Learning Processes and Consistency in the Quality of Software Development Based on Environmental Influences

by

Michael Christian Frick, B.Sc.

Dissertation

Presented to the

University of Dublin, Trinity College

in fulfillment

of the requirements

for the Degree of

Master of Science in Computer Science

University of Dublin, Trinity College

September 2016

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Michael Christian Frick

July 13, 2016

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

Michael Christian Frick

July 13, 2016

Acknowledgments

Give many thanks and stuff...

Michael Christian Frick

University of Dublin, Trinity College

September 2016

Learning Processes and Consistency in the Quality of Software Development Based on Environmental Influences

Publication No. _____

Michael Christian Frick, M.Sc.
University of Dublin, Trinity College, 2016

Supervisor: Stephen Barrett

Abstract

this is my abstract

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Be a yardstick of quality. Some people aren't used to an environment where excellence is expected."

Steve Jobs

A computer is the powerful hardware that is able to do much more than a human being could be capable to do. However, The enormous capacity is useless without software which is making use of it. In order to make the increasing computations controllable and keep the machines as customizable as possible, different layers of abstractions allow to control and use the computing power. The abstraction reaches from machine readable code over Assembly to high level programming languages which can almost be read as an English sentence and on top of that libraries and frameworks that already provide functionality [18]. A big part of the complexity is already encapsulated and the software engineers can focus on the functionality of the software and their specific problems.

Over the years, the performance of the computers rapidly increased and with it the complexity of the code [24]. Software can be a simple tool that is written in a short time by a single person or it can be a gigantic software project with several hundred

developers [3]. In order to allow to split the work on a software project, an encapsulation of the modules is mandatory. A general structure must be given to ensure that the different parts can integrate hand in hand and to keep the code understandable. The more people work on one project, the more important it is to provide a organized, well planned architecture and keep the code clean.

In today's world, software is everywhere. The traffic is controlled by computers, security systems, nuclear power plants or just a messenger app on a mobile phone. The ubiquity of computers can make life easier, but can also cause unpredictable trouble.

In the early 1890s at the United Kingdom's Royal Air Force an engineer found a bug that could fire a missile without any command. [20] The quality requirements varies for different software products. A crashing weather app on a mobile is not as bad a bug causing a production stop in a plant. However, the quality of the software can make the difference whether a company will be successful or just be one of many abortive start-ups with a good idea but a bad execution.

In the software industry, the most significant factor in the creating process is the human. The quality strongly depends on the performance of the programmers and that can change by various different reasons even a few times a day. Previous smart people already did research and their experiments and I will start the dissertation with their findings of previous studies that investigated different theories about influencing factors on cognitive performance and related work.

In this dissertation, my own approach will be to make use of the sensors and information provided by the user's mobile phone. I will write and Android application that the user can install on his/her device, which is gathering the location of the user, collecting sensor data from the light sensor, accelerometer, the environmental noise from the microphone and the data from the step counter of the device.
The sensors and the software will gather the data while the participant is working an a

provided programming task. Afterwards the user gets asked to answer some questions such as information about disturbance or for example mood and more.
Based on that information I will try to find patterns in behavior and environment that are influencing the quality of a programmer. The gathered information will be clustered into a specific classified behavior or context and compare them to find correlations with the code quality.

For determining the code quality I will use a tool that can analyze the code that is uploaded on GitHub and calculate a level of quality.

I expect that this work will find some influencing factors for the general programmer. Of course, programmers are all different and their optimal environment and way to work is different but I hope that this work is providing the basics and shows the importance for giving feedback to software developers in order to improve.

In the future, the app could become an every-day tool for developers and students. It could be used hand in hand with project management tools, that require the exact times, a programmer worked on a project and it could simultaneously provide real time feedback about the code quality itself or destructive aspects in the working environment. Rather than comparing the information with other app users, the app could make use of systematic learning of and optimal working environment and behavior of the specific programmer. Having this information, the app could inform the user when the optimal environment is not given and provide the information how it could be reached (e.g. "Find a place with better lighting conditions and less environmental noise to increase your productivity").

1.1 Motivation

Learning how to program is getting more and more important and still not a requiered subject in school. On the other hand, it is the computer with it's software which is

controlling almost everything in our everyday life such as traffic, gates, calendars etc. I failure could have dramatic impacts in peoples life.

Therefore it is very important that programmers produce high quality code. The problem is that it is not always obvious what high quality means. It can vary from good structured code to resource-aware, reliability and much more. A lot of programmers didn't learn coding in school or university. They taught it themself and they might just used in for fun-project which were not created for public usage. However, what I want to say is that a programmer doesn't really know how good or bad his/her code really is.

I experienced this problem myself. I started to work in an agency that specialized on iPad apps and design. I was hired because the previous mobile developers left the company and they urgently needed replacement. When I started I had no practical experience in writing mobile applications. I had to maintenance the current code and add new features. As is had no mentor or anyone who could give me feedback I just did it as good as I could. I still don't know whether I created good or bad code. With feedback of the code quality I could have learned a lot lot about the coding itself and by today I would probably write much better code.

The feedback of the code quality can then be used to find and improve influencing factors.

How important is a good working environment?

A new trend, especially in the tech industry is going from common clean looking office spaces to colorful creative environments closer to living rooms. Companies like Google or Facebook seem to get rid of the strict separation of work and personal life. Companies introduce unlimited holiday policies, provide free food and even have a laundry service for their employees. They try to remove all the obstacles from their employees life to allow them focus on their work. Also the social aspect at work changes a lot. Some years ago, things like having a beer with the co-workers after office hours or meeting for a ping-pong match during the day was unthinkable. Google tries to make developers communicate more with the team by placing the whole team in relatively small spaces and provide silent areas for tasks that require more silence. All these efforts to make the employees

more productive are very interesting approaches but hard to measure.

In my dissertation I am trying to find patterns between working environment, behaviors and the resulting code quality in learning and professional environments. I also think that the creation of awareness for code quality and performance is an essential factor in the evolution of a programmer and is important at any stage of the experience and my work could be a base for tools that are providing these information to the software developers and engineers.

1.2 Aims

I hope to find patterns in the working environment and behavior for in general that significantly influence the quality in software development. This knowledge might inspire and help future researchers and the industry to do more work in this area and create tools for bringing the code quality of future software to a higher standard. Also for academia, a tool that provides feedback on code quality for the students can help to bring them on a higher level when they leave college.

1.3 Road-map

In the next chapter I am summarizing the state of the art in the area of defining, measuring and improving code quality. I'm am also writing about previous research in cognitive performance and influencing working environments.
Afterwards I will describe my design concepts of my approach followed by description of the implementation. The following chapters will inform about a study which is done using the developed Android application and the results. At the end I will evaluate the results and conclude the outcomes and findings.

Chapter 2 - State of the Art Chapter 3 - Design of Experiments Chapter 4 - Implementation Chapter 5 - Experiment Chapter 6 - Evaluation Chapter 7 - Conclusion

Chapter 2

The State of the Art

Many researchers are concerned about finding metrics of software quality with the human factor as the most significant part in the development process. Research has also been done in the process of finding and testing factors which are influencing the cognitive performance of software developers.
In my dissertation I aim to identify environmental influences that can improve, worsen or influence the code quality of Computer Science students and professional software developers. With the students as a key factor in future software engineering, I will analyze their behavior and performance influences in a environmental and psychological sense but also compare it with the numbers from the professionals. [4]

Measuring quality of software projects and gaining information about the progress are valuable information for the software engineers and developers to reflect their performance. The provided feedback helps them to identify their weaknesses and improve their skills or optimize their work patterns [12] [18].
Also project managers have a great interest in details about the progress and the products quality in order to coordinate the schedules, resources and have an overview about the possible bottlenecks in the project. Early knowledge about potential problems can help them to target it and make a difference between the success or failure of a software project.

A good programmer nowadays is described as a person who can solve complicated problems by breaking them down in smaller targeted problems that are easy to understand and to solve. Good code is supposed to be clean, easy to read and as simplyfied as possible [12].
This new approach differs from the early days when programmers tried to find the shortest and most performant solutions. As long as code was using minimal resources it was fine. Less people were working on projects and the open source community was not as important and big as today and withe a lower computing power, the computers where unable to handle the complexity that software has today. More people are working together on different parts of complex systems. At the end every part must go hand in hand with all other parts and the code should have a similar structure so that people from one team could possibly also work or help out in another team.

From a research perspective it is very interesting to get an overview approaches from different years to gain a broader understanding. Thus the following paragraphs will summarize information about code metrics and code quality from several decades.

2.1 Software Metrics

Since the late 1960s, when the software engineering was in it's beginning, people wanted to measure and produce numbers to characterize code properties. The first metrics where used and developed to measure and evaluate the performance of a programmer. Lines of Code (LOC) per month and bugs per thousand lines of code (KLOC) are a very simple but efficient ways for examining the productivity, which can be used to for comparison with other programmers or general standards.

Software metrics is the term that is been used more then 30 years ago up to today. Today, some of the most metrics are still being used to investigate the productivity of the software developers. The amount of bugs in relation to the amount of code, the ini-

tiall number of requierements compared to the requierememts at the current point in the project and the effort it takes to fix faults versus the total time the project reqieres. [15] The metrics have been a great success in the industry. Most of the big software companies and even smaller ones use metrics, though they are barely used in academia. The metrics are created for larger software and scopes. Also maintenance and re-factoring is not as important in academia as it is in the industry with commercial software. Also, Software metrics in the industry are primarily important for the management rather than for the development process.

Industrial software metrics can be used to ensure quality, productivity and even for predictions of the software quality and it's reliability [5]. Several researchers investigated and developed approaches to improve the metrics and the results which they are generating. Yue Jiang et. al. from West Virginia University researched methods for improving software quality predictions. They used supervised machine learning algorithms with datasets and focused in improving of the information content of the training data in their research. The results at the end showed that the biggest differences in the quality of the predictions are generated by the choice of the right software metrics rather than applying different machine learning algorithm. [11]

2.2 Metrics Measurement and Analysis

PSP - Personal Software Process is a way to gather data about the Software engineering process and analysis of the information. Over the last decades, the University of Hawaii did a lot of research about PSP and they developed different approaches to make students to adopt and use it in their project and even later at work. Their first approach was originally described as âĂIJA Discipline for Software EngineeringâĂİ. It required the users to keep records about all the metrics by hand. The massive overhead was a high barrier for the students to adopt and keep on working with the PSP. For the best results they needed to write down every compiler error and they had to track the time they were

working on their projects and had to stop it for interruptions.

In 1998 the University of Hawaii started the Leap research project to provide a PSP with low overhead for the collection and analysis of the data. This generation of PSP was using automated tools which were asking the user for inputting the data. These tools were also able to display information and analyses to the user. Just a few students adopted the system. The researchers found out that another reason for the reluctant adoption was the constant context switches for the users. Inputting the data during the programming task interrupted and disturbed the ability to focus on the programming tasks. [14] In order to eliminate the adopting barriers, they started the Hackystat project in 2001. Hackystat is an open source framework for automatically gathering all the required metrics by data collection plugins in the development environments of the users.

Plugins, that are installed by the in their programming environments automatically collect the data and forward it to a centralized web service. The web service orders and analyzes the data. If interesting results occur, the webservice sends emails to the developers to inform them about it. The web service also provides a rich visual representation of the data. All the different approaches to provide feedback about the code lead to improvements in the quality and the ability to estimate software projects. [13] The Appalachian State University in North Carolina described a different approach. Their goal was to decrease the high attrition rate of computer science students and increase the attraction to get a computer science degree in general.

The researchers were monitoring the students software development behaviour in order to find good practises for successfully learning programming. For gathering the data of the individual students, they developed a tool called ClockIt. ClockIt allows to, fully automatically, collect the data, analyze it and compare the results with the results from better or more experienced students visually. A web interface provides access to measurements for the student, the course instructor or an administrator.

In their results they compared the data of three students out of 75 participants. The students with the best results, an average scoring student and the one with the lowest

grade. The comparison showed that the best student also spent the most time on the project, but wrote less code than the average scoring student who spend almost as much time. The worst student spend the least time and submitted the smallest amount of code. There was an interesting correlation between the grade and the compilation errors and the amount of compilations that were made. The best student compiled the code more than double as much as the average student and almost 6 times more than the worst student did. [19]

2.3 Mobile Data Gathering

Ferreira, D., et al. [6] from the University of Oulu, Finland and the Carnegie Mellon University were working on a toolkit for gathering the sensor data from mobile Android devices. They created an extensible framework that could have been used in any Android application at the time when the paper was released. They also released an application for research purposes (the Aware client). The Aware client is extendable with plugins to support more than the pre-installed sensors. By default, the application stores the gathered data on the local hard disk but can also be uploaded to a database.
The sensoring is optimized to keep the energy impact as little as possible and not to use more device resources than necessary.

Another approach in the area of mobile data gathering have been made by University of Science and Technology of China HUI XIONG, Rutgers University in cooperation with Nokia. In order to detect the context of the mobile device, Zhu, Hengshu, et al. [25] were reading the log files of the device. The device logs provide information about location, accelerometers and optical sensor as well as browser history or which apps were used and are automatically recorded by the Android operating system. These information can be used to provide context aware suggestions e.g. for other games or based on the physical location. To read the logged information, the needs to be physically connected computer. The information from the device logs are much richer than the information which can be

gathered in an application with the downside, that the device needs to be connected to a computer in order to access the information. A installed application can compute, store and transfer the data to a remote server from everywhere. The only requirement is access to the internet.

2.4 Cognitive Performance

Improving the perfomance in Software Development can be done in a several different ways. One approach to improve the performance is the optimization of the working environment. Amabile, Teresa M., et al. [2] wrote about a conceptual model for increasing creativity in the work environment. Five key factors were described. The first two factors were, the encouragement for innovation and creativity as part of the company culture as well as according autonomy or freedom for the employees. Another key described the adequate availability of resources for a project which might affect people psychologically by the feeling to work on a valuable project. Also pressure at work was identified to increase creativity on a balanced level between excessive demands and boring routine. The last key factor in their model described the organizational impediments to creativity which could be caused by internal competitions. A study was designed to investigate two hypotheses: The influence of the model in high-creative projects vs low-creative projects is expected to be much bigger. As well as obstacles scales are lower in high-creative projects compared to low-creative projects for workload with pressure and organizational impediments. Both hypotheses had clear result outcomes, which showed that beside the employee's itself, the management can significantly influence the level of creativity and innovation by forming the organization culture. The construction of the teams and definition of the individual roles can have a great impact on the creativity.

The cognitive performance can very based on the context and the environment. When the body is in a relaxed state, the mind also slows down to save resources. Back in the stone age that made sense because thinking was not as important as today. Cognitive

performance was mainly needed in dangerous or unusual situations where the heart beats faster and provides the brain with more oxigen and the arousal is higher than normally. Researchers from the Brunel University in the UK showed movie clips to invoke different defined moods of the participants before they solved given debugging/coding tests. The results showed improvements in the results after the participants were seeing high arousal video clips. Low arousal clips affected the performance negatively compared to neutral clips. [16] It is called the Yerkes-Dodson Law, that proclaims that a higher level of arousal leads to better cognitive performance. As caffeine also influences the arousal, it also can boost the cognitive performance and is not just helping to wake up in the morning. Watters, Paul Andrew et. al. [23] found out that the mean for the best cognitive results is an amount of 400 mg caffeine (the amount contained in ca. 5 espresso shots).

When an arousal is influencing the performance of a programmer other factors that are effecting the mood could also have an impact in the quality of the written software. Many people believe that, for example the weather has a strong influence in the daily mood of a person. Certainly, Denissen, Jaap JA, et al. [4] found out that the sunshine alone actually has no noteable effect in the mood of the most of the people. However, they found significant correlations between sunlight, air pressure and precipitation on the tiredness of the participants. A reason for the influence of sunlight could be vitamin D3. The most of it is obtained through exposure to sunlight and it changes the level of serotonin which was found to be partly responsible for the mood of a human.

2.5 Long Term improvements

A very different, but probably the most important factor in the long term cognitive performance is the consumed food. The human brain needs good fuel to run properly. A wrong diet can strongly influence the incidences of cognitive problems as well as healthy food can positively influence for healthy ageing [21].
Some foods demonstrated positive effects on the mental performance containing flavonoids

like for example grapes, tea, cocoa and blueberries. Different to the previous influences, the diet and the lifestyle are less obvious. Their impact is slowly showing over years and it's hard to prove their effects and that they are the influencing factors.

Studies on several mammalian species have shown that food which is rich of flavonoids have beneficial effects on memory and learning with the ability to support neurons and protecting them again stress-induced injury. It also decreases the chances of alzheimer and dementias. Other studies have shown that flavonoid-rich food improves the blood circulation and correlates with the growing of new hippocampal cells. These cells are located in the brain region that is identified to be responsible for the memory.

Chapter 3

Background

3.1 Mobile Device Sensors

Over the last decades the evolution mobile devices began with a wireless telephone far away from pocket size. Over the years the mobile devices got displays, SMS, telephone books, games and a lot more. In 2007, Steve Jobs introduced the first iPhone and with it the age of the smartphone [17]. Over the years, smart-phones became pocket size computers with a better display resolution than the most televisions and the computing power of what desktop pc users could just dream about a few years ago. More and more sensors were packet into the small handy devices were made easy accessible by developers. The sensors range from proximity detection over accelerometer to humidity sensors etc. Google even engineered a system for 3D objects and indoor environments with just a single device in real time.

So, a lot of people own the hardware with the capability to collect rich context information and they even carry it with them all the time and could be used for support and improve the peoples work and environment.

Chapter 4

Design

In this chapter, I describe the design of the Android application for gathering the data as well as the sever side implementation to store and compute the information. First, I will start with a brief description of the two components and will follow with my design decisions and my reasons for the choices.

4.1 Functionality Overview

The purpose of the software is to gather information from a mobile device of a participant while he/she is working on a programming task. Afterwards the application sends the collected data to a server for further processing and analysis. The participant also simultaneously submits the written code which code quality will be detected and then correlated with the processed mobile device information.

4.2 Mobile Application Design

In quarter 4 of 2015 Android had a market share of 80.7% in smart-phone sales by operating system (see Figure 4.1). The trend also shows that the number increased from the last year [7]. Therefore I decided to realize the mobile application implementation

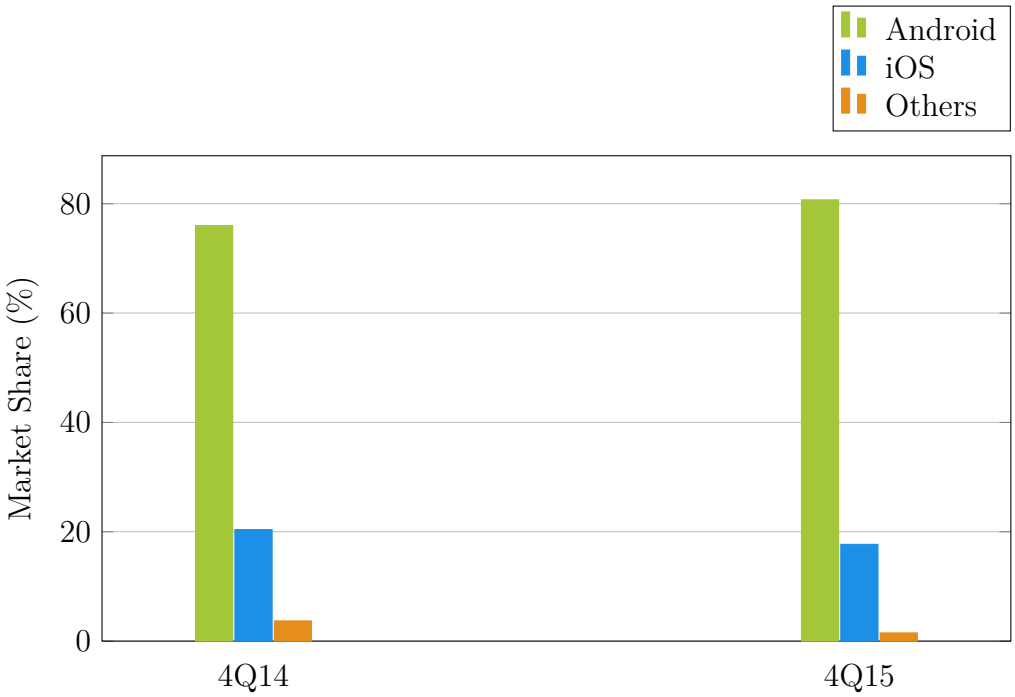


Figure 4.1: Smartphone OS Marketshare

for Android in order to be able to work with more users who have access to that application.

An alternative to the native implementation (e.g. iOS or Android) could have been a hybrid application. A hybrid apps is based on web-technology and using the addvantage of resposive web design to be able to work with every aspect ratio and resolution on an mobile device. One way doing that would be by using a framework such as PhoneGap, wich internally creates a native webview applicationand just loads the hybrid JavaScript, HTML, CSS in it. Another software for cerating a hybrid solution is Titanum accelerator which itself is using native UI components. Both frameworks have the advantage is the simple development and the OS independence. The problem with hybrid apps are the performance and limited accessiblity to hardware components including some sensors [10].

The Android application make use of its build in sensors and information provided by

the Android operating system. Different than iOS, Android is an OS that can be installed of different devices from different vendors and with different hardware components [8]. Thus the buit in sensors which are clustered in motion sensors, environmental sensors and position sensors [9] can differ between the different devices. Components which are required for standard functionality such as making phone calls are more common than other sensors. For example the the microphone for recording the users voice or the light sensor, which is used to detect whether the user has the phone at his ear can be found in almost every mobile android device.

4.2.1 App Architecture

The Android app is built based on the Model View Controller design principles. This design principle defines the interfaces between the three different parts, the Model, the View and the Controller and states the tasks and responsibilities of each part.
The Model is the data source and in this app represented by the SQLite Database and can only communication happens to the Controller. The Controllers are called Activities in the Android Framework and is responsible for managing the Views, which are defined in XML files and then modified by the responsible Controller. To keep the code base clean and to avoid bugs, the communication is separated by the controller. The Model doesn't directly communicate with the View and can't update it. It the case of changes, the Model informs the Controller, which decides whether or not to update the view etc.

4.2.2 User Interface

The Application contains of two main user Interfaces, the gathering view and the question view.
The gathering view is the control interface for starting and stopping the data gathering. Its interface changes depending the current state and the logical functionality and also provides an input field for the users email address.

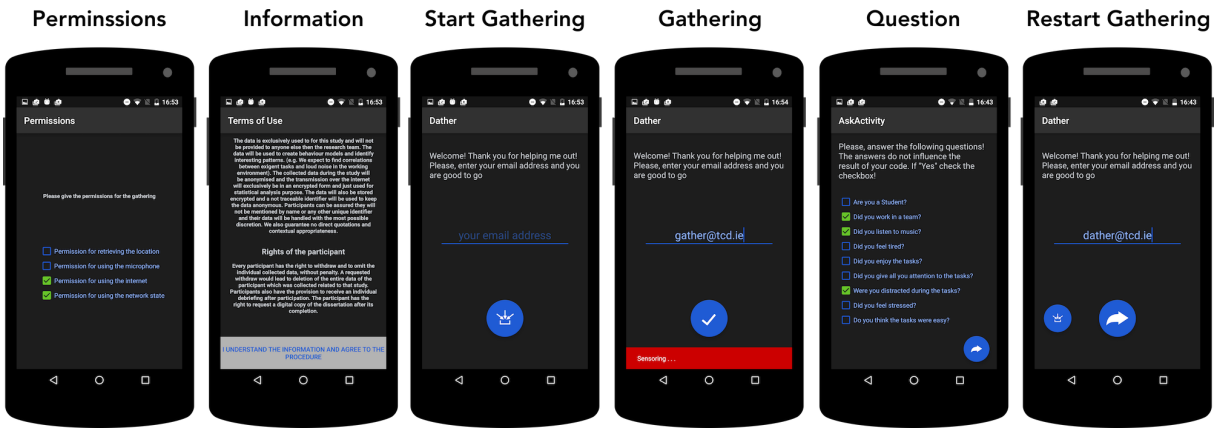


Figure 4.2: Android Views

After sending the gathered data to the server the application displays the question view. Here the user sees a number of questions and check-boxes to answer the binary yes and no questions. This interface also provides a send button to submit the answered questions to the Server.

In Figure 4.2 you can see the flow of the views in the applications.

4.2.3 Data Storage

For storing the gathered data entries, the app uses a SQLite datebase. SQLite uses SQL syntax which and is embedded in Android and well documented by Google. [22]. It is a very light weight database and provides an abstracted and easy way to store the values in an object oriented environment. SQLite is storing the data unencrypted by default. In order to make sure the stored data is save and can't be read, the data is been encrypted as soon it's been gathered.
Additional to the SQLite database, Android provides a way to store single entries such as the field with the email address of the user. The so called "shared preferences" are storing key-value-pairs persistently on the phone and can be accessed from everywhere in the application.

4.2.4 API

The gathered data and afterwards the answered questions will asynchronously send to the Server and written into the database. When the response will be received, the view shows a visual confirmation and moves on to the next step or in case of the questions back to the gathering interface.

The communication to the Server uses HTTP connection over a stateless REST-full service. The REST-Service is a centralized way to allow making entries to the database and use the computing power of the server. PHP is the programming language used server-side and is establishing the connection to the MYSQL database.
As this application just requires to send information to the server but not receive information, the data gets converted to JSON format and send to the server via a POST request. The server response with a success or fail and provides some additional information in case entries were added to the database.

4.2.5 User Information

For accessing the information from the mobile device, Android requires the user to granting the permissions for specific functionality such as using using the internet connection of the phone or access information like the telephone book entries.
The permissions were given when the the user by simply downloading the app from the PlayStore all in one place. However, since Android 6, the developer now is forced to ask for the permissions within the application itself [1].
Thus, the users with Android 6 or higher are provided by a additional interface which is specifically asking for permissions for Internet Access, access internet state information, using the device microphone and getting the location of the device.

Independent from the Android version the user needs to read and accept the terms of use at the first start of the application. The displayed text informs the user about the

rights and what is happening with the data and information the user provides through this app usage.

Within the app the user is asked to enter his/her email address. The email address then is being hashed with the SHA256 algorithm to ensure the data will be anonymous and can't connected to the user.

Also the gathered data are stored encrypted in the SQLite database on the mobile device and just decrypted on a local computer after the researchers downloaded the encrypted entries from the SQL database on the server. That ensure that the files are not accessible in readable format at any time.

4.3 Server Design

The server is hosted by 1&1 Internet SE as a completely pre-configured server backend PHP version 5.6 and the MYSQL-server phpMyAdmin in version 4.1.14.8. The also pre-configured server file system can be accessed using SSH or via FTP.

4.3.1 Server Frontend Design

The information for the participants of the study can access information web pages and access the downloadable Android application.
The frontend is implemented in HTML5 and CSS3 and simply uploaded to the server. As everything is public available and accessible there was no need or using a framework or any further security implementation.

4.3.2 Server Backend Design

The backend is implemented in PHP with the MYSQL-server phpMyAdmin database. No external frameworks have been used to implement the basic REST-full service and the establishing of the database connection and the SQL queries.

4.4 Encrpting Tool

The encryption tool an executable program written in Java for locally encrypting the downloaded gathered data.

Its interface contains of an input field for the path of the downloaded data in JSON-format, a button for first reading and afterwards decrypting and a output text that shows the current state of the application. The tool writes the decrypted input values in a separate text file in the same directory of the input file with the same name but with the file extension .txt instead of .json.

Chapter 5

Implementation

In this chapter I will describe the implementation of the main application for Android, the server back-end and webpages as well as the encrypting tool java application.

5.1 Android

As mentioned before, the Android Application is for sensoring data of the user and get environmental information. For This purpose I wrote an Android application which can gather these information.

Beside gathering the data using an App it is also possible to read the sensoring information which are being recorded continuously as described in the approach of Zhu, Hengshu, et al. [25]. They are reading the device logs and get all the logged device more information about the apps being used etc. . Sandboxing is an Android security concept that only allows an app to access the data of the app itself and isolates the content for other applications. Thus it is impossible to access the device logs via an app without having physical access to the device.

In terms of the ideas for future usage of the app it doesn't make sense to require physical access to the device itself. Thus, the decision to use an App, installed on the users device, is the best way to go for this purpose.

The implementation of the Android application has been done using the Android Studio IDE, which is provided free usage by Google, Inc. The code was written in Java, which is the official programming language for Android applications. Google also provides a variety of libraries and frameworks for user interface-Elements and basic functionality. For the user interface Android Studio has build in Solutions to either design the graphical user interface (GUI) using Java code or defining the elements in XML files.

5.1.1 User Interface

The user interface contains of two main views, the gathering view and the question view. There are two more views, one for asking the user for permissions and a second one for informing about the experiment and the terms of the usage for the application.
All view have a controller/activity java class which acts as the controller. The actual views contain of an activity XML and, depending on the complexity of the view, an additional content XML. Both are defining the UI-elements in XML tags and as well as their positioning within the view.
The colorscheme of the app is mainly a dark grey background with a combination of bright UI-Elements and simple lightgrey fonts for information texts.

Gather View

The gather view contains of an input filed for the users email address and a dynamic changing interface to for controlling the gathering and uploading process. The buttons are a blue circle shape with an icon for showing the functionality of the button itself. The Icons are a white shape without borders and designed to give a clear idea about the representing purpose of the button. Depending on the different states of the gathering process, the buttons change in functionality and look. In the first state, it only makes sense to display the button that starts the gathering of the data. Once pressed a red bar with an information text on the bottom of the view indicates the running gathering

process and the button that was starting the gathering changed to a new button for stopping the process.
A tap on the stop-button removes the red information bar disappears and the button changes its appearance and functionality to share/upload. At the same time, a smaller button appears on left hand side in the view which can restart the gathering process. After tapping on the share button, a green bar appears on the bottom and the question view opens.

Question View

The question view contains of a short information text that introduces the user to the new interface and a bunch of checkboxes for questions on it's left side. The questions can be either checked, to indicate a "yes" for the answer or can remain unchecked for "no".
On the bottom of the view is another share button which sends the answered questions to the server once tapped.
The successful send is also being indicated by a green bar at the bottom and the question view is being replaced by the gather view.

Information View

This view contains a scrollview with a long formatted text. At the bottom of the scrollview is a button. With a tap on the button the user confirms the he/she read and understood the previous text and the user can go on to the next step which is either the permissions view or the gather view.

Permissions View

The Permissions view just contains of four checkboxes with it's descriptions, each for one permission. This view is just shown on devices with an Android version of at least 6.0. Once all the permissions are checked, a button appears which allows to go on. A tap brings the user to the gather view.

5.1.2 Data Gathering

The data gathering is managed by the gather class while the functionality is been managed by the sensor class. The most sensors can just be accessed by creating an instance of the single sensors. However, some, such as the environment volume have been customized individually in separate classes. The volume is no predefined sensor and needed to be created from the recording framework but without actually recording the sound. It is calculating the decibel from the current recording and just saved the gathered volume value. That ensures the privacy of the user and also doesn't need so much memory of the mobile device capacity.

As well as the volume measuring, the location has a custom implementation that uses the GPS or Wifi signal to calculate the current latitude and longitude of the device.

The app is gathering the data of each sensor every few seconds, between every 2 and 10 seconds, depending on the device speed. After receiving all the values from the sensors, microphone and Android OS, the app is generating a timestamp, adds the user ID to the entry.

This way to handle the gathered data make each singly entry independent from each other and can still be used in case of damaged data in some other entries.

5.1.3 Data Storage

Variables and temporary available resources are stored in memory during the runtime of the app. Anyhow, the memory can just store information as long as it's powered. The memory is also managed by the Android operating system and can be overwritten by other applications, once they are higher prioritized.

To store the entries and the user information permanently on the device on the hard disc its been stored in an SQLite database. The SQLite database handles the organization and keeps everything in a ordered form. It is also is resource optimized and allows easy access to the database from the applications.

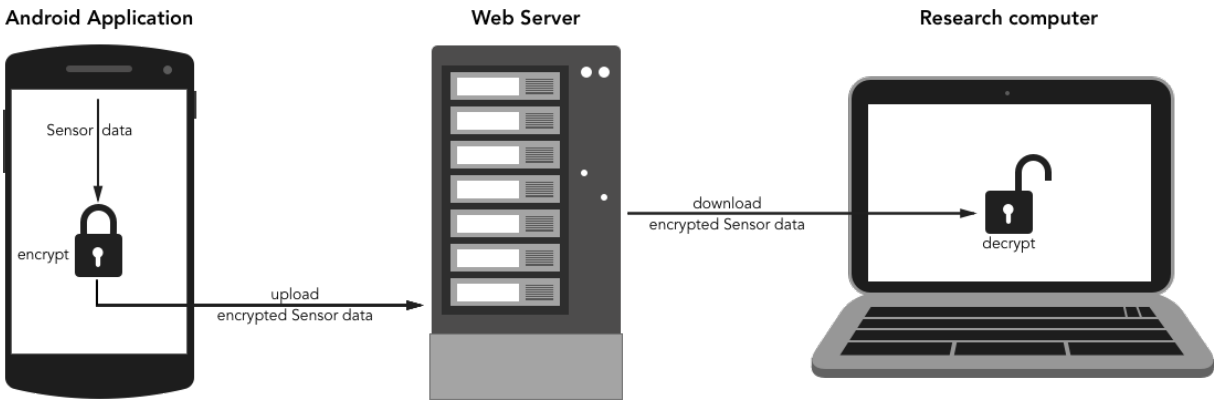


Figure 5.1: Security Dataflow

The only data that is being stored permanently is the encrypted gathered sensor data. The permanent storage make sure that the data is not lost in the unlikely case of a crash of the application or a failing in sending the data to the server.
In order to save states such as the information weather the user already confirmed the he/she read and understood the terms of use, Android provides a method called Shared-Preferences. They can store single key-value pairs and are additional to the app-states used to store the users email address to avoid that he/she has to type it in every time the app restarts.

5.1.4 Security

In order to prevent that the participants can be identified by the user id because it is been generated by a SHA256 hash function that is infeasible to invert. In other words, the SHA256 algorithm generates a base16-String from the email-address of the user and there is no mathematical known way to recover the original email address in feasible time from the base16-String.

For encrypting the gathered data, the entries are independently getting encrypted before written to the database using a hybrid cryptographic procedure. Hybrid cryptography means the combination of using a the faster and performance friendlier symmetric

cryptography (using the same key for encrypting and decrypting) and the slower but more secure asymmetric cryptography. In the asymmetric procedure also known as public-key cryptography, uses two different keys for encrypting and decrypting. A public key is used for the encryption of the data and the private counterpart is used to decrypt the data.

The encryption is the Android app works as follows:
A symmetric key will be generated every time the app starts using the AES CBC algorithm with an PKCS5Padding and a random SHA1 seed. This symmetric key will be used to encrypt the gathered data, while the symmetric key will added to each entry encrypted with the public key of a pre-generated RSA 1024 bit key-pair.
The private counterpart of the public key will later be used to decrypt the symmetric key. That symmetric key is then used to decrypt the entries. The decryption will happen with a separate written Java application locally on a computer.
Thus, a decryption within the applications is not possible because the functionality and keys are not even included.

5.2 Server

The server contains of three different parts, the back-end that handles the REST-full API calls, the MySQL Database and the web-pages for providing information to the participants of the study. In this chapter I will just write about the back-end and the web-pages because the Database design is already described in the previous design chapter.

5.2.1 Back-End

In the PHP script, the data from the POST gets extracted and decoded from JSON to an PHP-Array.
If the format of the data is correct, the script connects to the MySQL database and inserts the values into the corresponding table using SQL-Syntax. For each entry a counter is

increasing it's value and after completing the insertion, the counter-value gets returned as a response argument. When something wrong happens, the script is responds with an error-code.

5.2.2 Webpages

The websites are implemented as simple as possible. They are completely static and only for displaying styled text and images. Therefore the implementation is only been done using HTML5 for the structure and CSS3 for styling the fonts, images and visual structuring. Different fonts were embedded using Google-Webfonts from https://fonts.google.com which are dynamically being loaded at the page load or from the browser cache.

5.3 Encryption Tool

The encryption tool is a Java tool that is written to decrypt the downloaded encrypted JSON-File of the gathered data. The simple tool is implemented in Java and is using the IntelliJ interface builder which is based on XML. First the tool read the input JSON-File and writes the beginning of the file into the output textfield.
Afterwards it is decrypting the symmetric AES-key using the asymmetric RSA-algorithm with the counterpart private-key to the public-key which was used for encrypting. Having the symmetric key allows to decrypt the whole input line by line using the AES decrypting algorithm. At the end the decrypted entries are written to a new created file and the filepath is been displayed in the text label.

Chapter 6

Experiment

In the experiment, participants are solving a programming task while the Dather Android App is running to record behavior and environmental factors. After the submitted code is been analyzed, its been compared with the gathered data in order to find correlations between the participants performance and the information from the gathered data. You can see the flow of the experiment in Figure 4.1.

6.1 Setup and Execution

Every participants need to have access to a mobile phone with Android version 4.4 or later. In order to participate at the experiment the participants need to install the Dather Application of the device. The Application can be downloaded from the website `http://frickm.de` when it's been accessed from the Android device. After installing the downloaded apk-file, the participants permits the application to gather the data and allows the data to be used for research. The last step before being able to start the experiment, the user needs to enter his/her email address.

After setting up the application and accessing the website with a programming task, the experiment is ready to start. The participant starts the gathering process while working on the programming task.

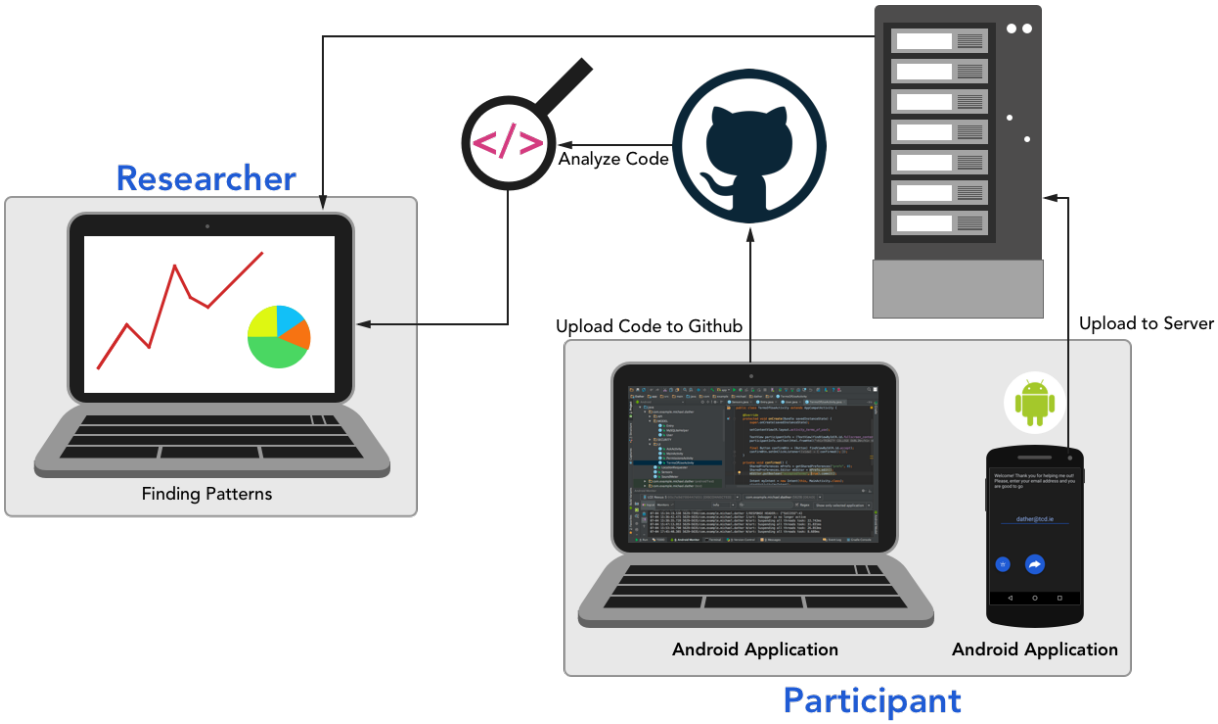


Figure 6.1: Experiment Execution

After completing the task, the participant uploads the solution code to Github and sends a link of the Github repository from the Email address which was entered in the Android application.

6.2 Expected Results

Based on the previous work of other researchers in this area and based on my own experience, I expect to find correlations between the code quality and the environmental noise. The high amount of distraction noises will probably negatively influence the concentration of the participants and decrease the code quality. My own experience showed me that background music can improve the cognitive functionality but depending on the concentration which is needed for the task. Before doing research I was sure to find correlations between sunshine and a better performance compared to cloudy or even rainy days for a big amount of the participants. Anyways, I think that the result might show different

groups of people that perform different in the same environment compared to each other. It would be possible to find out that some group of people work better at night listening to music and another group performs best during the morning on a bench in a quite park.

Chapter 7

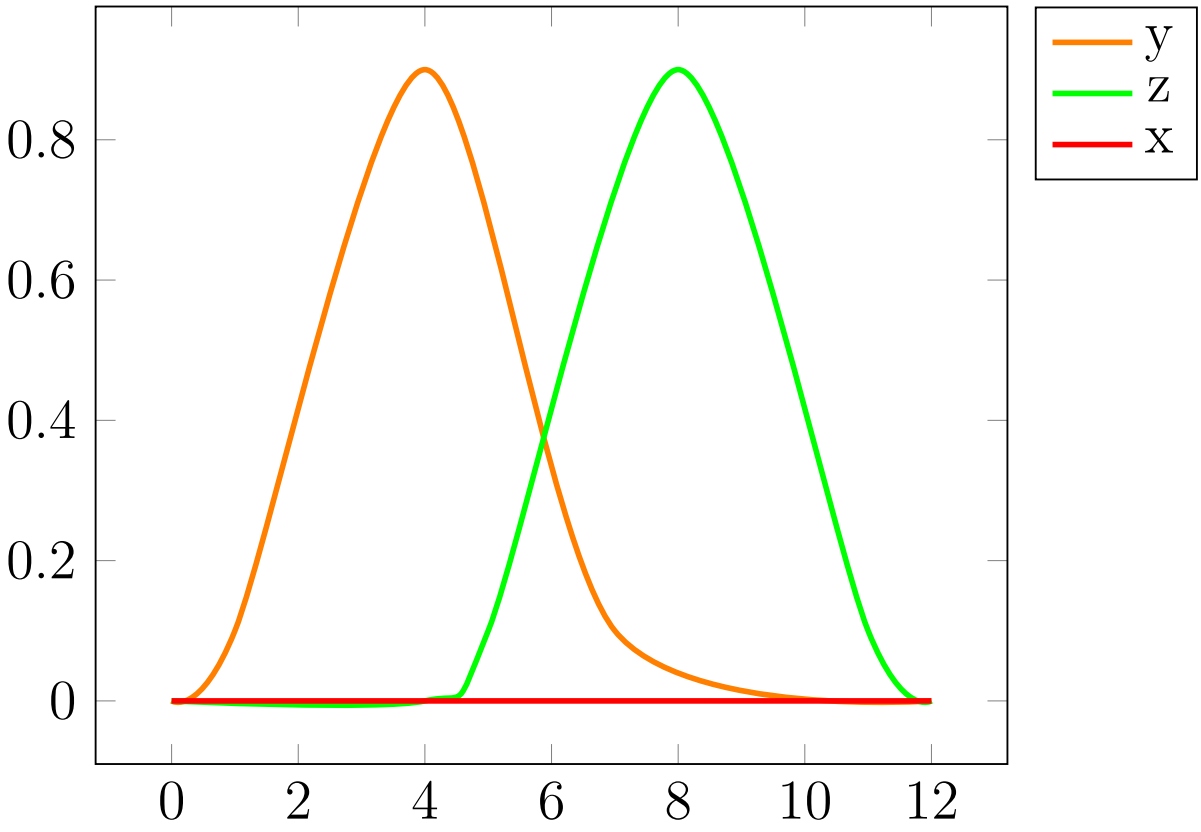
Results

The results.....

The outdoor light as seen in table 7.1!

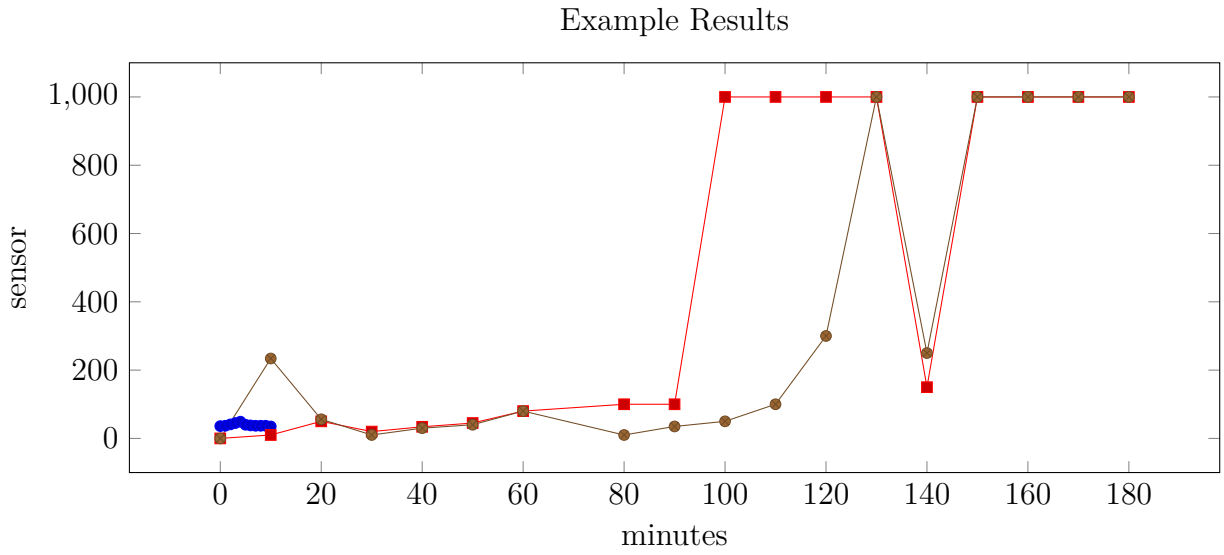
Common Light Levels Outdoor - Daytime	
Condition	**Illumination in lux**
Sunlight	107,527
Full Daylight	10,752.7
Overcast Day	1,075.3
Very Dark Day	107,527

Table 7.1: Common Outdoor Light Levels



Common and Recommended Light Levels Indoor	
Activity/Location	**Illumination in lux**
Warehouses, Homes, Theaters, Archives	150
Easy Office Work, Classes	250
Normal Office Work, PC Work, Study Library, Groceries, Show Rooms, Laboratories	500
Supermarkets, Mechanical Workshops, Office Landscapes	750
Normal Drawing Work, Detailed Mechanical Workshops, Operation Theatres	1,000
Detailed Drawing Work, Very Detailed Mechanical Works	1,500 - 2,000

Table 7.2: Common & Recommended Indoor Light Levels


Example Results
1,000
800
600
400
200
0
sensor
0
20
40
60
80
100
120
140
160
180
minutes

Chapter 8

Conclusions

And a fancy conclusion...

8.1 Future Work

Appendix A

Abbreviations

Short Term	Expanded Term
LOC	Lines of Code
KLOC	Thousand lines of code
PSP	Personal Software Process
CSS	Cascading Style Sheets
HTML	HyperText Markup Language
GUI	Graphical User Interface
UI	User Interface
UX	User Experience
SHA	Secure Hash Algorithm
AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman
CBC	Cipher Block Chaining
PKCS	Public Key Cryptography Standards
SMS	Short Message Service
API	Application programming interface
APK	Android application package

Bibliography

[1] System permissions. `https://developer.android.com/guide/topics/security/permissions.html`. accessed on 02.07.2016.

[2] Teresa M Amabile, Regina Conti, Heather Coon, Jeffrey Lazenby, and Michael Herron. Assessing the work environment for creativity. *Academy of management journal, publisher: Academy of Management*, 39(5):1154–1184, 1996.

[3] Michael A Cusumano. How microsoft makes large teams work like small teams. *MIT Sloan Management Review*, 39(1):9, 1997.

[4] Jaap JA Denissen, Ligaya Butalid, Lars Penke, and Marcel AG Van Aken. The effects of weather on daily mood: A multilevel approach. *Emotion, publisher: American Psychological Association*, 8(5):662–667, 2008.

[5] Norman E Fenton and Martin Neil. Software metrics: successes, failures and new directions. *Journal of Systems and Software, publisher: Elsevier*, 47(2):149–157, 1999.

[6] Denzil Ferreira, Vassilis Kostakos, and Anind K Dey. Aware: mobile context instrumentation framework. frontiers in ict 2, 6: 1–9, 2015.

[7] Inc. Gartner. Gartner says worldwide smartphone sales grew 9.7 percent in fourth quarter of 2015. `http://www.gartner.com/newsroom/id/3215217`, February 2016. accessed on 08.06.2016.

[8] Mark H Goadrich and Michael P Rogers. Smart smartphone development: ios versus android. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 607–612. ACM, 2011.

[9] Inc. Google. Sensors overview. `https://developer.android.com/guide/topics/sensors/sensors_overview.html`. accessed on 08.06.2016.

[10] Andreas Holzinger, Peter Treitler, and Wolfgang Slany. Making apps useable on multiple different mobile platforms: On interoperability for business application development on smartphones. In *Multidisciplinary research and practice for information systems*, pages 176–189. Springer, 2012.

[11] Yue Jiang, Bojan Cuki, Tim Menzies, and Nick Bartlow. Comparing design and code metrics for software quality prediction. In *Proceedings of the 4th international workshop on Predictor models in software engineering*, pages 11–18, Leipzig, Germany, 2008. ACM.

[12] Philip M Johnson. Leap: a ´personal information environment´ for software engineers. In *Proceedings of the 1999 International Conference on Software Engineering*, pages 654–657, Los Angeles, California, USA, Mai 1999. IEEE.

[13] Philip M Johnson. Project hackystat: Accelerating adoption of empirically guided software development through non-disruptive, developer-centric, in-process data collection and analysis. *Department of Information and Computer Sciences, University of Hawaii, publisher: University of Hawaii*, 22, 2001.

[14] Philip M Johnson, Hongbing Kou, Joy Agustin, Christopher Chan, Carleton Moore, Jitender Miglani, Shenyan Zhen, and William EJ Doane. Beyond the personal software process: Metrics collection and analysis for the differently disciplined. In *Proceedings of the 25th international Conference on Software Engineering*, pages 641–646, Honolulu, Hawaii, USA, 2003. IEEE Computer Society.

[15] Cem Kaner et al. Software engineering metrics: What do they measure and how do we know? In *In METRICS 2004. IEEE CS*, Florida Institute of Technology, Melbourne, Florida, USA, 2004. Citeseer.

[16] Iftikhar Ahmed Khan, Robert M Hierons, and Willem Paul Brinkman. Mood independent programming. In *Proceedings of the 14th European conference on Cognitive ergonomics: invent! explore!*, pages 269–272, London UK, Aug 2007. ACM.

[17] John Laugesen and Yufei Yuan. What factors contributed to the success of apple's iphone? In *Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on*, pages 91–99. IEEE, 2010.

[18] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, ISBN 0132350882, 9780132350884, Upper Saddle River, NJ, USA, 1 edition, 2008.

[19] Cindy Norris, Frank Barry, James B Fenwick Jr, Kathryn Reid, and Josh Rountree. Clockit: collecting quantitative data on how beginning software developers really work. *ACM SIGCSE Bulletin, publisher: ACM*, 40(3):37–41, 2008.

[20] Philip E Ross. The exterminators [software bugs]. *Spectrum, IEEE*, 42(9):36–41, 2005.

[21] Jeremy PE Spencer. Food for thought: the role of dietary flavonoids in enhancing human memory, learning and neuro-cognitive performance. *Proceedings of the Nutrition Society, publisher: Cambridge Univ Press*, 67(02):238–252, 2008.

[22] Lars Vogel. Android sqlite database and contentprovider-tutorial. *Java, Eclipse, Android and Web programming tutorials*, 8, 2010.

[23] Paul Andrew Watters, Frances Martin, and Zoltan Schreter. Caffeine and cognitive performance: the nonlinear yerkes–dodson law. *Human Psychopharmacology: Clinical and Experimental, publisher: Wiley Online Library*, 12(3):249–257, 1997.

[24] Niklaus Wirth. A brief history of software engineering. *IEEE Annals of the History of Computing*, 1(3):32–39, 2008.

[25] Hengshu Zhu, Enhong Chen, Hui Xiong, Kuifei Yu, Huanhuan Cao, and Jilei Tian. Mining mobile user preferences for personalized context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):58, 2015.