions regarding when to load ads at runtime through the Policy Specifier. While a predefined set of basic policies is provided, developers also have the option to design custom policies using a policy framework API. During the next step, ad-supported apps are automatically refactored by the AD-APT Compiler to incorporate the policies into their business logic. Android apps in the Dalvik executable form (.dex files) are first converted to a more understandable readable form, i.e. the smali language. They are then modified according to the policies provided by developers, and the new app versions are converted back to Dalvik executables.

Finally, the AD-APT Dispatcher packages the Android modules into .apk files, which include all the information necessary to run an app, e.g. the .dex files and resource files. It then digitally signs the modified app with a certificate to allow for installing it on the device.

Policy Framework

AD-APT offers a choice between basic policies to adjust the frequency of mobile ad occurrences. Such policies include: (1) if battery level Q B% then ad frequency = F%; and (2) if network connectivity type == TYPE then ad frequency = F% (where TYPE = {Wi-Fi | 3G | 4G}). In addition, AD-APT provides a framework for specifying more complex custom policies that account for arbitrary combinations of information such as the device's remaining battery life, network usage and the network connectivity type.

Custom policies need to be specified using the following format: policy: {(condition), (adjustment)}, where condition is a boolean expression representing the rules that must be satisfied in order for the adjustment to be triggered. For example, a policy that loads only 25% of ads when (1) the battery drops below 50% and (2) the device is connected over 3G, is specified as follows: Policy1: {((battery level, < , 50%) AND (network connectivity type, ==, 3G)), (ad frequency = 25%)}. AD-APT then automatically translates custom policies into Java code, which is used to re-factor apps accordingly (see Section 4). This code uses Android APIs such as the android.telephony, android.net and android.content APIs, which expose the required device information.

Application re-factoring

Given a policy, AD-APT automatically generates a Java method called adAptPolicy, which checks whether the policy conditions hold and, using a probabilistic formula, returns a boolean value to indicate whether the next mobile ad will be loaded. The AD-APT Compiler first generates the binary files for the adAptPolicy method, which along with the original app's binary files are reverse-engineered to obtain the equivalent smali code, i.e. an intermediate representation of the Dalvik bytecode, using the Baksmali disassembler [20].

AD-APT then statically analyses the smali code to identify execution points that are responsible for loading mobile ads, for example, calls to a loadAd method used for retrieving ads from a list of trusted domains. Such calls are wrapped around a conditional statement that determines whether a mobile ad will be loaded based on the return value of adAptPolicy. If adAptPolicy returns false, ADAPT bypasses the call to loadAd, otherwise execution continues as it would normally. To include functionality in the modified app that uses either

custom or built-in Java libraries, the corresponding smali code for such libraries needs to be included in the modified app's smali code.

In particular, the Baksmali disassembler generates a folder structure similar to Java packages and all new libraries used by the modified app, for example, libraries related to the policy framework API, are copied into the corresponding folders of the modified app.

Finally, the AD-APT Compiler re-compiles the modified smali files into binary files using the Smali assembler [20]. The ADAPT Dispatcher then re-signs the modified app using the ZipSigner tool [21] to allow for installing it on the device.

Evaluation

AD-APT is evaluated on a set of ten diverse real-world Android apps, which include networked apps, mobile games and web browsers. For our experiments, we use a Samsung I9505 Galaxy S4 mobile device with the Android v5.0.1 OS. We conduct experiments with two types of network connectivity: (i) a 802.11g Wi-Fi network with an average round trip time (RTT) of 23 ms and an average bandwidth of 8 Mbps; and (ii) a 3G mobile network with an average RTT of 425 ms and bandwidth of 0.4 Mbps.

To measure network and energy consumption, we use AT&T's Application Resource Optimizer (ARO) [22], which is a diagnostic tool for monitoring an app's behavior at runtime. We run all apps under typical workloads for a minimum of 5 mins and until at least ten mobile ads are loaded in the unmodified apps. This number was empirically chosen to ensure that the specified frequencies of ad occurrences are realised in practice. For all apps considered, this took at most 7 mins of execution. During each experiment, we measure both the network and energy usage of different versions of apps, i.e. versions for which the frequency of ad occurrences is set to 0% (without mobile ads), 25%, 50%, 75% and 100% (unmodified apps). The results reported correspond to averaged values over five experimental runs with negligible variations across runs.

Network Usage

First, we evaluate AD-APT's ability to reduce network usage by adjusting the frequency of mobile ad occurrences at runtime. For approximately five minutes of use per app, mobile ads cause additional network usage that ranges from 1 MB to 13 MBs. The most notable savings

in network usage can be obtained by apps that include ads that are larger in size, namely video and interstitial ads. In the set of apps considered, two of them include video ads, while another app uses interstitial ads with animations. For such apps, savings of up to 30× in network usage are attainable using AD-APT. Users on metered data plans implicitly pay a cost for using ad supported apps by downloading mobile ads during execution. ADAPT, however, offers the option to reduce the associated network overhead when connected to 3G/4G networks using appropriate policies and without disabling mobile ads altogether.

Energy Usage

Next, we evaluate how AD-APT can improve the energy consumption on mobile devices. Using the app versions with 0% frequency of mobile ad occurrences as reference, mobile ads may contribute a significant increase to the total app energy consumption, which ranges between 17%–40%. Reducing the frequency from 100% to 0% reflects nicely on the energy consumption of apps since less ads are retrieved at runtime, which avoids part of the tail energy overhead. Even for a small reduction by 25% (i.e. 75% frequency), AD-APT is able to yield savings of up to 20% in battery life. The corresponding results for Wi-Fi connectivity follow similar trends and are therefore omitted.

Since tail time for Wi-Fi networks is significantly less than for 3G/4G networks, the savings in energy consumption are slightly reduced for some apps. Nevertheless, Wi-Fi consumes more energy per second of use, which explains why the reduction is negligible.

We described AD-APT, a system that strikes a balance between using mobile ads to increase app revenue and dismissing altogether mobile advertising as too disruptive of user experience, therefore avoiding the associated network and energy overheads.

AD-APT automatically re-factors ad-supported Android apps to allow them to reduce the frequency of mobile ad occurrences dynamically. It currently supports a predefined set of basic policies to govern the decision-making process regarding when to load the next mobile ad, but also offers a framework for specifying more complex policies, which account for combinations of contextual information such as the remaining battery life, the type of network connectivity and the network usage caused by mobile ads. Using ten actively

maintained Android apps, we demonstrated experimentally the potential of AD-APT as a practical approach to reducing adaptively both the network and energy overheads of mobile advertising in today's ad-supported mobile apps.

AD-APT was the first system that was created with the purpose of increasing user experience as well as the revenue of the developer by taking into consideration device factors. ADDICTED as the new improved system will include everything that was implemented in AD-APT and new policies that will take into consideration the human factor. To create the new human centered policies, we found, reviewed and selected publications that will enable us to design and implement valid and accurate algorithms to recognize emotional and physical context.

4.2 ADDICTED Algorithm Design

Physical State Recognition Algorithm - PSR

This algorithm was designed based on the algorithm proposed in the **"Toward Physical Activity Diary: Motion Recognition Using Simple Acceleration Features with Mobile Phones"** publication that I reviewed in Chapter 3. PSR relies heavily on the algorithm of the paper but with some changes. One thing that we removed from the original algorithm is the jittering part of the algorithm because newer devices don't have any significant jittering. Another thing we changed in our design was the size of data that we gather to determine the physical context. We decided to collect 100 raw readings from the accelerometer of the wearable device and then parse them into the algorithm to decide the physical context. To gather 100 readings from the accelerometer of the wearable device we need approximately 1 second depending on the movement of the hand and the body in general. The last thing we

changed is the decision tree. We decided to remove the Cycling and Driving options because mostly the users will use the handheld device if they sit, stand, walk or run.

By removing those options, we had to create a new simpler decision tree (Figure 7).

To parse all the readings from the wearable device to the handheld device we append all the

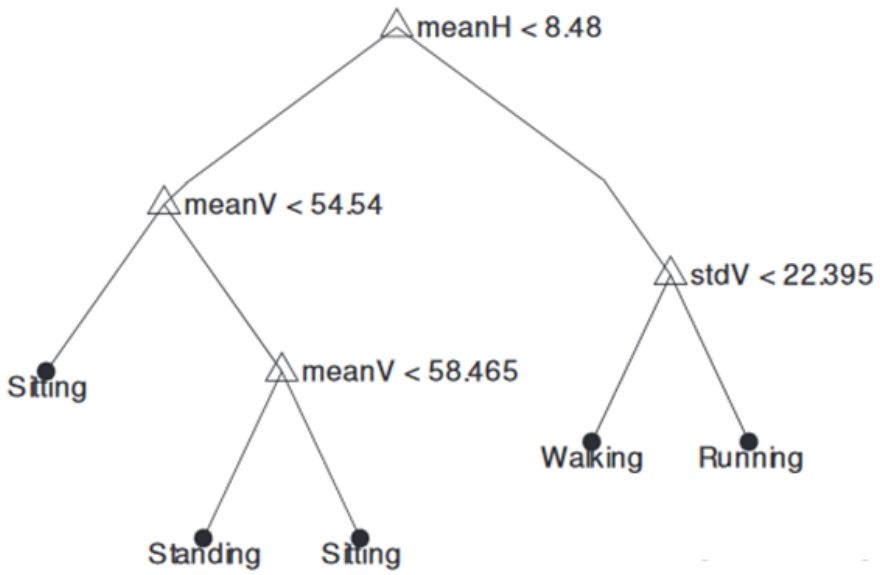

Figure 7. Refined Decision Tree for PSR.

readings of the triaxial accelerometer sensor into a String with specific delimiters so that we broadcast all the readings once and have a way to split them later.

E.g. The triaxial accel. sensor gives the following readings:

x-axis_value_1 = 2

y-axis_value_1 = 3

z-axis_value_1 = 7

x-axis_value_2= -4

y-axis_value_2 = 8

z-axis_value_2 = 9

We append to the string the following: 2:3:7#-4:8:9#...

The ":" delimiter is used to distinguish the x, y and z values and the "#" delimiter is to distinguish each reading from the previous.

After the final String with all the readings is broadcasted to the device then the algorithm starts to work.

How it works:

- Firstly, it creates a Vertical acceleration vector: $\bar{U} = (mx, my, mz)$ where mx, my, mz are means of respective axes.
- Then $\bar{U}$ is normalized to $\bar{U}norm$.
- Let $\bar{a}i = (Xi, Yi, Zi), i = 1, 2, \ldots, N$ be the vector at a given point, where N is the length of sample points.

  The projection of $\bar{a}i$ onto the vertical axis $\bar{U}norm$ can be computed as the vertical component inside $\bar{a}i$.
- Let Pi^{in} be the inner product and pi be the projection vector i.e.

  $Pi^{in} = < \bar{a}i, \bar{U}norm >$, $pi = Pi^{in} * \bar{U}norm$ then the horizontal component hi of the accel. vector $\bar{a}i$ can be computed as vector subtraction i.e. $hi = \bar{a}i - pi$
- Then take the magnitude of hi, denoted by $||hi||$ as a measure of horizontal movement.

The results generate two wave forms of $\{Pi^{in}, i = 1,2, \ldots, N\}$ and $\{||hi||, i = 1,2, \ldots, N\}$ which are amplitude of vertical components and magnitude of horizontal components.

From those two tables, we can calculate the mean value of the horizontal and vertical movement as well as the standard deviation (meanV, meanH, stdV, stdH).

The mean values and standard deviation values will determine at the end the physical state of the user based on the refined decision tree we discussed before (Figure 7).

The values of the decision tree might need some adjusting depending on the sensitivity of the sensor of the wearable device. We decided to categorize the results in two major categories, Moving and Not Moving with both categories having 2 subcategories each. The

subcategories are: Walking and Running for Moving category and Sitting and Standing for Not Moving category.

Emotional State Recognition Algorithm – ESR

This algorithm was designed based on the algorithm proposed in the **"Mathematical Algorithm for Heart Rate Variability Analysis"** publication that I reviewed in Chapter 3. As the PSR algorithm, ESR algorithm also relies heavily on the proposed algorithm but this time with minor changes to the original algorithm. Instead of a 10-15 second gathering of raw data from the heart rate sensor we decided to collect data for approximately 20 seconds basically until we have 20 heart beats. Another thing we wanted to change is the decision tree. Basically, the algorithm only decided if the user is stressed or not based on a Central Index -CI. If CI >50% then the user is relaxed else the user is stressed. We wanted to add some more granularity to this by adding two more outcomes. So, we ended up we four outcomes; Very Stressed, Stressed, Relaxed, Very Relaxed (Figure 8).

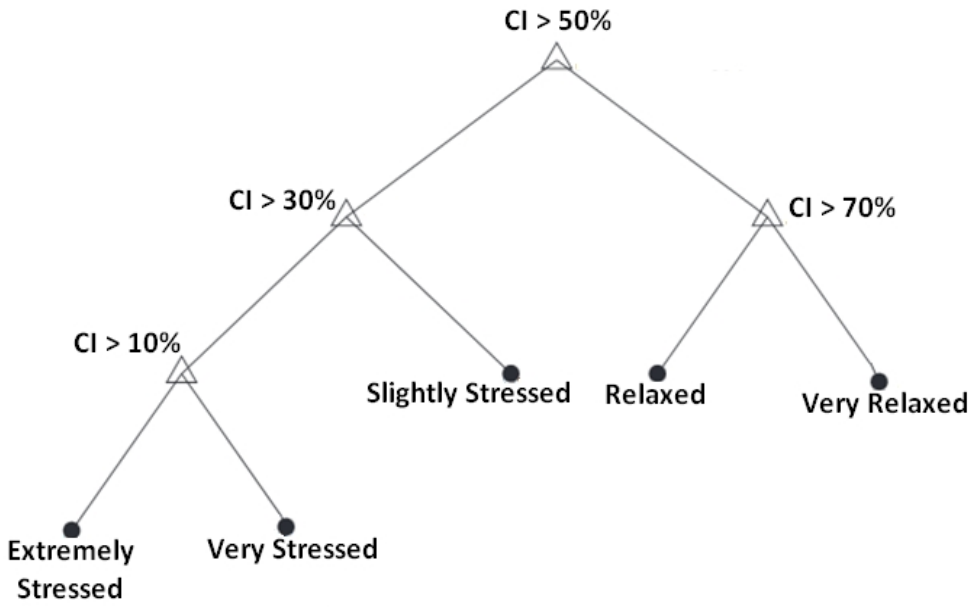

Figure 8. Refined decision tree for ESR

We used the same approach at parsing as we used in PSR algorithm. We collect 20 heart beats from the heart rate sensor of the wearable device, parse them all together into a String and then broadcast the string to the handheld device to be processed by the ESR algorithm.

4.3 ADDICTED Policies Design

After the implementation of the two main algorithms, we had to come up with a set of policies that would satisfy all possible combinations of emotional and physical states. We decided that our policies will take into consideration the type of movement while moving or the type of stance while not moving so the possible outcomes are sixteen. (Table 2).

Physical State	Emotional State	Type of ad
Not Moving (NM) - Sitting (SIT)	Very Stressed (VS)	Banner (B)
Not Moving (NM) - Standing (ST)	Stressed (S)	Interstitial Image (II)
Moving (M) - Walking (W)	Relaxed ®	Interstitial Video (IV)
Moving (M) - Running (R)	Very Relaxed (VR)	

Table 1. Possible outcomes of PSR, ESR and Types of ads

As stated before ADDICTED will dynamically adjust the type of ads that will be shown to the user based on Physical and Emotional context. By doing this we ensure user satisfaction and enforce positive user experience. Additionally, we decided to adjust the frequency of the ads as well based on the outcomes of the algorithms. By doing that we increase battery life and reduce internet usage regarding ads (video and image downloading etc.). By combining the two - frequency reduction and ad type adjusting, we increase the time that the user spends using the app and therefore increase the revenue of the developer [4].

Ranking Status	Ranking Ad Type	Ranking Ad Frequency	Justification
M-R & VS	**NO AD**	**NO AD**	**WORST**
M-R & S	**NO AD**	**NO AD**	**WORST**
M-R & R	BANNER	20%	ACTIVITY > EMOTION
M-R & VR	BANNER	40%	ACTIVITY > EMOTION
M-W & VS	BANNER	60%	ACTIVITY > EMOTION
M-W & S	BANNER	80%	ACTIVITY > EMOTION
M-W & R	BANNER	100%	ACTIVITY > EMOTION
M-W & VR	BANNER	100%	ACTIVITY > EMOTION
NM-ST & VS	B & II	20%	EMOTION > ACTIVITY
NM-SIT & VS	B & II	40%	EMOTION > ACTIVITY
NM-ST & S	B & II	60%	EMOTION > ACTIVITY
NM-SIT & S	B & II	80%	EMOTION > ACTIVITY
NM-ST & R	B & II & IV	60%	EMOTION > ACTIVITY
NM-SIT & R	B & II & IV	80%	EMOTION > ACTIVITY
NM-ST & VR	**B & II & IV**	**100%**	**BEST**
NM-SIT & VR	**B & II & IV**	**100%**	**BEST**

Table 2. Rates of ads based on the rank of the outcome of both algorithms.

We decided that our policies will have the above rates regarding the type of the ad that will be loaded, by taking into consideration various scenarios. To decide the type of ads and the frequency we first created a ranking status for all the combinations of the results of PSR and ESR algorithms. We consider the **Moving – Running & Very Stressed** and **Moving - Running & Stressed** as the two worst possible outcomes and **Not Moving – Standing & Very Relaxed** and **Not Moving – Sitting & Very Relaxed** as the best outcomes. All other outcomes in between were ranked accordingly. After deciding the global ranking system, we split the global ranking into two sub ranking system. In the first sub ranking system, the PSR

result has more gravity than the ESR result so the ranks were adjusted based on the movement e.g. Running & ESR result outcome is considered worse than the Walking & ESR result outcome. In the second sub ranking system, the result of the ESR algorithm has more gravity than the result of the PSR algorithm because in any of the two cases the user is not moving therefore it is logically correct to take into greater consideration the result of the ESR algorithm instead the result of the PSR algorithm.

4.4 Tools and devices used

Development tools - IDEs

For the development of our applications we've decided to work with Android Studio which is the Official IDE for Android. Based on Intellij IDEA, Android Studio provided the fastest possible turnaround on our coding and running workflow.

Android Studio provides a plethora of features that were ideal for developing our apps. The fast and feature-rich emulators with the combination of the Instant Run feature were essential for the app development. Another great feature that was ideal for our case was that we could develop apps for all kinds of android devices, in our case handheld device and wearable device [5].

Testing Devices

For the development of ADDICTED and for the testing we used both handheld devices and wearable devices. For the handheld device, we used a Samsung Galaxy S5 SM-G900F. We chose this device because according to a report from Kantar World Panel [6], Samsung's most popular device in the U.S.A. remains the Galaxy S5, surprisingly. As of February 2017, the Galaxy S5 represents 15.6% of active Samsung smartphones within the U.S. This is followed by the Galaxy S7 at 11.5% and the Galaxy S6 at 11.4%. The Galaxy S7 Edge accounted for 5.8%. Furthermore, the device operates on the Android KitKat 4.4.2. By targeting 4.4.2 and later versions, our app can run on approximately 89.1% of the devices that are active on the Google Play Store (Table 3).

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.00%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.80%
4.1.x	Jelly Bean	16	3.20%
4.2.x		17	4.60%
4.3		18	1.30%
4.4	KitKat	19	18.80%
5	Lollipop	21	8.70%
5.1		22	23.30%
6	Marshmallow	23	31.20%
7	Nougat	24	6.60%
7.1		25	0.50%

Table 3. Android API Distribution.
Data collected during a 7-day period ending on May 2, 2017 [7].

Samsung Galaxy S5 SM-G900F Specs [8]:

- GSM / 4G LTE Capable
- International Variant/International LTE
- 16MP 1/2.6" Rear Camera Sensor
- Phase-Detection Autofocus + UHD 4K Video
- GHz Snapdragon 801 Quad-Core CPU
- 16GB Storage Capacity + 2GB of RAM
- 5.1" Super AMOLED Display
- Full HD 1920 x 1080 Resolution
- Fingerprint Scanner + Heart Rate Sensor
- Android KitKat 4.4.2

For wearable devices, we had a broad range of devices to choose from. We decided to work on the Motorola Moto 360 2nd Gen. It was one of the best-selling smart watches when it

launched and it is rated amongst the Top 5 smart watches of 2017 [9]. Moto 360 2nd Gen runs on Android Wear OS and it is compatible with both Android and iPhone devices. (Requires a phone running Android 4.3+ or iPhone 5, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus with iOS 8.2+. Features may vary between Android and iOS platforms. (check compatibility at g.co/WearCheck) We chose this wearable device because it has exceptional performance with its Qualcomm® SnapdragonTM 400 with 1.2 GHz quad-core CPU (APQ 8026) Adreno 305 with 450MHz GPU and has all the sensors that we needed for our app. It has Accelerometer, Ambient Light Sensor, Gyroscope, Vibration/Haptics engine and an Optical heart rate monitor (PPG). Also, it can connect via Bluetooth® 4.0 Low Energy or Wi-Fi 802.11 b/g.

Motorola Moto 360 2nd Gen Specs

LAUNCH	**Announced**	2015, September
	Status	Available. Released 2015, September
BODY	**Dimensions**	46 x 46 x 11.4 mm (1.81 x 1.81 x 0.45 in)
	Build	Stainless Steel
	SIM	No
		- IP67 certified - dust and water resistant
		- Water resistant up to 1 meter and 30 minutes
		- Compatible with standard 22mm straps
	Type	IPS LCD capacitive touchscreen, 16M colors
	Size	1.56 inches (~66.7% screen-to-body ratio)
	Resolution	360 x 330 pixels (~233 ppi pixel density)
	Multitouch	Yes
	Protection	Corning Gorilla Glass 3
PLATFORM	**OS**	Android Wear, planned upgrade to 2.0

	Chipset	Qualcomm MSM8926 Snapdragon 400
	CPU	Quad-core 1.2 GHz Cortex-A7
	GPU	Adreno 305
MEMORY	**Card slot**	No
	Internal	4 GB, 512 MB RAM
CAMERA		No
SOUND	**Alert types**	Vibration; MP3, WAV ringtones
	Loudspeaker	No
	3.5mm jack	No
		- Active noise cancellation with dedicated mic
COMMS	**WLAN**	Wi-Fi 802.11 b/g
	Bluetooth	4.0, LE
	GPS	No
	Radio	No
	USB	No
FEATURES	**Sensors**	Accelerometer, gyro, heart rate
	Messaging	SMS, Email, IM
	Browser	No
	Java	No
		- Qi wireless charging
		- MP3 player
		- Photo viewer
		- Voice dial/commands
		Non-removable Li-Ion 400 mAh battery
	Talk time	Up to 48 h (mixed usage)

Table 4. Motorola Moto 360 2nd Gen. Specs

Both of the above devices were used during the development of the wearable device app as well as the handheld device app.

4.5 Testing app development

To test our algorithms and our policies we had to create several versions of testing applications. We first of started by developing a basic naive wear device application which

its sole purpose was to read the data from the sensors of the wearable device and display that data on the screen of the same device. At that point, the TestAppV1.0 only gathered the data

Figure 9. TestAppV1.0 Accelerometer data on wearable device

from the heart rate sensor and the accelerometer sensor and displayed the data on the wearable screen. (Figure 9)

In next version of the app, TestAppV1.1 (Figure 10), we managed to broadcast the data from the wearable device to the handheld device with the help of Listeners from the Message API

Figure 10. TestAppV1.1 Accelerometer readings on handheld device

provided and display them on the screen of the handheld device. (More details in Chapter 4, Section 4.2)

The next version of the app, TestAppV2.0 included the PSR algorithm. We developed the PSR algorithm first because it was easier to test, calibrate and evaluate the results. In this version, the app displayed the accelerometer readings and the result of the PSR algorithm. (Figure 11)

Figure 11. TestAppV2.0 Accelerometer Data with PSR result

In the next version of the app, TestAppV2.1 we refined the PSR algorithm and added more granularity to the results. The results of the algorithm now include Moving (Running) , Moving (Walking), Not Moving (Standing), Not Moving (Sitting) and Calculating (Transitioning). (Figure 12)

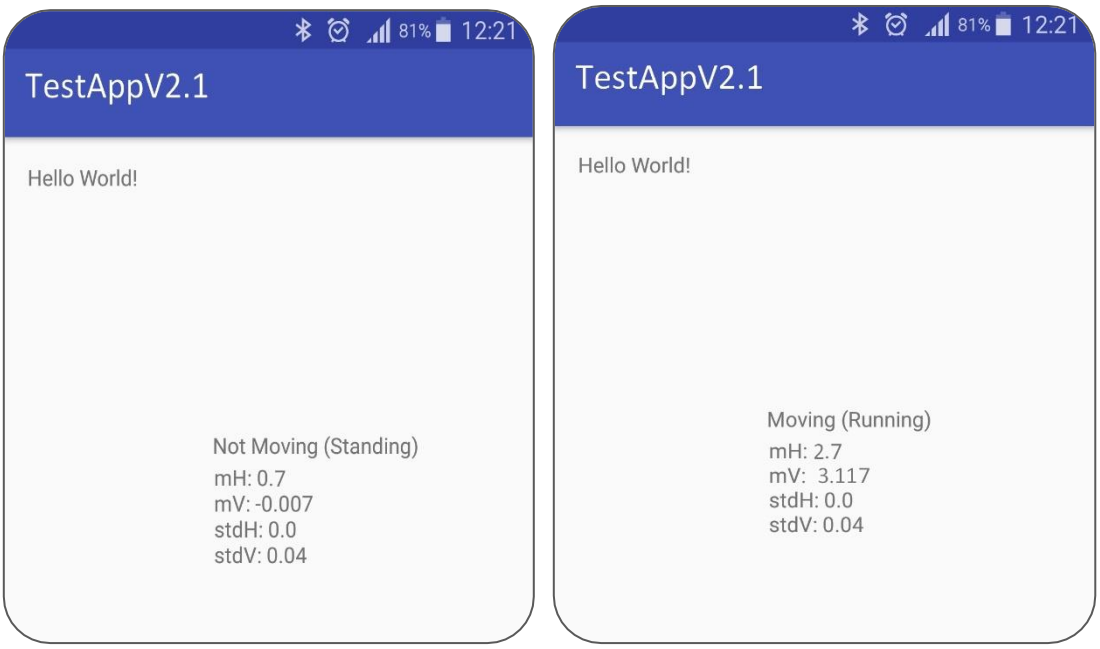

Figure 12. TestAppV2.1 Accelerometer data with refined PSR results.

In TestAppV3.0 we developed and include in the app the ESR algorithm. The ESR algorithm recognizes four states, Very Stressed, Stressed, Relaxed and Very Relaxed. The current version displays both the Emotional State and the Physical State of the user. (Figure 13)

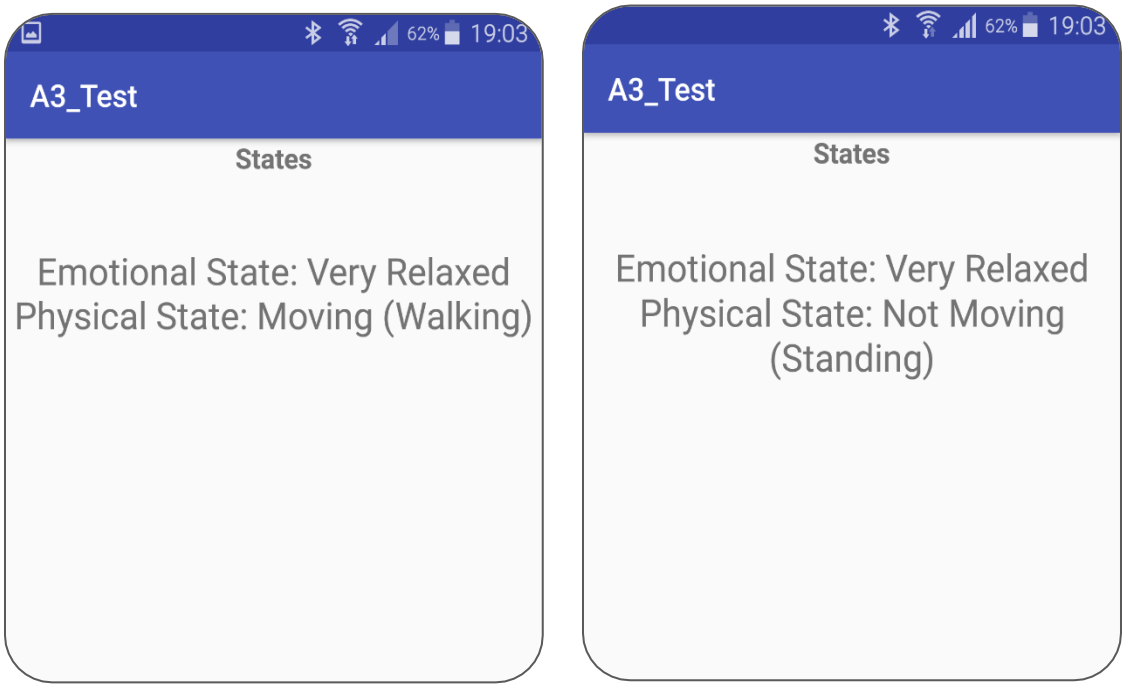

Figure 13. TestAppV3.0 PSR and ESR results

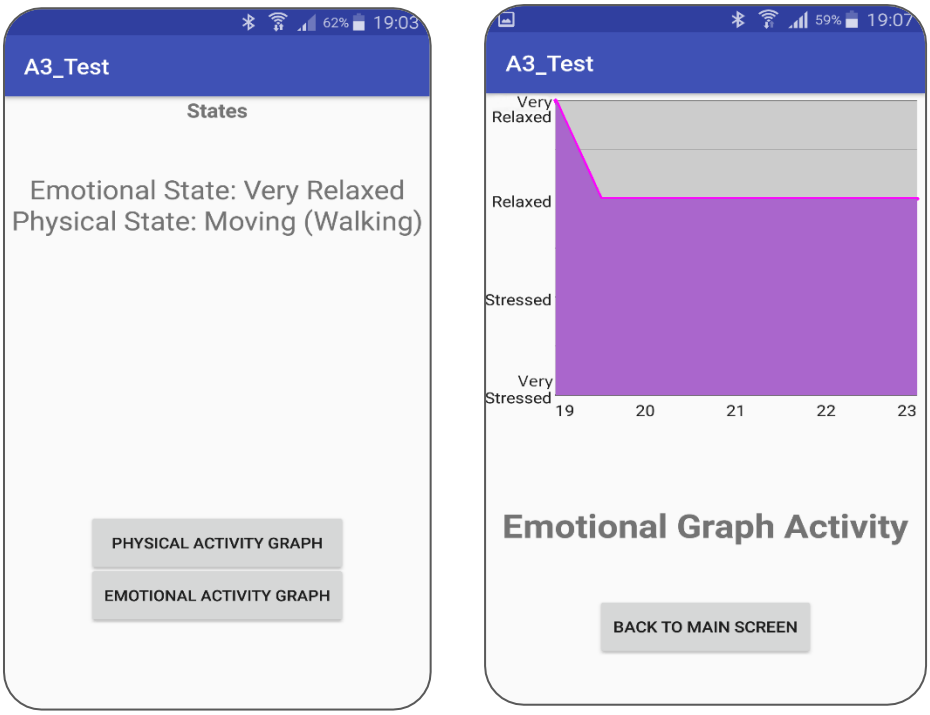

Figure 14. TestAppV3.1Real-time graphs for ESR and PSR results

After refining the two algorithms we incorporated two real-time graphs for the two algorithms in the newer version of the app, TestAppV3.1. (Figure 14 & Figure 15)

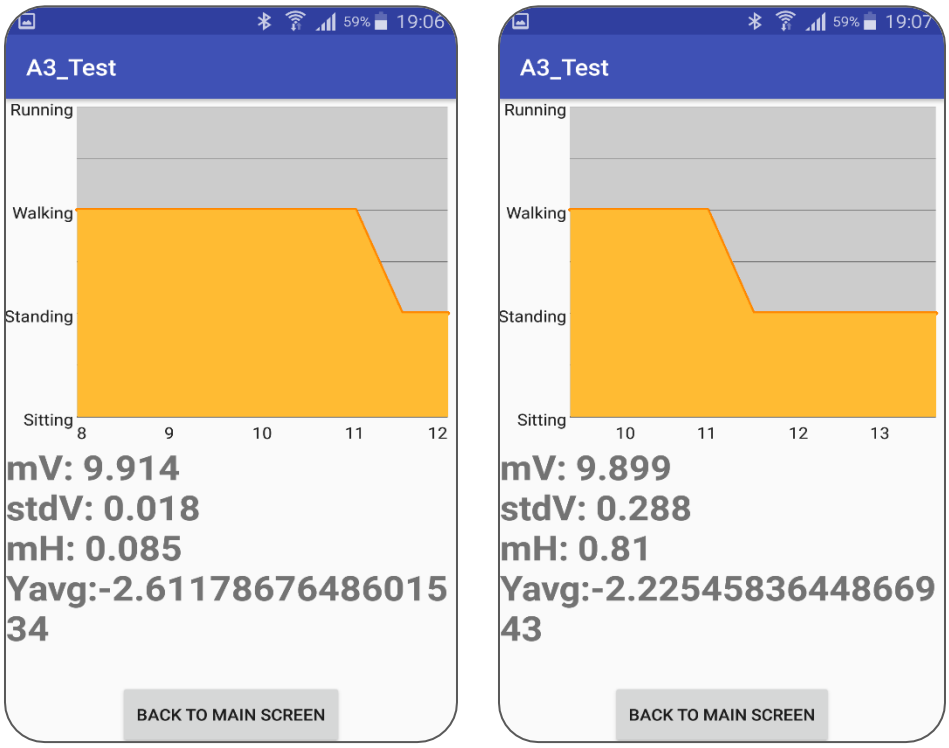

Figure 15. TestAppV3.1Real-time graphs for ESR and PSR results

In the last version before the testing, TestAppV3.2 we added a data recording and a feedback option. When the user presses the “Record Data” button, the app automatically records and saves locally both the PSR and ESR results for 5 minutes. After the recording finishes, the user is prompted to the feedback activity where he can give us a feedback stating his emotional and physical state. The recorded data in relation with the feedback of the user helped us evaluate the accuracy of our algorithms. (Figure 16)

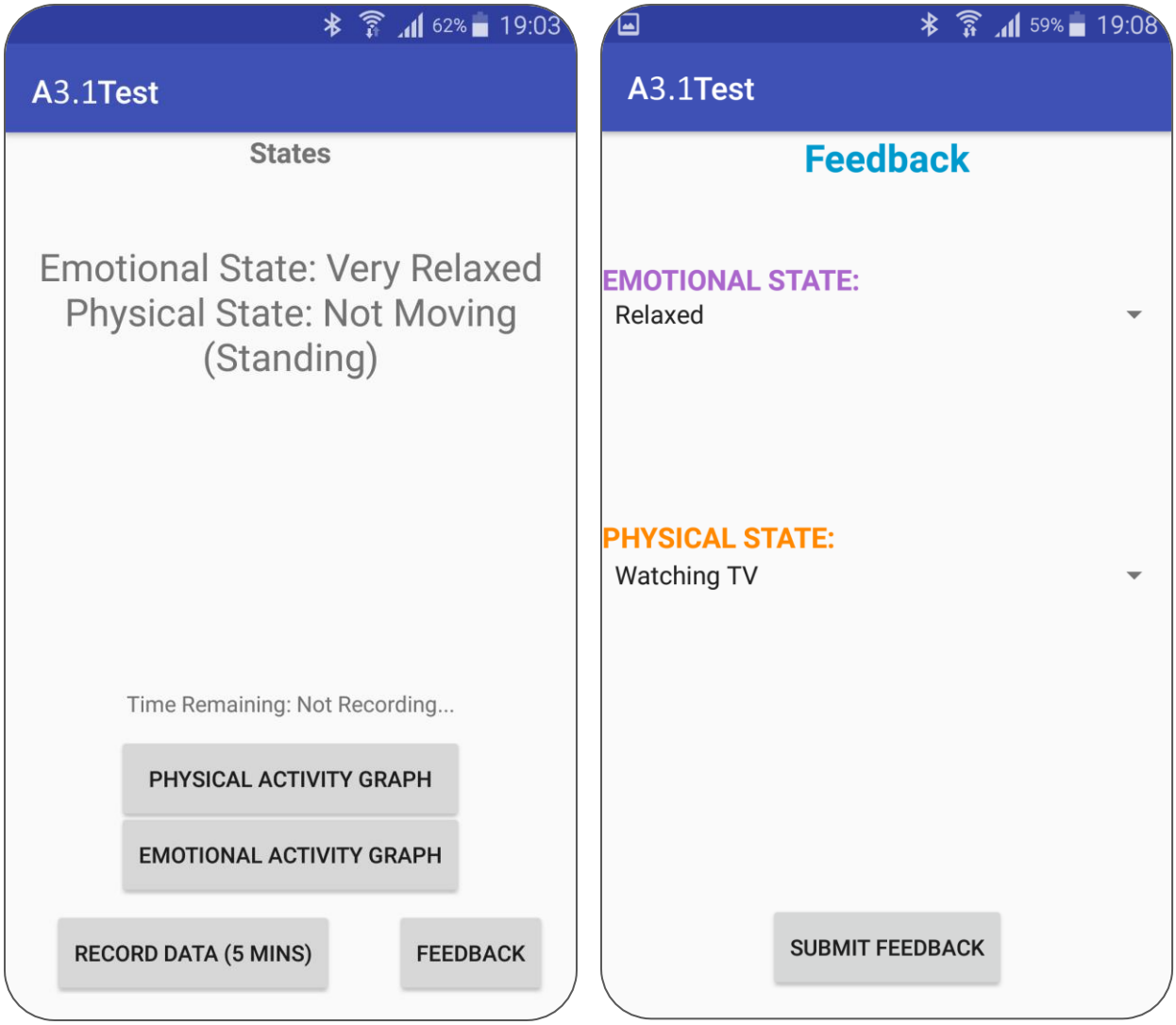

Figure 16. TestAppV3.2 Data recording and feedback feature added.

4.6 Custom Facebook application development

After the development of our policies and the implementation of the TestApp, we started the development of a demo application that would have our policies included. That application will be given to users for a long-term study to evaluate if our policies increase the user satisfaction by dynamically adapting ads.

Having in mind that the application must be used every day for the long-term study to be valid, we created a pool of possible candidate apps that we would implement and include our policies. We decided to create an application that will host the Facebook mobile site. We decided to go with Facebook because it is one of the most popular social media networks and has over 1.5 billion monthly active users worldwide [23]. Thus, by using Facebook our users will not be forced to use a new application but continue to interact with the social network with the difference that they will interact with it via our custom application. To create this custom application, we created a sort of a wrapper app that connected to the original Facebook site via a browser but for mobile devices (https://mobile.facebook.com).

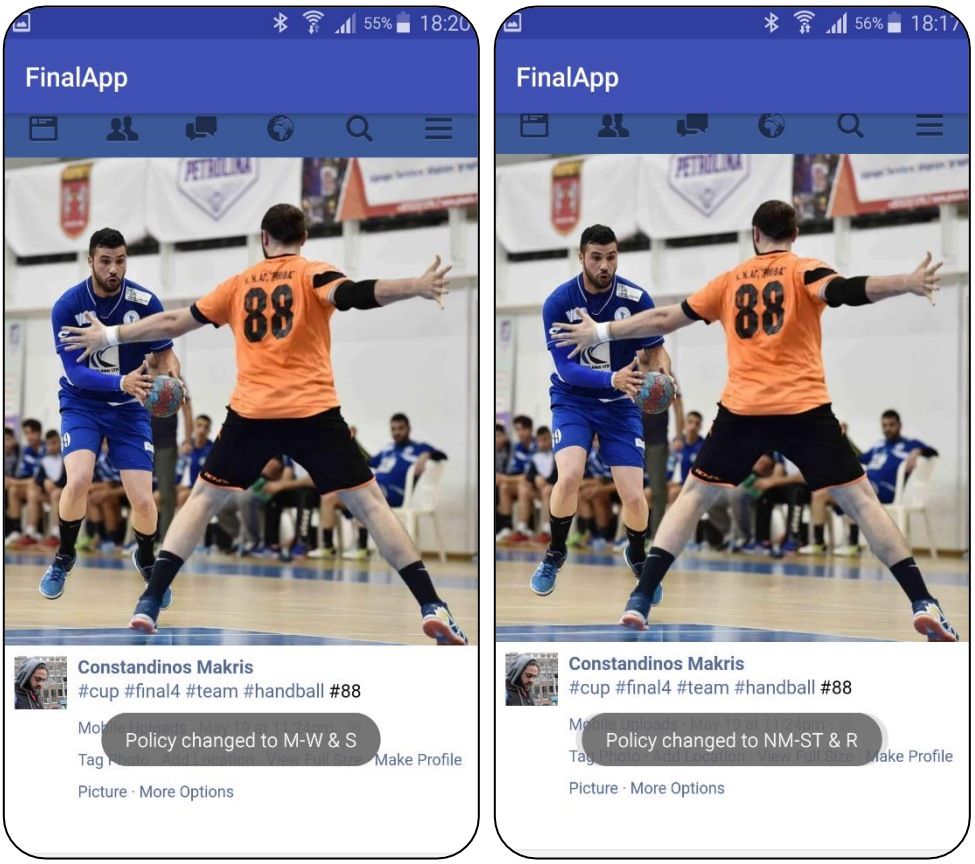

Figure 17. Facebook app. Hidden Ads in different policies

After creating the application, we added the code behind that connects the app to the wearable device and we also added the algorithms that will decide the physical and emotional state of the user. After testing that the handheld app can connect with the wearable device we added the policies into the code that will show or hide the designated ads (see Chapter 4, Section 4.3) (Figures 17,18).

Finally, we performed a laboratory task-oriented test to examine whether our algorithms and policies function correctly.

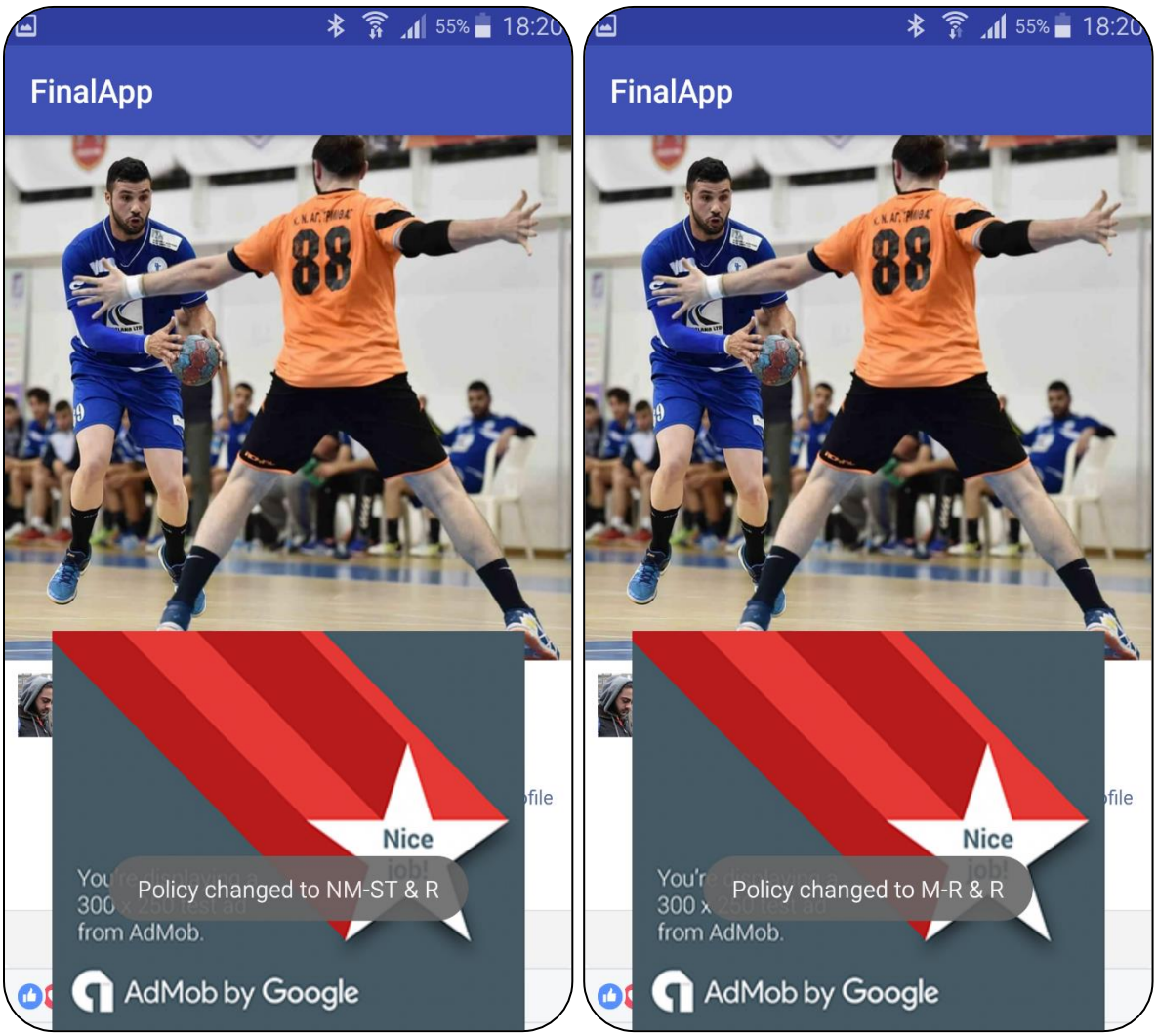

Figure 18. Facebook app. Visible ads in different policies

Chapter 5

Evaluation

5.1 Method of study

5.2 Analysis of results and interpretations

5.1 Method of study

A user study was conducted with the aim to initially evaluate the accuracy of the PSR and ESR algorithms.

Sampling and procedure

In the user study, a total of 15 participants (8 male, 7 female, age 20-50), were asked to use our test app for one day and record data for 5 minutes every hour while doing their everyday tasks. Wearable devices were given to the participants and the test app was then installed onto their own handheld devices. The wearable device that was used during the user study was a Moto 360 2nd Gen. (Chapter 4, Section 4.4).

The participants only had to run the app on their handheld device and on the wearable device and press the "Record Data" Button. After the end of the 5 minutes the users were prompted to submit a feedback to state their current physical task that they were doing and a self-estimation of their emotional state.

The goal was to test our algorithms without the user doing instructed movements or tasks. We wanted to test the accuracy in real life situations and non-instructed movements. At the end of the day we retrieved the data files that were saved on their handheld devices for analysis.

5.2 Analysis of results and interpretations

To analyze the data, we first had to retrieve the data files from the handheld devices that contained the raw data recorded from the accelerometer and heart rate sensors of the wearable device. After retrieving the files, we had to manipulate the raw data in a way that enables us to compare the results of the algorithm with the feedback that the user provided.
(Table 5 & 6)

X	Y	Z	X	Y	Z
1.5753847	1.4652514	8.944737	1.2593501	1.364695	10.041282
1.4221559	1.3263878	9.14585	0.9768343	1.3503298	9.797073
1.2162545	1.5753847	9.313444	1.0965444	1.5227122	9.653421
1.2354081	1.8483237	9.380483	0.80445176	1.6903064	9.840169
1.340753	1.8148049	9.251195	0.60812724	1.7669208	9.873688
1.3455414	2.1260512	8.901642	0.866701	1.8483237	9.294291
1.3215994	1.8531121	9.241618	0.89543146	1.5227122	9.557653
1.3215994	1.8531121	9.241618	0.51235914	1.685518	9.706094
1.3694834	2.1883004	9.227253	0.58897364	1.8435353	9.854534
1.4652514	2.207454	9.227253	0.6368576	1.7669208	9.581595
1.1348516	2.1499932	9.691729	0.59376204	1.436521	9.514558
1.364695	2.15957	10.103531	0.7038953	1.5179238	9.662998
1.6232687	2.0207062	9.89763	0.6991069	1.3359646	9.998186
1.6424223	1.6567876	9.337387	0.7565677	1.6998832	9.586384
1.6950948	1.3215994	9.030929	0.866701	1.5179238	9.390059
1.5131354	1.0390835	9.26556	0.89543146	1.460463	9.524135
1.2162545	1.283292	9.437943	0.92895025	1.5131354	9.390059
1.2162545	1.2114661	9.251195	1.0773908	1.7046716	9.260772
1.1683705	1.6519991	9.591172	1.1252748	1.685518	9.241618
1.1348516	2.1691468	9.595961	1.2306197	1.8387469	9.490616
1.4556746	2.3511062	9.65821	1.3311762	2.078167	9.677363
1.5753847	2.0350714	10.271125	1.4748282	2.4277205	9.500193
1.6472107	2.4277205	10.208876	1.5897499	2.3223755	9.504981
2.0398598	2.0446482	9.524135	1.5275006	2.1068976	9.9550905
1.7621324	1.4556746	9.409213	1.7860745	2.0207062	9.883265
1.5753847	0.7661445	9.615114	1.8435353	2.0398598	9.241618

1.4939818	1.0486604	9.595961	1.5610195	1.3311762	9.744401
1.5658079	1.4556746	9.074024	1.5227122	1.6759412	9.595961
1.3455414	1.5227122	9.227253	1.3694834	1.7142484	9.222465
1.1923125	1.733402	9.485827	1.3120226	1.7046716	8.911219
1.1683705	1.8722657	9.414001	1.1252748	1.6376339	9.078813
1.2785037	1.934515	9.466674	0.9768343	1.7764976	9.323022
1.2785037	2.1404164	8.987833	0.94331545	1.9967642	8.987833
1.3215994	2.1021092	9.251195	0.9528923	1.9440918	9.14585
1.340753	1.6519991	9.677363	0.842759	1.7764976	9.710882
1.1587936	2.0685902	10.180145	0.9768343	2.0973208	9.825804
1.3934253	1.9393034	9.68694	1.4269443	2.255338	9.672575
1.6328455	1.7142484	8.7053175	1.6232687	2.078167	9.399636
1.6376339	1.0390835	8.63828	1.6998832	1.8531121	9.126697
1.5131354	1.5035586	9.308656	1.8962077	1.3311762	9.016563
1.4269443	1.5179238	9.715671	1.7812861	1.4891934	9.476251
1.3311762	2.0254946	9.375694	1.7142484	1.6184803	8.90643
1.0390835	1.5849615	8.992621	1.4077905	1.5370775	9.212888
0.90979666	1.460463	9.557653	1.0773908	1.5370775	9.619903
0.81881696	1.6567876	9.959879	1.2497733	1.8866309	9.260772
1.163582	2.4708161	9.639056	1.1252748	1.9488802	9.021352
1.283292	2.1883004	9.255983	1.0151415	1.7621324	9.734824
1.2066777	2.0159178	9.356541	0.9768343	1.8291701	9.835381
1.2306197	1.3934253	10.156203	1.2785037	2.0015526	9.299079
1.2354081	1.3838485	10.132261	1.2593501	1.7238252	9.476251
Not Moving (Standing)					

Table 5. Raw data from wearable accelerometer and PSR result

Heart Beat	82	84	86	86	87	91	94	95	97	98
	100	101	102	103	103	104	105	104	105	104
ESR result	Relaxed, 72%									

Table 6. Raw data from heart rate sensor of wearable device and ESR result.

The above data is the raw data collected from the wearable device and saved on the handheld device. Approximately, 3-5 data files were saved for the emotional context containing the raw heart beats and the ESR result and 5-50 data files were saved for the physical context. The number of the data files regarding the physical context may vary depending on the movement of the user. A data file is created whenever the physical state of the user changes.

E.g. if the user is sitting on the couch watching TV the physical state of the user is always "Not Moving – Sitting". In this case only one data file will be created for the entire duration of the recording. In the case of a user walking or running, many more data files would be created because the physical state of the user may vary from "Moving-Running" to "Calculating – Transitioning…" to "Moving-Walking". Thus, many different data files will be created every time the new physical state is different from the previous. We chose to have this approach regarding the data files because it was easier for us later to analyze the data. Based on the previous statement, a total of 36-60 ESR data files and 60-600 PSR data files were saved for every user that recorded data every hour for 10 hours.

Data Processing

The results of the ESR and PSR algorithms of every recording were then inserted into a spreadsheet along with the feedback of the user. (Table 7)

Makris C.	11:54:42	
User Feedback	**ESR Result**	**PSR Results**
Relaxed , Walking	Relaxed, 61%	Not Moving (Standing)
		Calculating...(Transitioning)
		Not Moving (Standing)
		Calculating...(Transitioning)
		Not Moving (Standing)
		Calculating...(Transitioning)
		Not Moving (Standing)
		Calculating...(Transitioning)
		Moving (Walking)
		Calculating...(Transitioning)
Makris C.	12:23:17	
User Feedback	**ESR Result**	**PSR Results**
Relaxed , Driving	Stressed, 39%	Not Moving (Standing)
		Calculating...(Transitioning)
		Not Moving (Sitting)
		Calculating...(Transitioning)
		Not Moving (Standing)

Table 7. First processing of the algorithms results.

After the initial processing, the structure of the data was not in an optimal way that would enable us to plot graphs or analyze even further the data so we could evaluate the accuracy of our algorithms. A second data processing occurred were all the data recorded per user were inserted into a new spreadsheet with a different structure. In the second processing, we decided to insert into the new spreadsheet the starting physical state along with the ending physical state of the user instead of writing down all the states during the 5-minute recording. (Table 8)

User	Emotional Feedback	Physical Feedback	PSR Start State	PSR End State	ESR Result	ESR %
1	Relaxed	Walking	Not Moving (Standing)	Moving (Walking)	Relaxed	61%
	Relaxed	Driving	Not Moving (Standing)	Not Moving (Sitting)	Stressed	39%
	Stressed	Sitting	Not Moving (Sitting)	Not Moving (Sitting)	Stressed	28%
	Relaxed	Sitting	Not Moving (Standing)	Not Moving (Sitting)	Relaxed	67%
	Stressed	Sitting	Not Moving (Sitting)	Not Moving (Standing)	Stressed	28%
	Relaxed	Watching TV	Not Moving (Sitting)	Not Moving (Sitting)	Relaxed	50%
	Relaxed	Walking	Moving (Walking)	Moving (Walking)	Relaxed	72%
	Relaxed	Walking	Not Moving (Standing)	Moving (Walking)	Relaxed	56%
	Relaxed	Standing	Not Moving (Standing)	Not Moving (Standing)	Relaxed	72%

Table 8. Final structure of data recorded per user.

Finally, to evaluate the accuracy of our algorithms we had to compare the feedback of the user with the results of our algorithms. By plotting onto the same graph the feedback and the result of all the recorded data of a user we are able to see if the algorithm predicted the physical or emotional state correctly by examining if both graphs pass through the same points. (Figures 18 - 20)

Figure 19. Comparison of ESR result and User's emotional feedback

In the above graph (Figure 19), you can see the comparison of the ESR algorithm result with the Emotional State feedback of the user. In this case, the results of our algorithm are the same with the feedback of the user in most of the recordings. The two graphs overlap each other which means the ESR algorithm is accurate in most of the cases. Based on literature [10], for accurate results, stress level calculations should be recorded with a time interval of 1-6 minutes. That means any subsequent calculation before an interval of 1 minute may be inaccurate or invalid. As a result, to increase our accuracy, we decided that our algorithm will return three consecutive estimations of the emotional state of the user. For our algorithm to give a valid result, 20 seconds of consecutive heart rate reading must pass. Thus, in one minute we can have three calculations of the emotional state. For accuracy reasons, we decided to select as the most valid and accurate the second result of our algorithm. The main reason of this decision is the fact that many times the first heart beats of the user are not

recorded correctly due to the fact that the wearable device might not be fitted correctly on the wrist and as a result the first result of the ESR algorithm may not be valid.

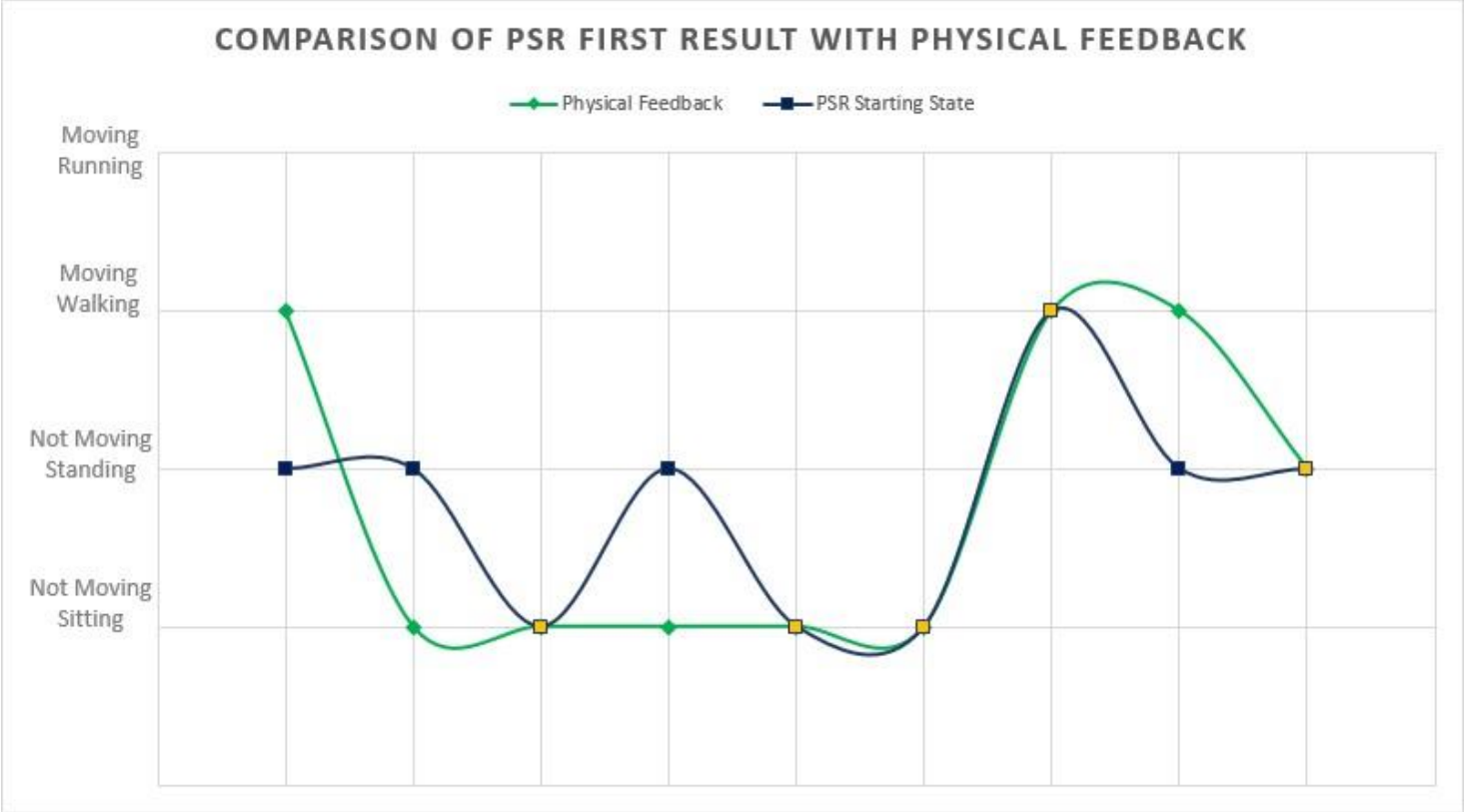

Figure 20. Comparison of PSR first result with the Physical feedback of the user

In the above graph (Figure 20), you can see the comparison of the PSR algorithm result with the physical feedback of the user. The approach is the same as the comparison of ESR result and the emotional feedback of the user. The difference is that we compare the first result of the PSR algorithm with the feedback of the user that was submitted after the 5-minute recording. Thus, there are many differences in the two graphs as we can see in the above figure. This was an expected outcome because the PSR algorithm calculates the physical context of the user at real time, thus, the last result of the algorithm is the most accurate in comparison with the user's feedback. This can be seen in the next graph (Figure 21) where we compare the physical feedback of the user with the last result of the PSR algorithm during the recording. We can see that now the two graphs have much more similarities than the graphs in the previous figure.

Figure 21. Comparison of PSR last result with the Physical feedback of the user

Interpretation of results

The analysis of results allows us to draw conclusions in regards to the accuracy of our algorithms.

ESR algorithm result interpretation

Cohen's κ was run to determine if there was agreement between two stress elicitation mechanisms (user's feedback and implicit extraction) on whether the corresponding user was feeling relaxed or stressed. Among 80 user sessions, the two elicitation mechanisms agreed on 46 sessions when the user was relaxed and 14 sessions when the user was stressed. However, the user feedback mechanism recorded 19 sessions as relaxed when the implicit mechanism recorded them as stressed, and the user feedback mechanism recorded 1 session as stressed when the implicit mechanism recorded it as relaxed.

There was moderate agreement between the two elicitation mechanisms, $\kappa = .439$ (95% CI, .256 to .621), $p < .0005$. (Table 9)

Count

		Implicit		
		Relaxed	Stressed	Total
Feedback	Relaxed	46	19	65
	Stressed	1	14	15
Total		47	33	80

*Table 9. Feedback * Implicit Crosstabulation*

We decided to evaluate the accuracy with only stressed and relaxed feedbacks because there wasn't a significant number of "very stressed" or "very relaxed" responses to evaluate the accuracy of all four states. There were some cases where the feedback of the user stated a different state than our algorithm. In some cases, self-evaluation of the emotional state is hard to be precise.

E.g. A user submits feedback stating that he was Stressed but the ESR algorithm gives a result of "Relaxed" with a CI of 50%. If the ESR algorithm gave an CI of 49% then the result would be that the user is "Stressed" and the user feedback would agree with the result. Taking that into consideration we can improve the current 75% accuracy of the ESR algorithm by adding a "Neutral State" when the CI is between 45-55%. By doing that we could eliminate any cases like the example above and increase the accuracy of our algorithm.

PSR algorithm result interpretation

a) PSR algorithm data analysis all states:

Cohen's κ was run to determine if there was agreement between two mechanisms for extracting the user's physical activity (based on user's feedback and based on implicit extraction) on whether the corresponding user was sitting, standing, walking or running. Among 80 user sessions, the two elicitation mechanisms agreed on 45 sessions when the user was sitting, on all 9 sessions when the user was standing, on 13 sessions when the user was walking, and on 1 session when the user was running. However, the user feedback mechanism recorded 10 sessions as sitting when the implicit mechanism recorded them as standing, and the user feedback mechanism recorded 2 sessions as walking when the implicit mechanism recorded them as standing.

There was good agreement between the two elicitation mechanisms, $\kappa = .729$ (95% CI, .593 to .864), $p < .0005$.

Count

		Implicit Physical				
		Sitting	Standing	Walking	Running	Total
Feedback	Sitting	45	10	0	0	55
	Standing	0	9	0	0	9
	Walking	0	2	13	0	15
	Running	0	0	0	1	1
Total		45	21	13	1	80

*Table 10. Feedback * Implicit Crosstabulation - all states*

b) PSR algorithm data analysis two states (moving / not moving):

Cohen's κ was run to determine if there was agreement between two mechanisms for extracting the user's physical activity (based on user's feedback and based on implicit extraction) on whether the corresponding user was moving or not. Among 80 user sessions, the two elicitation mechanisms agreed on 14 sessions when the user was moving and on 64

sessions when the user was not moving. The user feedback mechanism recorded only 2 sessions as moving when the implicit mechanism recorded them as not.

There was very good agreement between the two elicitation mechanisms, $\kappa = .918$ (95% CI, .806 to 1.029), $p < .0005$.

Count

		Implicit Physical		
		Moving	Not Moving	Total
Feedback	Moving	14	2	16
	Not Moving	0	64	64
Total		14	66	80

*Table 11. Feedback * Implicit Crosstabulation - two states*

For this algorithm, we decided to evaluate the results in two ways. First, we analyzed the data taking into consideration all four states that the algorithm can predict. The analysis gave us an accuracy of 85% regarding the four states. Finally, we decided to analyze the data taking into consideration only the two primary states, moving or not moving. The analysis gave us an accuracy of 97.5%.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

6.2 Future Work

6.1 Conclusions

The purpose of this dissertation was the design and implementation of a system that increases user experience and revenue for the developer by dynamically adapting the type and frequency of mobile ads. ADDICTED is the improved and enhanced version of AD-APT, an already existing system that we developed during my studies and published in IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft'16) in 2016. ADDICTED re-factors ad-supported apps automatically by adjusting the type and frequency of mobile ad occurrences at runtime based on policies created during the AD-APT development that consider mobile factors – internet consumption and battery life. Furthermore, ADDICTED includes policies that consider the human factor- the user's physical and emotional state that derive from raw data from a wearable device. These policies were developed during the course of my individual thesis.

A mobile and a wearable app were developed and tested on more than 30 users to evaluate the accuracy of PSR and ESR algorithms.

Analysis of the data that was collected during the testing phase showed that ESR algorithm has an accuracy of 75% at recognizing if the user is "stressed" or "relaxed". Also, the analysis showed that PSR algorithm has an accuracy of 85% at recognizing the physical state of the user regarding the four states that we discussed in previous chapters – "running", "walking",

"standing" and "sitting". A third analysis of the PSR algorithm showed that PSR algorithm has an accuracy of 97.5% at recognizing if the user is moving or not.

6.2 Future work

After the testing and evaluation of the algorithms a long-term study will be performed for approx. 1-2 months. During that period users will use a custom Facebook app that we developed but without our policies so they will see the ads without any adaptations. After a period of one week the users will use another custom Facebook app but this time our policies will be included in the app and ads will be dynamically adapted. After the second week of interaction with the modified Facebook app, a questionnaire will be handed to the users, that will evaluate our apps and state if the policies had an impact regarding their interaction with the app and the user satisfaction.

Bibliography

[1] Univ. of Cambridge. What Is The Price of Free? 2012. http://www.cam.ac.uk/research/news/what-is-the-price-of-free.

[2] P. Mohan, S. Nath, and O. Riva. Prefetching Mobile Ads: Can Advertising Systems Afford It? In EuroSys, 2013.

[3] What is HRV. http://www.megaemg.com/knowledge/heart-rate-variability-hrv/

[4] A. Pamboris, G. Antoniou, C. Makris, P. Andreou, and G. Samaras, "AD-APT: Blurring the Boundary Between Mobile Advertising and User Satisfaction," in IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft'16), ser. Proceedings of the 3rd IEEE/ACM International Conference on Mobile Software Engineering and Systems, vol. 1. Austin, Texas, United States: IEEE, May 2016. http://dl.acm.org/citation.cfm?id=2897090

[5] Android Studio IDE. https://developer.android.com/studio/index.html

[6] https://www.kantarworldpanel.com/global/News/Samsung-Focuses-on-New-Goodies-for-Installed-Base

[7] Android API Distribution https://developer.android.com/about/dashboards/index.html#Platform

[8] SAMSUNG website: http://www.samsung.com/us/search/searchMain?N=/_/N-10/Nr-QU5EKFNpdGVUeXBlOmdsb2JhbCxJc0hpZGRlbjpOKQ==&Ntt=s5

[9] http://www.toptenreviews.com/mobile/smartwatches/best-smartwatches/moto-360-review

[10] http://www.wseas.us/e-library/conferences/2011/Florence/AIASABEBI/AIASABEBI-62.pdf

[11] D. Mizell. Using gravity to estimate accelerometer orientation. In ISWC '03: Proceedings of the 7th IEEE International Symposium on Wearable Computers, page 252, Washington, DC, USA, 2003. IEEE Computer Society.

[12] J. Gui, S. Mcilroy, M. Nagappan, and W. G. J. Halfond. Truth in Advertising: The Hidden Cost of Mobile Ads for Software Developers. In ICSE, 2015.

[13] A. J. Khan, V. Subbaraju, A. Misra, and S. Seshan. Mitigating the True Cost of Advertisement-supported "Free" Mobile Applications. In HotMobile, 2012.
[14] P. Mohan, S. Nath, and O. Riva. Prefetching Mobile Ads: Can Advertising Systems Afford It? In EuroSys, 2013.
[15] A. Pathak, Y. C. Hu, and M. Zhang. Where is the Energy Spent Inside My App?: Fine Grained Energy Accounting on Smartphones with Eprof. In EuroSys, 2012.
[16] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, K. Papagiannaki, H. Haddadi, and J. Crowcroft. Breaking for Commercials: Characterizing Mobile Advertising. In IMC, 2012.
[17] R. Mittal, A. Kansal, and R. Chandra. Empowering Developers to Estimate App Energy Consumption. In Mobicom, 2012.
[18] C. Shepard, A. Rahmati, C. Tossell, L. Zhong, and P. Kortum. LiveLab: Measuring Wireless Networks and Smartphone Users in the Field. In SIGMETRICS, 2011.
[19] InMobi. InMobi Ad Specs, 2015. http://japan.inmobi.com/ui/pdfs/ad-specs.pdf.
[20] Bitbucket. Smali/Baksmali, 2015. https://bitbucket.org/JesusFreke/smali/downloads.
[21] GooglePlay. ZipSigner, 2015. https://goo.gl/ymUoY5.
[22] AT&T. Application Resource Optimizer (ARO), 2015. https://developer.att.com/application-resource-optimizer.
[23] https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/