Influencing Factors in the Quality of Software Development

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

August 29, 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

August 29, 2016

Acknowledgments

First, I want to thank my family who always believed in me and helped me to be the person who I am now. They taught me to work hard for what I want to achieve in my life and never give up until I reached my goals.
I also want to thank my supervisor Stephen Barrett for the support and trust in my work as well as the guiding I received during my dissertation.
Thank you to all my friends who kept me motivated and always listening and helping me with my problems and troubles I had.
Last but not least I want to thank everyone who volunteered to participate for the experiments of this dissertation.

Michael Christian Frick

University of Dublin, Trinity College

September 2016

Influencing Factors in the Quality of Software Development

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

Supervisor: Stephen Barrett

This dissertation aims to develop an approach to find factors that influence the software development process in either negative or positive ways.
Creating Software requires cognitive skills such as logical thinking, creativity or problem solving but also teamwork and good communication. However, the focus of this work lies on the influences in the cognitive performance of the developers.
Most people notice inconsistency in the quality of their work. There is always a risk that a developer is producing bad code which could lead to bugs and/or delays.
Many developers don't really know about the exact quality of their code, neither in a general perspective nor in their temporary performance. Even if they do, reasons for negative or positive changes in their code quality may not be obvious.
Software metrics have been around for decades with the purpose of evaluating the quality and the performance of the programmer but they are used primarily for project management rather than for providing feedback to the developers.

In two experiments, mobile devices are being used to collect the contextual data of the environment and the work patterns of the developers. An installed application on the device of participants gathers information from sensors and collects data which is provided

by the operating system. It accesses the light sensor, the noise level, the step counter, a 3axis-accelerometer and the location of the device. This information is then clustered and linked to a context.

Two experiments were executed to demonstrate the functionality of the developed approach.
Overall, the data gathering app generates valuable information about the environment and context. The application leads to findings that give evidence for factors that influence the brain performance for an individual participant and for patterns which could be influences in general.

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

Over the last years, the performance of computers rapidly increased and with it the complexity of the Software [45]. It 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 involved in it[6].

Different layers of abstraction from low level to high level programming languages actually make it possible to reduce the complexity of software projects. On top of that, there are libraries and frameworks that provide features that already have been implemented [30]. An encapsulation of the modules is mandatory in order to allow splitting the work in a software project. A general structure must be given to ensure that the different parts can integrate easily and to keep the code understandable.

The more people work on one project, the more important it is to provide an organized and well planned architecture to keep the code clean.

In today's world, software is everywhere! The traffic is controlled by computers as well as security systems, nuclear power plants or just a messenger app on a mobile phone etc. 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 have fired a missile without any command. Luckily it was found before a disaster happened [38].

The quality requirements vary for different software products. A crashing weather app on a mobile is not as bad as a bug that is 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 as a single person or in a team. That performance quality can change by various different reasons even a few times a day.

Previous researchers already did a lot of work in the field.
This dissertation will start with an overview of their findings which will include previous studies that investigated different theories about influencing factors on cognitive performance and related work.

This dissertation describes a new approach where sensors and information from mobile phones were used to find evidence in factors that influence the performance of developers. We developed an Android application which was installed on the participants mobile device for the experiments. The app 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.

Two experiments investigated patterns in behavior and environment that are influencing the cognitive quality of a programmer.

In the first experiment, the mobile application gathered the data while the participant is solving a provided programming task and afterwards answering additional questions.

In a second experiment, only the cognitive performance of one participant but in a more controlled environment was investigated.

The first experiment investigated correlations of written code and data which was gathered during the development. The results have shown that more participants are necessary in order to find specific influencing factors.

The individual experiment with the single participant provided evidence about influences of different music genres. Both resulting performances were also compared to a control scenario with no music played. The results for the participants arose that classical music reached the best results compared to heavy metal and the absence of music.

Also caffeine seemed to reduce the cognitive performance of the participant while, on the other hand running before solving the task showed evidence of a positive effect. The results from the experiment with a group of participants shows no distinct patterns but traces of correlations between strong changes in the environment and distraction of the participant.

In the future, monitoring of developers performance could become ubiquitous in the industry and students. It could be used conjointly with project-management tools that track the development time on a specific project. It could simultaneously provide real time feedback about the code quality itself or divergent aspects in the working place and work patterns. Rather than only comparing the information with other app users, the app could also systematically develop an optimal working environment and behavior of the specific programmer.

1.1 Motivation

This section will describe the factor that motivated the topic of this dissertation and some background information.

1.1.1 Mobile Device Sensors

Over the last decades the evolution of 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 [28]. Over the years, smart-phones became powerful 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.
The range of sensors reaches 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 [40].
Thus, 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 has a lot of potential to support and improve the people's work and environment.

1.1.2 Learning to write code

Learning to write code is getting more and more important but still not a required subject in school. It is the software that is controlling almost everything in our everyday life such as traffic, gates, calendars etc. It's failure could have dramatic impacts in peoples life.
Thus, it is very important that programmers produce high quality code and also be able to find good frameworks and libraries. 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.
Still, many programmers didn't learn coding in school or university. They taught it to themselves and might just have used it for fun-project which were not created for public usage. However, they might not really know about best practices in the industry or how good or bad his/her code really is.
I experienced this problem myself. When I started my first job as an iOS developer, I had no practical experience in writing code mobile applications. I had to maintain the current code and add new features in a big and unknown project. As I 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 a feedback tool for my code quality I could have learned a lot about the coding itself and by consequence, I would probably write much better code.

1.1.3 The importance of Software Metrics

Software is becoming more complex than ever and used in almost every environment. The deadlines in professional software projects are very strict and there is no time to develop everything from scratch. Development teams depend on libraries rather than reinventing the wheel over and over again.
As already mentioned in the previous section, a lot of people do programming for a hobby where the quality and performance doesn't matter to them as much as in a professional environment. They also create libraries and frameworks with their quality standards. The increasing amount of open source libraries and dependencies make it very hard, but also very important to professional developers to be able to trust the quality and functionality. At this point of time, the only indicators are user ratings and the amount of times it's been used in different frameworks and projects. Some few frameworks are also recommended in public reviews articles.

Frameworks and libraries also need to be dynamic and maintained. Operating systems and programming languages are being updated more frequently which requires fast changes. A constant measuring of quality at the first place could give direct feedback that the quality stays constant after changes and as well helps the developer to create a better structure and code to improve the maintainability from the beginning.

The most used platform in the open source community is Github. Github is based on git and is a web server that can be used to host software projects and allow to make them accessible to others developers [7].

1.1.4 Working environment

A new trend, especially in the tech industry shows that company move from common clean looking office spaces to colorful creative environments. Companies like Google or Facebook seem to reduce 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, having a beer with the co-workers after office hours or meeting the colleagues for a ping-pong match during the day was unthinkable.

Google tries to motivate developers to communicate more with the team by placing the whole team in relatively small spaces and provide separate 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 this dissertation, we are trying to find patterns between working environment, behaviors and the resulting code quality in learning and professional environments. Also, 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.

1.2 Aims

The goal is create a solid base and a working demonstration of a system that can gather data from a mobile device and show the significance of influences the quality in software development. This work and framework hopefully inspires and helps others to do more work in this area to bring code quality 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

The next chapter summarizes the related work in the area of software metrics, measurements and analysis, data gathering, data clustering, influences in cognitive performance and software quality.
Ensuing, we describe the design of the different software components to gather the data, provide information to the participants and to ensure the privacy of the gathered information.
Chapter 4 contains information and the process of the implementation of the different software components. The experiment is being described in chapter 5 and includes the setup and execution, the expected results. the next chapter characterizes data classification as well as the questions that are being asked to the participants.
The last two chapters describe and interpret the results and conclude the subject and information of the experiment. The appendix contains an overview of the abbreviations, links to the source code and more details about results of the experiments.

Chapter 2

The State of the Art

Many researchers are concerned with finding metrics of software quality. The human factor is one of the most significant factor in the development process. Therefore, the related work in cognitive performance and influences is an important area and related knowledge to find factors in the software development.
This chapter provides a summary of their previous research and related work.

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 [22] [30].
Also project managers have a great interest in details about the progress and the products quality in order to coordinate the schedules, resources and having 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 [22].

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. A lower computing power in that days made computers 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 development 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 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 has been used since more then 30 years. Today, many metrics are still used to investigate the productivity of the software developers. The amount of bugs in relation to the amount of code, the initiall 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 [25].

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. Maintenance and re-factoring is not as important in academia as it is in the industry with commercial software. After all, Software metrics in the industry are primarily used for the management rather than for the development process itself.

Industrial software metrics can be used to ensure quality, productivity and can even make predictions of the software quality and software reliability [9].

Several researchers investigated and developed approaches to improve the metrics and the results which they are generating. Yue Jiang et. al. [21] from West Virginia University researched methods for improving software quality predictions. They used supervised machine learning algorithms with datasets and focused on improving of the information content of the training data in their research. Their results showed evidence 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.

2.1.1 Summary

Software metrics are values that indicate the quality and performance in software development. It is more common in project management for measuring the progress and for making predictions rather than for improving the development process and giving feedback to the developers.

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 [14]. Over the last decades, the University of Hawaii did a lot of research in PSP and they developed different approaches to bring students to adopt and use it in their projects and even later in their profession as a developer.

Their first approach was originally described as “A Discipline for Software Engineering” [18]. 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 [24]. 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. Table 2.1 shows the evolution of the PSPs from the University of Hawaii.

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 [23]. 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 behavior in order to find good practices 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 [32].

Characteristic	**Generation 1**	**Generation 2 - Leap**	**Generation 3 - Hackystat**
Collection overhead	High	Medium	None
Analysis overhead	High	Low	None
Context switching	Yes	Yes	No
Metrics changes	Simple	Software edits	Tool dependent
Adoption barriers	Overhead, Context-switching	Context-switching	Privacy, Sensor availability

Table 2.1: University of Hawaii - PSPs

2.2.1 Summary

This section describes the evolution of analyzing software engineering processes. It started with documenting every step by hand up to fully automated plugins that gather and analyzed the data without any work of the developer. The section ends with an example of a study that was executed with a data collecting and analyzing tool and its results.

2.3 Mobile Data Gathering

Ferreira, D., et al. [10] 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. [47] 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.

2.3.1 Summary

This section describes two different approaches of using the mobile phone for gathering the sensor and device data. The first approach is using an application for that purpose while the second research team was reading the device logs which required physical access to the mobile device.

2.4 Data Clustering

In order to give data a meaning it makes sense to cluster in a logical way and make assumptions about the needed features or contexts. The article "Data Clustering: A Review" [19] provides a wide overview of different techniques and ways to classify data into groups. They describe a variety of different clustering techniques, all with the goal to find patterns and assign data to a specific category that allows to work it. The clustering can be be distinct between hierarchical and partitional techniques. Different from a hierarchy, the partional methods don't produce a hierarchy but simple partitions. In their paper they describe different techniques for clustering different kinds of data such as image segmentation, object recognition, document retrieval and data mining. Different techniques can be useful for different kinds of applications and therefore no perfect solution for everything is found. In another study, researchers from the University of Ouluu, Finland used the data of a wide range of different sensors to detect the context of the user [27]. They gathered data from a microphone, a three directional accelerometer, thermometer, light sensors and measured the humidity and skin conductivity. they were using naive bayesian classifiers to combine the gathered data and correlate them with samples. For example, they recorded the ambients of different environments such as being in an elevator or the sound of a car or conversations. They got the features from their audio files by using algorithms from the MPEG-7 standardized metadata. Different environments have different key features in their ambient, such as constant noise (e.g. tap water) or peaks(e.g. conversation). Audio was their most valuable information, but for example the humidity was helping to detect very accurate whether the user is in- or outside. The results of the experiment show a very accurate detection of the correct context in 2.1. Features like being inside or outside as well as detection whether rock or classical music was played were detected very well. Detection between walking and running or active and still were less accurate. The combined true positive rate was more than 90% and the true negative value was over 85%. Their usage of good test data placing their sensors in a good way helped a lot to get good results. In reality, when users have phones, they carry them in purses, pockets or their hand which makes it much harder to detect the current context. Context aware computing was already the topic of a great paper in 1995 written by a team from the Columbia University and the Xerox Corporation [39]. They describe a context aware system for an office environment. The systems uses the user's location, lightning, communication bandwidth and proximity to other users within the office in order to customize the application functionality depending on the context. This could for example include the displaying of experiment information, when the user is in his lab or showing the calendar when another person is in a close range. It can also be used to create context based reminders for specified locations, time, when seeing specific people or all combined.

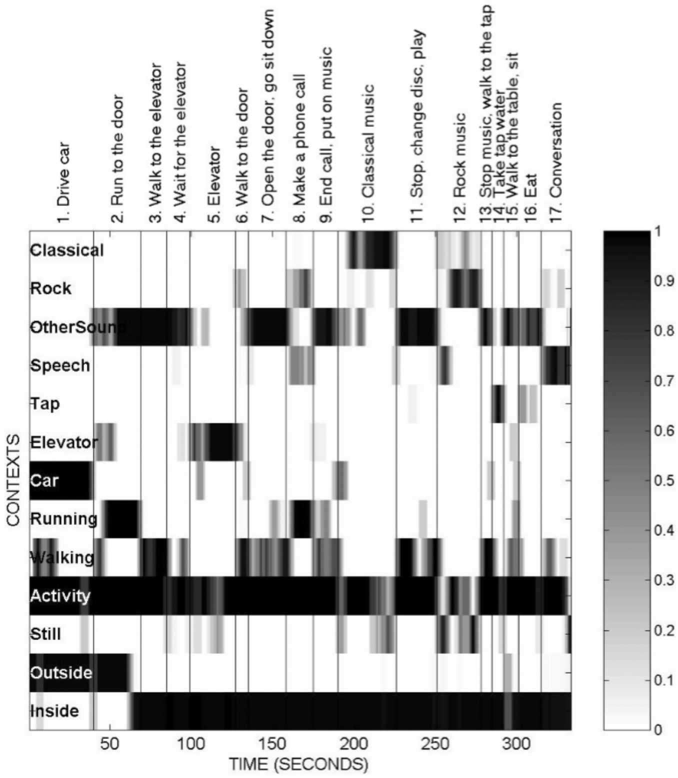


Figure 2.1: Data Clustering [27]

2.4.1 Summary

In this sections, approaches and ideas about bringing data into context are described. Clustering techniques are mentioned as well as different approaches to classify the information by comparing them to patterns. Sometimes single sensor-values are enough to be able to cluster properly to identify the meaning of the data. Though, mostly it is the combination of different sensor-values that give more clarity about the context.

2.5 Variable Quality Influences

The majority of software quality is based on the cognitive performance of the software developer and the communication within the team. When the single developers write brilliant code, but don't know what the others do or need, the code can't work. On the other hand, the code quality still stay bad even when the development team communicates perfectly but the individual programmers write bad code [31]. The following sections will provide an overview about the previous research of the individual factors in that area.

2.5.1 Team Communication

The differences between teams with high cooperating team-members against project teams with less communication have been investigated an discussed by Mary Beth Pinto and Jeffrey K. Pinto [33]. They tested the two different groups on performance in tasks and the psychological outcomes.
Several different factors have been tested and resulted significantly better in the high communicating group. They scored higher in resolving problems, brainstorming, progress review, obtaining information, gaining authorization to perform tasks and in receiving feedback. The low cooperating team did only get a better score in resolving conflicts, which is not surprising as fewer communication already avoid conflicts.

The importance of the communication in software teams is gaining more and more attention from companies within the last few years. The companies and teams came up with several ideas to improve the internal communication. Agile software engineering methods is the solution for a lot of teams and companies to reach the goal of a better exchange of information within the company. The concept is based on flexibility and responsibility within a software project.
Instead of having a the whole project scheduled and structured at the beginning, agile concepts allow to react to problems and new information in a faster way [5]. One of the most used and successful methods to work in agile teams is scrum.

In scrum, the tasks are separated in different phases that are called sprints. A sprint is a short time period in that a defined goal should be reached. This goal can be, for example a feature or a new component. After the sprint the team comes together again and decides about the next sprint and defines the next realitic realizable goals. I this way the team has a lot of responsibility about the project and a lot of freedom how they reach their goals within the sprint period. At the end of each sprint or a defined period, the team comes together for a retrospective to discuss the last sprint/s and how to improve the processes in the next period and if they change some methods such as the daily meeting. The daily meeting is done buy some scrum teams, where every team member summarizes the achievements and problems from the previous day. This meeting can be useful or just wasting time. In order to find the best working and management patterns the teams can test different methods and discuss them in the retrospective. This dynamic changing and regular feedback is one of the reasons why scrum is used more and more in modern software teams [36] [31] even with downsides that the company needs to increase the trust in the employees and give up some control [35].

Another problem with the communication in teams comes with the increasing globalization and the internet. The ability that and employee can work from every part of the world with an internet connection brings the disadvantages that the software developers do not necessarily sit in the same room anymore or have their working place within a walking distance. Also allowing homeoffice for a few days a week is an option that employers provide their employees in order to be a more attractice and family friendly oriented. These changes also requiere new techniques to communicate within the teams. Communication can be done by using video conferences or email. However, both techniques have their disadvantages. A videocall needs to be scheduled and requeres a good internet communication and the problem with emails are the delayed response times and it is too easy for others to ingnore an incoming email[4].
One possible solution is chat software which finding their way more and more in the daily communication in software development teams [20].
Slack is the most used tool for chatting at work. The great success in these new way to

communicate shows in the rediculous growth. The company Slack is the fastest growing Startup in the world. After just twenty month after its launch in February 2014, already more than 1.7 million people where using Slack [3].

Summary

This section describes the research in teamwork and communication. I was found that a team with more communication created better results than a team with less communication. In order to improve communication new agile methods such as scrum were introduced. This section also mentioned the problems that employees are not necessarily working in the same office all the time. The latest approaches that deal with this new problems are for example chat software or video calls.

2.5.2 Cognitive Performance

Looking at the individual programmer, the most important factor is obviously the cognitive performance of the individual person. This section shows the research in the circumstances and influences that can impact the performance in a long or short term.

Working Environment

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.

Context Switching

Devin G. Pope and Ian Fillmore from the University of Chicago [34] inspected correlations in cognitive performance of students and the time between written exams. Depending on the schedule of the examinations, students from one year have a different amount of time between exams than students in different years. As a comparison they name the example of physical performance. If the body has a longer time to recover from one task, it performs the seconds task better compared to a shorter recovery time between these two tasks.
In this article they compare the scores of the students in their exams and the amount of days between the examination days. The study involves information about the students as class(Senior, Junior, Sophomore), Gender and their Race. They all were writing Advanced Placement (AP) Exams in the USA. Their results show that a longer break increases the probability that the students pass the exams by 6-8%. The increasing of the performance is linear up to 10 days.

As one of the possible reasons for the outcome the researchers names fatigue which is caused by the exhausting task of studying and writing the exam. Another theory is that the last-minute preparations are important for good results but harder to realize when exams are closer together. Rogers and Monsell from the University of Cambridge [37] executed an experiment to find out how a context switch can influence the performance on cognitive tasks. It showed that a frequent context or task switching has a negative impact on the error rate and the reaction time of the participants for the tasks they did. Repeating this experiment for three days yielded that the practice has no positive influence on the error rate and thus shows as well that context changing is negatively influencing productivity and performance.

Arousal Effects

The cognitive performance can vary based on the context and the environment. When the body is in a relaxed state, the mind also slows down to save resources. It made sense back in the stone age because thinking was not as important as today. Cognitive performance was mainly needed in dangerous or unusual situations where the heart beat is faster to provide the brain and the muscles with more oxygen and a higher arousal than normally.

Researchers from the Brunel University in the UK showed movie clips to participants in order to invoke different defined moods before the participants had to solve given debugging/ coding tests. The results showed improvements in their score after the participants were confronted with high arousal video clips. Low arousal clips affected their performance in a negative way compared to neutral clips [26].
It is called the Yerkes-Dodson Law, which proclaims that a higher level of arousal leads to better cognitive performance. As caffeine also influences the arousal, it also can be used to boost the cognitive performance and is not just helping to wake up in the morning. Watters, Paul Andrew et. al. [44] found out that the average caffeine for the best

cognitive results is an amount of 400 mg for one person. That is the amount that is contained in ca. 5 espresso shots.

When an arousal stimmulation can be 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. [8] found out that the sunshine alone actually has no notable effect in the mood of the most of the people. Certainly, some individuals have a so called seasonal affective disorder (SAD). Their mood is indeed strong being affected by the seasons with fall and winter depressions.
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.

2.5.3 Activity

The researchers Hillman C, et al. [16] found evidence for positive effects of regular activity on cognition and brain functionality for human and animals. They found that especially aerobic has a strong positive influence. A meta analysis showed that children who were physically active were better in all tested categories (IQ, perceptual skills, verbal tests, math, memory, academic readiness and others). These effects were also shown in other age groups but were the strongest for children. Older people who were active during their life showed a smaller risk of Alzheimer and Dementia.

Diet

A very different, but probably the most important factor in the long term cognitive performance are temporary diets and the consumed food during the lifetime. 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 healthy ageing [42]. Some eatables demonstrated positive effects on the mental performance when they were containing flavonoids like for example grapes, tea, cocoa and blueberries. Different to the previous influences, the diet and the lifestyle are less obvious in their consequences. Their impact is slowly showing over several 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. These foods also decreases the chances of Alzheimer and dementia. Other studies have shown that flavonoid-rich groceries 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.

Summary

This section is about the different influences in the cognitive performance. Starting with the influences of the working environment created by the company with the stress and interest it creates with the projects itself as well as motivation and creativity by giving employees the chances to share their ideas and feel valuable. The nest part investigates the problem and the lower performance that occur when people switch during different tasks and contexts. Afterwards the work of influence of arousal in cognitive performance is summarized and the section end with the influences of physical activity and the diet and the food that has positive impacts.

Chapter 3

Design

The following chapter details the design of the data gathering Android application. It will also outline the sever side implementation to compute, store and provide information. Afterwards follows a short description of some additional tools, needed for the data analysis.
First, it will start with a brief description of the two components and will follow with my design decisions and my reasons for the choices.

3.1 Approach

This approach has been created a part of an idea for a system that helps to improve the working behavior and environment of developers. The whole system is described later in the section 'Dynamic Code and Developer Analysis'. In this dissertation we created a toolset that allows to gather data from the mobile phone of the participants, analyze it and correlate it with the code quality of their work. The technical design for this approach is created for the experimental data gathering of a limited amount of participants. The described system is not capable to handle high traffic and to analyze the data entries of great amount of users.

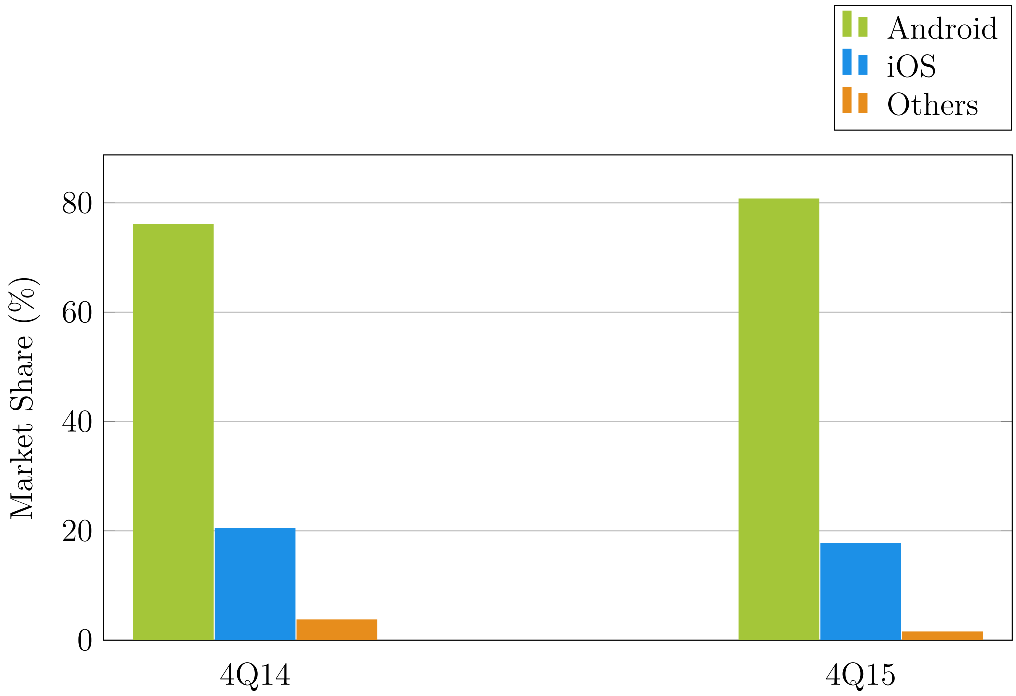


Figure 3.1: Smartphone OS Marketshare

3.2 Functionality Overview

The name of the application was chosen to be ”Dather”, which is a a combination of ”Data” and ”Gather”. The purpose of the application 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.

3.3 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 3.1). The trend also shows that the number increased from the last year [11]. Therefore, we decided to realize the mobile application implementation 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 [17].

The Android application make use of its build-in sensors and information provided by the Android operating system 3.2. Different than iOS, Android is an OS that can be installed of different devices from various manufactorers and with different hardware components [12]. Thus, the buit-in sensors which are clustered in motion sensors, environmental sensors and position sensors [13] can differ between the diverse devices. Components that are required for standard functionality such as making phone calls are more common than other sensors. For example, 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. Furthermore, the different mobile device manufacturers customize the open source android operating system for their devices and their purposes. This nature can lead to different settings or behavior for the hardware and also software components.
This device diversity and the variety of different screen sizes and aspect ratios make it more difficult to support a range of android devices and make sit impossible to test all devices that are running a specific operating system. Apple at the other hand uses the same aspect ration up from the iPhone5 and supports the hardware components in a

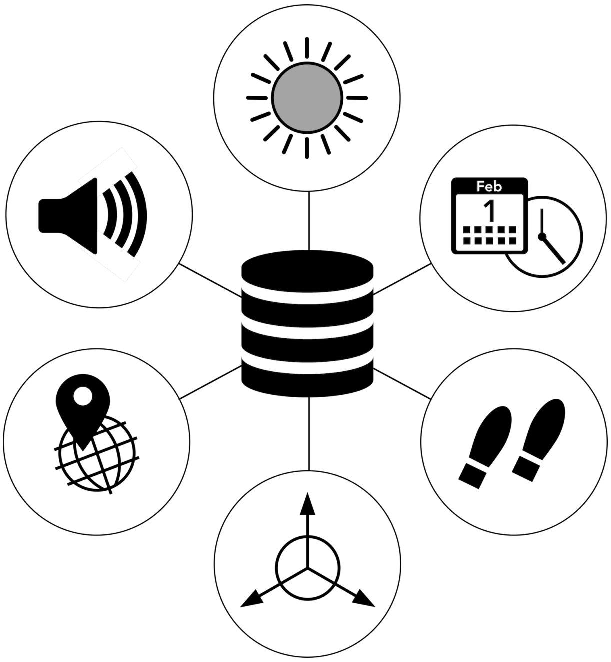


Figure 3.2: Gathering: light, timestamp, steps, accelerometer, location, volume

similar way in iOS.

3.3.1 Requirements

The application is primarily created for the research project and therefore not for the everyday use. The participants should not waste much time in finding out how the app works. The goal was to create a simple and intuitive interface and a leading flow trough the functionality. The apps purpose is to gather the data of the participant during coding. That includes the usage of the mobile device during work. Thus, the gathering app must be able to run in the background, so the participant can use the app as he/she would normally do (e.g. listing to music, texting etc.)

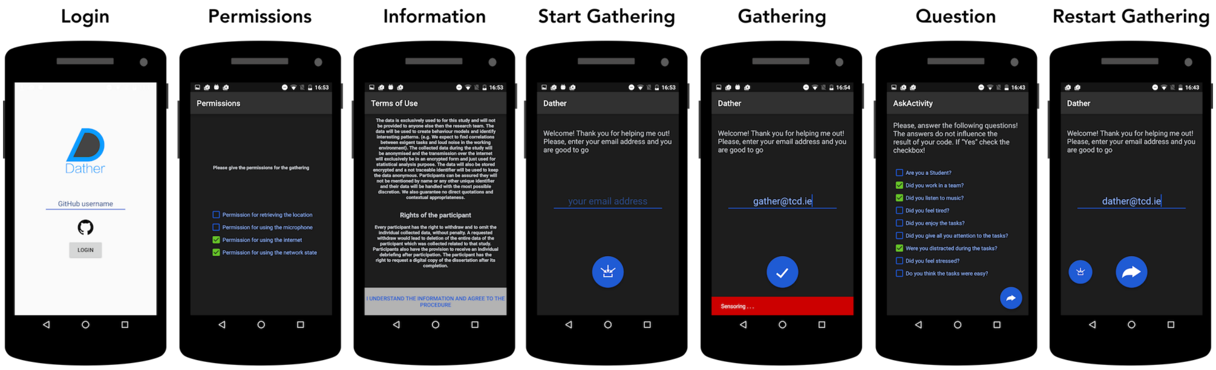


Figure 3.3: Android Views

3.3.2 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.

3.3.3 User Interface

The Application has two main user Interfaces, the gathering view and the question view. The gathering view is the control interface for starting, stopping and transmitting the data gathering. The appearance of the user interface changes related the current state and the logical functionality. This view also displays the username and a short informing text.

After successfully 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 3.3 you can see the flow of the views in the applications. It shows screenshots of the flow within the app from left to right. Starting at the login view, where the participant enters his/her Github username. The next view asks for the permissions followed by the view that informs about the sensible data handling. In the middle you can see the gathering view, where the user controls the process. The right screenshot shows the gathering process in action. The next view contains the questions to the user. The last view shows the gathering view with a changed functionality. Here the user can gather data again after completing the first gathering process.

3.3.4 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 [43]. 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.

The database has a capacity of 1024 MB which is equal to 1,000,000 KiB and can store 2,380,952 entries of an average of 0.42 KiB (this value is calculated based on the used storage of 1,000 entries). Assuming that the maximum gathering get 30 entries per minute, the database can store 7,9365.07 minutes or 1,322.75 hours of the gathered data.

bigbreak 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.

3.3.5 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.

3.3.6 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.

3.4 Server Design

The server is hosted by 1&1 Internet SE as a completely pre-configured server back-end 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.

3.4.1 Requirements

The Website is required to provide all the information a participant could need to do the experiment. The Interface should be clean and the participant should be also able to download the android application from the same website as he/she gets the information about the experiment. The back-end will only be used by the researchers and therefore the user interface doesn't matter as much as for the participants. The functionality and the customization are the most important features.

3.4.2 Server Front-End Design

The information for the participants of the study can access information web pages and access the downloadable Android application.
The front-end 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.

3.4.3 Server Back-End Design

The back-end 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.

3.5 Tools

The following tools are were used during the experiment primarily for the decryption of data to automate some processes.

3.5.1 Requirements

The tools are also just for the usage of the researchers and the requirements therefor also the functionality, the security that no information are getting lost.

3.5.2 Decrypting 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.

3.5.3 Data Structuring Tools

For working with the resulting data from the experiment, a tool-set was created to take over specific tasks. All the tools for this purpose are implemented in Python because it provides a very handy way to work with external files. The first tool allows to extract only the entries from a specific participant by providing the his/her Github username.

Another tool extracts the single values per entry with its timestamp and the last tool averages the results per minute and converts the data into a form that can directly be used in latex to draw plots. The last tool calculates the standard deviation and the average of the values.

3.6 Dynamic Code and Developer Analysis

The work in this dissertation is part of an analysis system for code and developer. The limited scope of the dissertation didn't allow to target the whole idea which will be described in this section.

The idea for the full approach is a system that constantly gathers data from the developer.

The system provide the developer real-time feedback on the code quality and ways to improve the performance.

Many developers are already using applications to track their time on projects. Our approach could combine the time tracking and the performance analysis and thus avoid more work for the developer.

In order to reduce the problem of the limited and different sensors in the variety of mobile phone, an additional hardware device (base station) with a collection of sensors, placed on the developers desk could gather the main environmental information. The mobile phone would transfer specific data, that only the mobile phone can gather to the base station. This data could be for example the location and the rotation of the mobile device. The base station collects all the data and could also request additional information from various sources in the internet such as the weather based on time and location.

The combined information constantly analyze the gathered data and the latest commits of the source code and compare the data with the results of other developers. Furthermore, automatic learning algorithms could compare the gathered data and the code of the developers to find more specific trends and influencing factors in the performance for individual developers. Last, notifications sent to device of the developer could provide feedback and tips for improvement, either in the working patterns or environment.

3.7 Summary

This chapter is about the design decisions of the different software parts that are needed for the experiments. First, the functionality of each part are being described, followed by the part about the Android application. Android was the mobile OS of choice because it has the highest market share of all mobile operating systems. The app needs to be simple and easy to use for single usage which is seen in the UI and the UX specification. The app was programmed in Java using the MVC design pattern while using an SQLite database for the permanent data storage. The app-server communication is been realized using a REST-API and in order to do nothing against the users will, the user needs to grand permissions and accept term to be able to use the app.

The Server back-end is implemented in PHP with an MySQL database while the front-end is implemented in HTML5 and CSS while the decryption tool was created in java and the rest tools in python.

Chapter 4

Implementation

This chapter is about the implementation of the main application for Android, the server back-end and webpages. It also describes implementation of the tooling for working with the data.

4.1 Android

As mentioned before, the Android Application (Dather) is for gathering data of the user and get environmental information. For This purpose we developed 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. [47]. 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 no possible, using an official way, 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 for 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.

4.1.1 User Interface

The focus in the implementation if the user interface was to create a simple and intuitive user experience. As the participants will use the app probably just once, the interface must be as simple and intuitive as possible while reducing the possibility to make mistakes. In order the achieve that, self-designed Icons are being used to simplify the handling and the available functionality is limited to a logical order. For example, it is not possible to send data to the server without any previous gathering process. The experiment showed that the participants felt comfortable to use the interface itself. In order to help the participants during the experiment, the project website [1] provided a walk-through guide through the experiment. Though, the participants didn't use these information source and tried solving it on their own or where asking for personal assistance. Therefore it is even more important to provide a clean interface and reduce the possibility of wrong actions as much as possible.

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.

[1] `http://frickm.de`

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.

Login View

The login view only shows an input field and a login button. The input field requires the Github username of the participant in order to login. After taping on the login button, the usernames existence is been verified by the Github API.

Permissions View

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

Information View

This view contains a scrollview with an HTML-formatted text. A button is located at the bottom of the scrollview. A tap on the button the user confirms that he/she read and understood the displayed information. The user is then forwarded to the next step which is either the permissions view or the gather view.

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.

4.1.2 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.

4.1.3 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

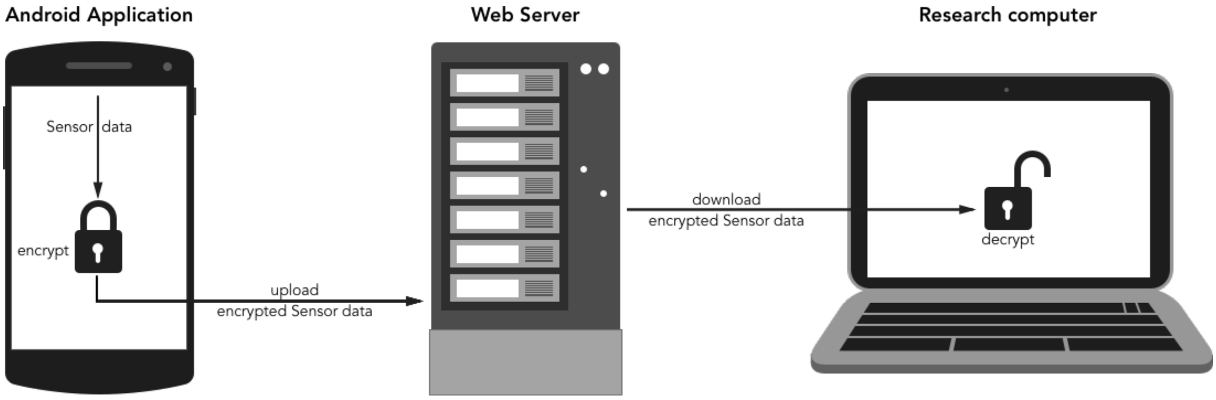


Figure 4.1: Security Dataflow

access to the database from the applications.
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.

4.1.4 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 of the participants Github username. There is no known way mathematical to recover the original email address in feasible time from the hashed 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.

4.2 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. This chapter will be about the back-end and the web-pages because the Database design is already described in the previous design chapter.

4.2.1 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.

4.2.2 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 [2] which are dynamically being loaded at the page load or from the browser cache.

4.3 Tools

The development of the following tools was necessary to work with the data that can be downloaded from the MYSQL database.

[2] `https://fonts.google.com`

4.3.1 Encryption Tool

The encryption tool is a Java application 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.

4.3.2 User Separation Tool

This and the following three tools are written in Python. This tool reads the decrypted text file that has been created by the java decryption tool. First, it creates a SHA256 Hash from a Github username and compares the entries of the text file with it. It only takes the matching entries and writes them to a new text file.

4.3.3 Value Separation Tool

This tool can be used to define which of the entries are needed to work with. For example the user can decide just to create an output file with the latitude of the participants location. The selected data and its timestamp gets written in a new text file as well.

4.3.4 Latex Plot Syntax Creating Tool

This little tool is calculating an average for every Minute of the timestamp of the read text file. As the gathering saved a value every few seconds, it makes no sense to display all the values in a plotted graph.

The output contains the timestamp with the value for every minute in a syntax that can directly been interpreted by latex and the pgfplots library.

4.3.5 Map to Duration Tool

This tool calculates takes a duration and maps the current measurements on that specific duration. That allows to compare the results in a graph independent of the duration the participant needed to solve the task.

4.4 Summary

This chapter describes the implementation of the software and tooling for the experiments.

The Android app contains of:

- Login View - verifies username with Github API
- Permissions View - asks to grand permissions to app
- Information View - shows terms of the experiment
- Gather View - control center for the gathering process (start, stop, send, restart)
- Question - Asks participants questions and sends to server

The next section describes the data gathering in process in detail followed by the implementation of the data storage within the app. The app uses a hybrid encryption using AES and RSA.

The next part will point out the simple PHP implementation with the database connection of the Server back-end. For the front-end we used standard HTML and CSS components and Googler web-fonts. The decryption tool uses RSA and AES for decrypting the data. The other tools read a textfile, manipulate the data and write the results in a new textfile.

Chapter 5

Experiments

This chapter describes the details about the two different experiments. These experiments should demonstrate the way how such a system can monitor and analyze the development in order to improve it and find weaknesses.
The goal of the first one is to to find correlations between the gathered data from the mobile devices and the code quality. The second experiments purpose is to find individual factors that influence the cognitive performance of a single person.

This chapter shows the execution of the experiments as well as the usage of the gathered data and how the data is been interpreted.

5.1 Group Experiment

Five participants solved a programming task while the mobile application was gathering the environment and working patterns. After completion, they submitted the gathered data to our server and deployed their solution in a public Github repository. An overview of the experiment procedure is shown in figure 3.1.

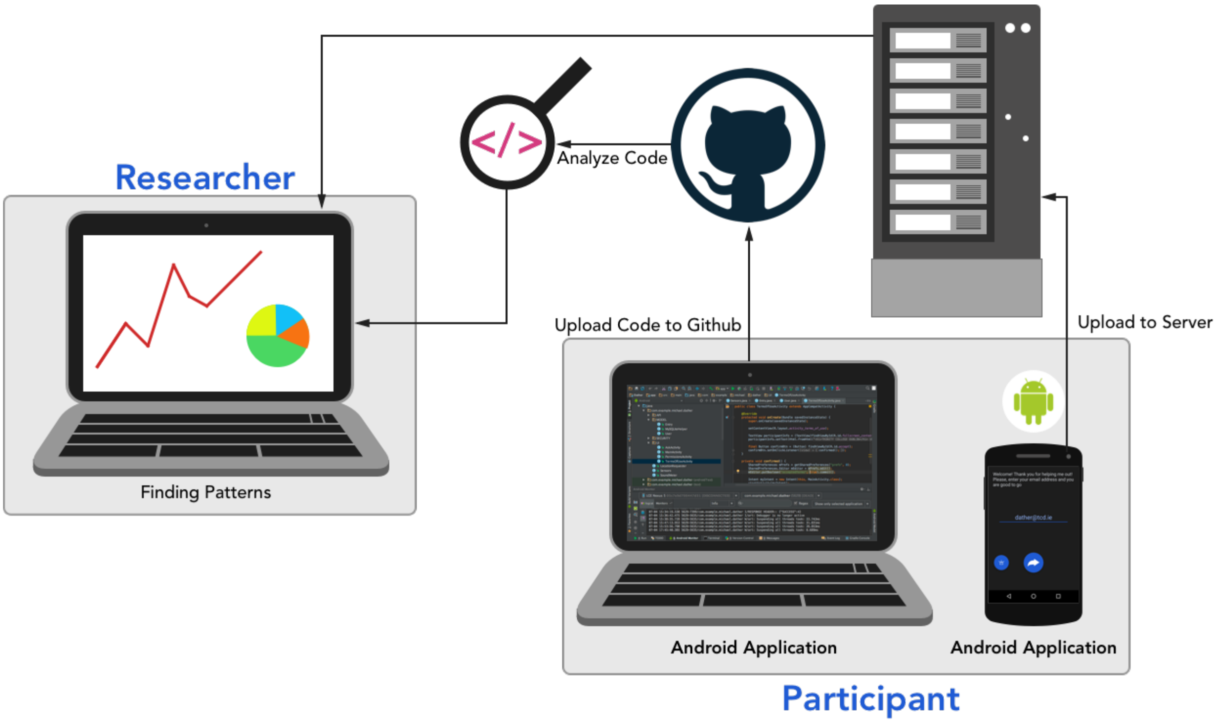


Figure 5.1: Experiment Execution

5.1.1 Setup and Execution

Every participant needed to install the 'Dather' app on a mobile phone with at least Android version 4.4. They downloaded the app from the project website [1] by accessing the URL from the Android device. The website also provided information about the setup and usage of the application.

At the first start, after installing the downloaded app, the participants needed to login with his/her Github username, followed by granting the permissions to use the sensors and access the required device information.

After setting up the application the experiment is ready to start. The participant runs the gathering process while working on the programming task as described in the next section .

[1] `http://frickm.de`

After completing, the participants uploaded their solution code to Github and provided a link of the Github repository to the research team. The Github account name was later be used to match the gathered data with the uploaded solution-code of the participant.

5.1.2 Programming Task

The programming task for the participants has been provided via the project website [2]. The Participants were asked to create a solution to calculate the number of character from a string, that can not be used to create a palindrome [3]. The whole programming task with examples can be found in the appendix of this dissertation. The participants could use their favorite programming language to address the problem while they let the android application gather their data.

The experiment showed that the participants had some issues understanding the explanation of the problem. The example brought clarity and helped to understand but wasted time of the participants who partially read the question and the next steps while they were working on the task.

5.2 Individual Experiment

The purpose of the second experiment is to find evidence of specific factors that influence the ability of cognitive thinking. Different isolated scenarios are been tested by a participant in order to find correlations between the specific environments. This experiment allows to test the factors in a more controllable environment but based on one individual person.

[2] `http://www.frickm.de/codingTask.html`

[3] A palindrome is a word which reads the same from left to right and right to left such as anna or racecar

5.2.1 Setup and Execution

In this experiment a participant solved some cognitive tasks while being in a controlled environment in order to test the performance influences of isolated factors. Of course it is very unlikely or even impossible to test a factor in complete isolation one factor. There are always side factors that which are unavoidable. They could be for example the human itself, sudden unpredictable changes in the environment and of course the problem in keeping the factors of one part of the measurement equal to the factors of other measurements. To minimize these factors, the 'Dather' Android application helped to monitor the environment and remove recorded tasks where the environment information are too different of results which are were correlated with each other. However, with this problems in mind, the idea to measure changes in the cognitive performance of a person, was measuring the time of finishing a Sudoku game. The game, where the goal is to systematically add missing numbers in a 9x9 matrix, requires concentration and logical combining of numbers. The sudoku game was already used in previous research for measuring the cognitive performance [41] [46]. Another reason for using Sudokus is that they can be randomly generated with a specific calculated difficulty level to make sure that every Sudoku is equally hard to solve. A website [4] generated the Sudokus uses an engine which is part of the gnome-sudoku software [5]. A medium difficulty level and a limited calculated range of difficulty to +/- 0.02 of 0.5 was the base for generating the Sudokus which were then printed on paper, one per page.

5.2.2 Scenarios

The following scenarios have been tested. Each scenario was performed 10 times to get a good mean which decreases the randomness in the experiment. In order to control the environment variables, a modified version of the Android app recorded the environmental

[4] `http://www.opensky.ca/~jdhildeb/software/sudokugen`
[5] `https://sourceforge.net/projects/gnome-sudoku`

light and volume and it was made sure that the values don't differ much to have a influence in the results. The experiments were executed over a several days in mixed up order to avoid that the training-process in solving the Sudokus can also influence the overall average of the outcomes.

Music

The scenarios to compare in this part the influence of two different types of music and as a control scenario no music at all. The participant did the Sudokus while listening to Spotify-Radio [6] 'Heavy Metal' and 'Classical' over headphones on a defined level of volume. In the control case without music, the participant was not wearing headphone but working in a very quite environment.

Coffee

In this scenario we wanted to test the influences of Coffee in the cognitive performance as discovered by Watters, Paul Andrew et. al. [44]. Simultaneously to their results we used a caffeine level of almost the value that they found out is the optimum for cognitive performance (400 mg). The whole experiment was executed in 5 days in a row with two tasks before, and two tasks after having a coffee. First, the participant solved the Sudokus without taking any caffeine for more than 16 hours, which is more than enough to make sure no other caffeine intake can influence to experiment [29]. Additionally the participant had the same breakfast every day before every experiment. For the second part of the experiment, the participant had the coffee drink. The Coffee Franchise declares one espresso with 75mg caffeine each, which sums our drink up to 375mg at an amount of 5. After having the coffee, the participant waited 40 minutes for the caffeine to be absorbed [29] and started with the Sudoku.

[6] `https://www.spotify.com`

Running

This scenario compares the Sudoku result from before and after running for 30 minutes at a speed of 10 kph in a gym. 10 minutes break are between finishing the run the beginning of solving the Sudoku. The 10 kph for 30 minutes was a duration which was very exhausting for the participant during the test. Hillman C, et al. [16] found evidence for long term improvement of the cognitive ability now it is to find out how activity up to a level of exhaustion influences the brain performance. A possibility would be an increase of the performance related to the supply of more blood and oxygen while the heart beat is significantly faster during activity it would also be possible that the exhausted body enters a state to save energy after high activity and decreases the energy and heart rate.

5.3 Summary

Three experiments are described in this section. The first experiment gathers data while participants work on a programming task. Second, a participant is solving 10 Soduku riddles for each isolated environment (normal vs. after drinking 375 ml caffeine, silence vs. classical music vs. heavy metal, normal vs. after running for 30 min) and being compared to the its counter parts.

Chapter 6

Classification

In this Chapter, the classification of the gathered data is been described. That means that the data is interpreted and put into a context which can be used to generate results in the next section.

6.1 Context Classification

In order to understand the gathered values from the sensors rather than just using them, it makes sense to interpret them and bring it into a context. Previous research results and also classifying controlled tested events using the gathered values will be described in further detail within the next paragraphs.

Indoor Outdoor differentiation

The brightness of indoor lightning is different from the brightness outdoors. Indoor environments are mostly receiving light from an artificial light source which flickers in a rate than can't be noticed by the human eye. Sadly the light sensor of the mobile devices is not precise enough to detect that flickering. Anyhow, also the luminance is different indoors and outdoors. Indoor lights are just not as powerful as the sun and it would require a ridiculous amount of artificial light sources and windows to create the same brightness within buildings as they are outside. As seen in the two tables 6.1 and 6.2, based on the lux from the light sensor it is possible to give an educated guess whether the mobile device is indoor or outdoor.

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 6.1: 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 6.2: Common & Recommended Indoor Light Levels

Movement of Mobile Phone

The Y and Z axis of the 3D accelerometer can be used to detect whether the participant moves and uses the mobile phone.

The values from the 3 axis can give an indicator how the participant interacts with the device. For the results, we want to know, how many times user picks up the mobile phone and how long he/she was interacting with it. That behavior, as shown in the graphic 6.1 detects changes between the mobile device laying flat on the desk and the device being in a vertical position by the changes in the rotation of the z- and y-axis.

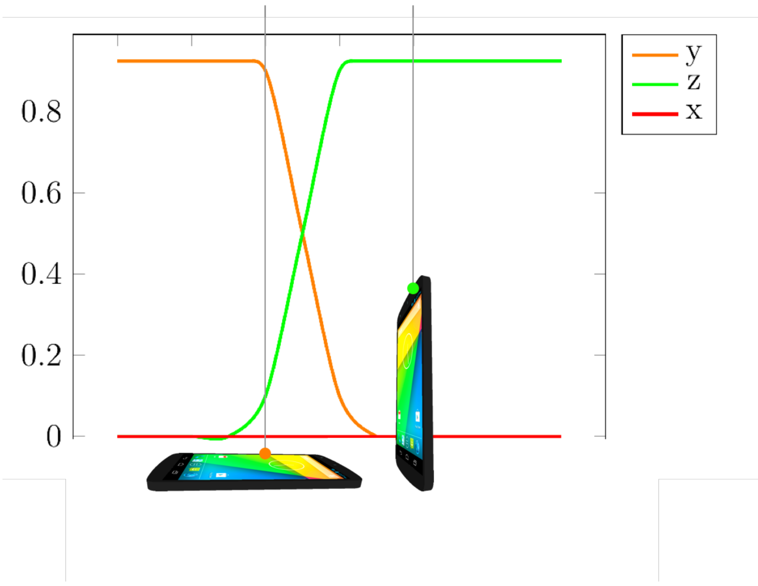


Figure 6.1: Device Rotation

Location

In order to detect the location of the user, the location with the environmental noise as well as the detection whether the user is indoor our outside. The location accuracy depends on the way how it is been calculated which is ether the network or GPS. However, it can vary and can't ensure a the exact location but also using the noise and information from the light sensor can help to limit the results to fewer possibilities.

When, for example the location shows a radius in an area with a library, a coffee shop and a public crowded square it's a high chance that the library is not an option in the case of a noisy environment. In order to detect whether the user is in the coffee-shop or the square, the light sensor can detect whether the light value is in the outdoor or indoor brightness range.

Movement

The movement of the user can directly be seen by the steps he/she walks during the start of the gathering until the ending. The distance and the frequency shows if the user just walks to the fridge, toilet or somewhere close or actually walks from one place to another. Also the locations can indicate that.
The location can also show whether the user was on public transport, on a train/car or an Airplane depending on the travel speed and from where the user started and where he/she arrives (airport, garage, train-station etc.).

Weather Conditions

With the location and the timestamp of the gathering and it is possible to get information about the local weather of the users location at the time when the gathering happened using the timeanddate-website [1].

Dynamic

The dynamic in values is the way it differentiates in itself. This information can be used to detect changes in the noise level and environmental light. For detecting a range within the dataset a range must be set which value are within a normal range and which are

[1] `http://www.timeanddate.com/weather`

falling out. One way to calculate a range is to use the standard deviation + and - to the mean of the values.

Music

Using the environmental noise it is possible to find patterns that can be related to music. In general modern music has a very constant noise level rather than the dynamic classical music. The iTunes top 100 [2] songs at July 14th 2016 have an average length of 3:39 minutes, the shortest song is 2:42 minutes and the longest 5:13 minutes long.
In order to detect whether the participant is listening to music, the volume should go down for 2-5 seconds between a track with a duration between 2:30 minutes and 5:30 minutes. A regular pattern with these attributes should indicate that the user is listening to background music while working on the coding task.

6.1.1 Questions

After the gathering process, the participants were asked to answer some questions:

- Are you a student?
- Did you work in a team?
- Did you listen to music?
- Did you feel tired?
- Did you enjoy the tasks?
- Did you give all you attention to the tasks?
- Were you distracted during the tasks?
- Did you feel stressed

[2] `http://www.apple.com/ie/itunes/charts`

- Do you think the tasks were easy?

All the questions can either be checked to indicate 'yes' or leave unchecked for 'no'. The answers can help to clarify the classification or to get new additional contexts. Some of the questions are created based on the knowledge from previous work of researchers and their results that can possibly influence cognitive performance.

In a long term, asking questions is not optimal. In the future the app is supposed to learn and slowly make the questions unnecessary. Currently there is a way to detect whether the user is listening to music by identifying patterns but the accuracy is not exactly known and therefore also asked as a question as well could it be that the user is wearing headphones. If the detection using the environmental noise is highly accurate, the question can be removed from the app.

Chapter 7

Evaluation

This chapter shows the results of the experiments and how the gathered data can be interpreted.

7.0.1 Output Format

Figure 7.1 shows the output format of a single entry after the decryption.

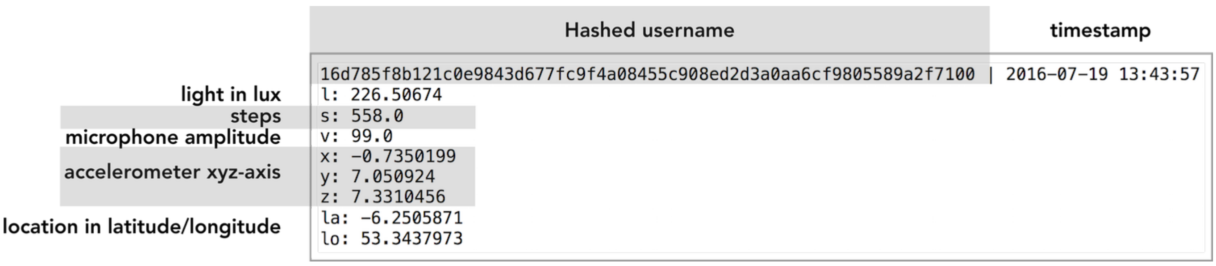


Figure 7.1: Gathered Data

7.1 Group Experiment

This section shows the finding in the experiment where the participants had to solve the programming task. This section is divided into the different categories where the findings were clustered into.

7.1.1 Noise

Regular mobile phone microphones are not calibrated to provide an general loudness. Therefore we can use the values of the environmental noise only for finding peeks or changes. The environmental noise shows a steady low noise level for four of the participants while one participant has some high peeks in the data.

7.1.2 Dynamic in Light and Noise

Noise	**P1**	**P2**	**P3**	**P4**	**P5**
Dynamic Score	139.72	78.51	37.61	67.80	59.63
Average	1,676.59	549.59	159.82	1033.95	498.67

Table 7.1: Dynamic Noise Level

Light	**P1**	**P2**	**P3**	**P4**	**P5**
Dynamic Score	2.46	2.44	-	12.73	0.26
Average	7.37	5.69	-	59.73	1.35

Table 7.2: Dynamic Light Level

The standard deviation added and subtracted from the mean is the range that separates normal noise and peeks. The values are only in a positive range. If the minimum value went negative in the calculation, it was set to zero instead.

The values outside of the range, as a percentage relatively to the standard deviation were added together and result in a number which can give a good indicator of the amount and level of dynamic. These value divided by the number of minutes of the gathering create the dynamic score which indicates the level of dynamic in a dataset. The average value is the mean of all the percentage values outside of the range.

7.1.3 Location

The location of the participants was not able to be accessed from two of the participants. Two of the other participants were working in Dublin, Ireland while a third one did the task in Madrid, Spain.

The step counter of the participants didn't provide valuable information as the step count was never more than 10 and didn't change during the gathering process. Also the location of the participants didn't change during the experiment. Thus, we can assume that the participants didn't move during the experiment.

7.1.4 Movement

The 3-axis accelerometer of the participants devices don't show any changes. In all the gathered results, the device was laying on the desk without big movements in between. A value close to 0 for the X and Z axis of the measurements as well as a value close to 10 for the Z axis show that the mobile phones were positioned screen upwards on a horizontal surface. Any movements of the device would have changed at least one of the values of an axis.

	P1	P2	P3	P4	P5
Description	-	Cloudy and Drizzly	-	Sunny	Cloudy
Temperature	-	17°C	-	31°C	14°C
Humidity	-	82%	-	31%	62%
Pressure	-	1007 nPA	-	1018 nPA	1028 nPA

Table 7.3: Weather during experiment

7.1.5 Weather

The weather is looked up based on the location, data and time. Therefore, the weather of the two participants without location was not possible to find out. The weather of the experiments of two participants was cloudy for one, and cloudy with a little bit of rain for the other one. The third participants had a hot sunny day. The individual information can be seen in table 7.3.

7.1.6 Light and Environment

The results of each individual participant (see appendix) show a good probability that all the participants where working on the task indoor. Based on the classification mentioned in the previous section, the highest brightness of the participants environment with 554.33 lux, which is a in the section of a office space or a very dark day 6.2 6.1. However, the mentioned maximum value is a peek in the results of a single participant. Another participant whose light level was on a high level was working during a sunny time which removes the possibility that it was a very dark day and that the participants was working outside. The other two participants had a lower light measured and therefore just a small possibility that they were actually working outside. Also the location of the participants are all in urban areas within a range of several buildings.

7.1.7 Music

Only two of the five participants answered the questions if they listening to music, with 'yes'. However, there didn't seem so be a correlation between any of the results and the fact of listening to music. Also the analysis of the gathered noise level showed didn't show any evidence of people listening to music. That either means that the participants were wearing headphones or the noise level didn't match the pattern that is described in the classification chapter.

7.1.8 Development Performance

The performance in software development is not only measured by the quality of the resulting code also the actual efficiency of the programmer matters especially for companies. The code from the experiment is analyzed based on these three aspects.

Light	**P1**	**P2**	**P3**	**P4**	**P5**
Src code lines per min	5.74	5.12	1.16	0.58	0.36
lines of src Code	69	95	20	35	33
max nesting depths	1	3	2	2	2
unnecessary lines	0	0	2	5	0
no intending	0	0	0	0	0
duplicated code	0%	0%	0%	0%	0%

Table 7.4: Development Performance

A software named "Teamscale" was used for the code analysis. The systems analysed the code for Structure Metrics like lines of code, nesting depth etc. It also detects comments, test coverage, architecture conformance, code duplications and Anomalies like naming convention violations [15].

The limited size of the code from the experiment also provides only limited code analysis information. Writing unit tests of the code was not part of the experiment and therefore a factor which can't take any influence in the results.

Efficiency

The efficiency of the code is been determined by the written source code lines per minutes. The more code the developer wrote per minute, the more productive he/she is. The lines of code in the project is an indicator of the complexity of the solution. Solving the same problem with less code is mostly a sign that the developer reduced the complexity and created better code. Also the nesting depth within the code is a factor in the efficiency of solving a problem. The deeper the nested loops are, the more complex it gets. In most of the cases it is better to keep the nested depth as low as possible. The longest method length was not used as an indicator because long method names can be more descriptive as well as be confusing. Therefore it can't be detected only from the metrics if it's positive or negative.

Code Quality

Having single lines of white-space is important to keep the structure of the code clean and separate parts such as methods from eachother. Anyhow, several lines are just reducing the code that a developer can see on the screen and should be avoided. In this section also the formatting is been looked at. Code without indented code blocks is massively decreasing the readability of the structure and makes it very hard to get an overview. It is also bad style to have duplicated code at different places in the project. That reduces the maintainability and can cause bugs or unexpected behavior. Here, the amount of duplicated code is shown in percentage. Luckily, none of the participants had any duplications in the code.

7.1.9 Variable and Method/Function Naming

The naming of the variables is detected by hand. A script can extract the variable names and method names of a project to make it easier for a large code base and multiple projects. For this experiment, we just looked at meaningful names rather than abbreviations which meaning can just be guessed. The evaluation of the naming of variables and methods should be descriptive and help others to understand the code. The code of the participants is not showing a very good example neither shows it a very bad one. Participant 4 was using the best naming for the variables while the method names are not clear. The best naming for methods can be found in the code of participant number 2. The least descriptive variable naming was done by participant 5 and the method name of participant of participant 4 where confusing in the functionality of the methods.

7.1.10 Coding Conventions

Coding conventions such as using camelCase for variables and method names, starting lowercase and Class names starting Uppercase, are some examples for coding conventions that should not be changed within a project to help to increase readability. The mentioned conventions above and keeping a constant style was followed by all participants except of participant number 5, who switched between camelCase and underscore for variable names.

7.1.11 Results

The experiment demonstrates that the application for collecting data works well and creates valuable data that can be used to to find evidence for influencing factors in development quality when it will be applied in a larger scope.
bigbreak The evaluation of the data showed that different results in the developing quality always had more than just one factors that are differentiating the work and environmental patterns 7.4. Also, the quality in software developing is assembled from different components and can't be used as a general term.

Participant four was the only participants who stated to be distracted during the experiment results show that there could be a possible correlation between the dynamic changes in the light level of the distraction as this participant has by far the highest dynamic score and highest average in the relative amplitude 7.2. The noise didn't show clear evidence as the results in the coding task don't show a pattern for the two highest values of participant one and participant four. Also no evidence was found that the solving time of the experiments has any influence in the length and quality of the code. Correlations between weather and any code metrics were not found within the results.

7.1.12 Problems

The gathering process showed some problems with the permissions of different mobile devices and different settings. Also, not every device has the same sensors and some devices have different default security settings that permit to access specific data within the application even when the application itself has permissions granted. Furthermore the usage of different programming languages for the experiments made it difficult to compare the results with each other. A larger group and a larger project with more resulting source code was not achievable in the scope of this dissertation but would have been of great value to detect influences rather than just plausible factors.

A first idea was to let the participants do the task whenever they want to find out whether they work at night or day. However, it turned out that the majority of the participants confirmed to do participate at the experiment but actually forgot to execute the experiment for several times. At the end, they did it right now after a being reminded. So, the time of the day can not be used as a indicator for detecting the working times of participants.

7.2 Individual Experiment Results

The results of the individual experiment are demonstrating the usage and the abilities for the 'Dather' application. Rather than in the first experiment, it is taking only the data of a single person into account. The data only shows a plausible factor but doesn't confirm the influences of the tested factors. A lot more test cases would be needed to create a more accurate average value. However, the purpose of this experiment was to demonstrate the usage of the gathering application to ensure more controlled comparable environments rather than in the other experiment which detected the environment and patterns.

The graphs of the three scenarios 7.2, 7.3 and 7.4 shows an overview of the durations that the participant needed to solve the Sudokus. The black dots are signaling the solving time on the y-axis for each of the ten measurements per scenario. The horizontal line in every scenario is signaling the mean duration that the participant needed to solve the Sudokus.

7.2.1 Coffee

The the scatter plot 7.2 shows the times of ten Sudoku solvings of the participant. The exact time can be seen in table E.1 in the appendix.

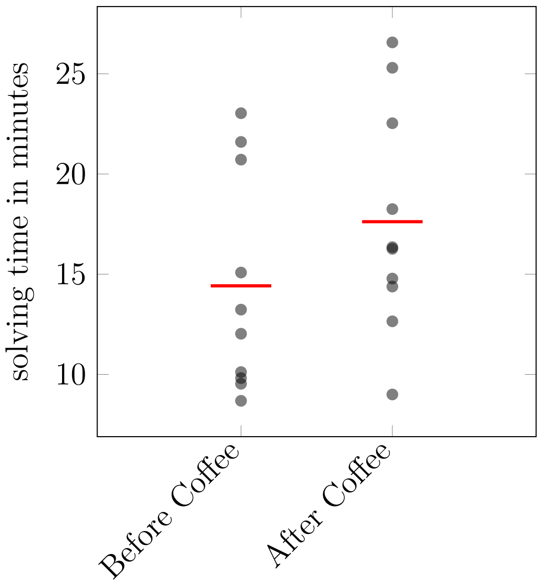


Figure 7.2: Scatter Plot: Coffee

The results shows an average time which the participant needed without drinking coffee was **14:24 minutes** and after consuming **17:37 minutes** minutes per Sudoku. The participant needed 3:13 minutes or 22.34% longer after having a strong coffee.
These results are different than the findings in [29], who found that a similar amount of caffeine (400 mg) increases the cognitive performance.
However, in our case the participant of the Experiment mentioned to feel fretful after the intake of the high amount of caffeine. That could be a reason for the lower performance of the subject. Thus, it is possible that a overdose had negative impact on the participant and lower amount of caffeine would have had resulted better.

7.2.2 Music

7.3 and E.2 show the times of the Sudokus that have been solved by the participant and the duration it took. It shows that the average time of solving was the shortest when the participant was listening to music. The participant solved the ten Sudokus in 89.35% of the average time compared to the results archived without music. That is **1 minute and 34 seconds less time in average**. On the other hand, the average solving time while

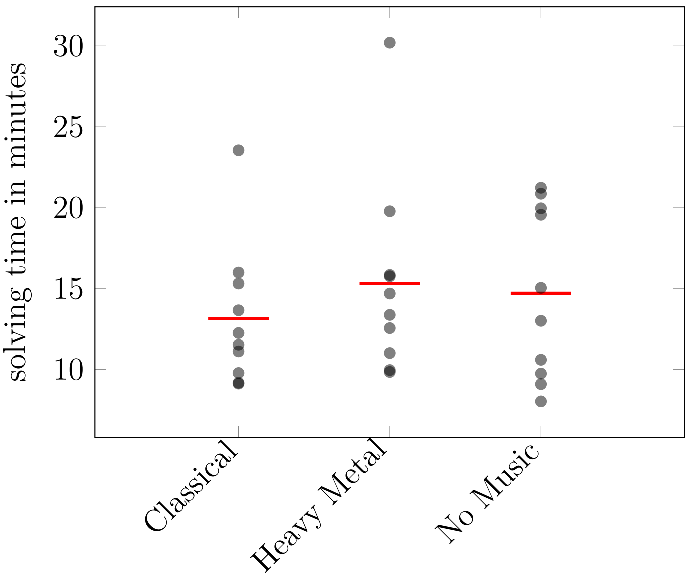


Figure 7.3: Scatter Plot: Music

listening to heavy metal music was 4.08 % or 36 seconds slower than without listening to music. The results show evidence that for the participant the cognitive performance in solving Sudoku riddles was increasing when listening to classical music and decreasing at heavy metal music.

7.2.3 Running

The graph 7.4 results show a trend for a better performance after strong physical activity. Table table with the exact times can also be found in the appendix E.3.The average solving time after the run (**10:52 min**) is 3:20 min faster than the **14:12** minutes before running. That is a decrease of **22.5%** from before to after and can be seen in E.3. Compared to the two other individual experiments, the results of this scenario are differing more at each measurement. 4 times, the solving after the running took actually longer than the completion in the pre-run state.

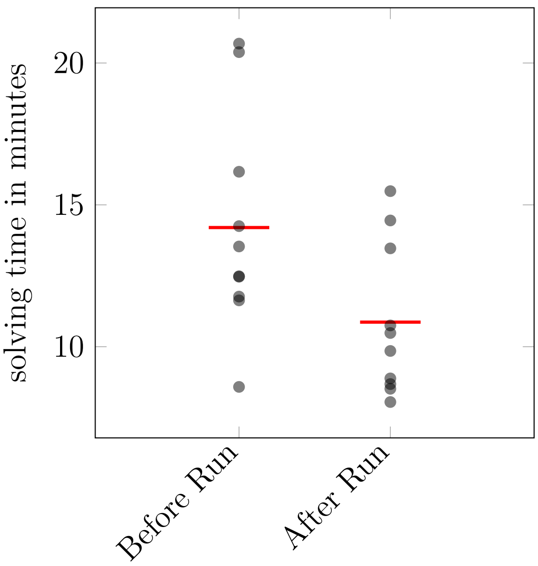


Figure 7.4: Scatter Plot: Running

Chapter 8

Conclusions

This chapter summarizes the findings of this dissertation and the evidence that have been accumulated. It concludes the work and results as well as the contribution to the research area. The chapter will be completed with a direction for future work in this area and the potential of research in this field.

8.1 Project Overview

This dissertation investigated correlations between temporarily environmental influences and human behavior in their cognitive performance. An Android application was used by participants to gather data about light, volume, location, accelerometer and the step-counter with a timestamp. Two different experiments, one with a single participant and another one with a group of subjects, were executed. The first experiment gathered data while the participants worked on a given programming task. The second experiment investigated isolated factors against each other. The results of the first experiment did not provide clear evidence for factor. It gives a plausibility that distractions of the participant can be correlated to a high dynamic light level in in the environment. The results from the second experiment gave more clarity and showed evidence of a negative influence by a high dose of caffeine as well as listening to heavy metal music while working. On the other hand, indicators point out that classical music while working has a positive effect on the cognitive performance as well as high physical activity before the tasks.

8.2 Contribution

The dissertation describes the development of a system for finding influences in the software development. The results show that the approach with an application for gathering data and afterwards comparing with analyzed code worked well and has the potential to find evidence in influencing factors for a larger scope. The source code of the Android app and all the tools for this work are available on Github for unrestricted use. The links to the repositories can be found in the appendix.

This dissertation portrays the first steps of measuring the environment and the behavior of the developer. It distinguished the data into context and compared it with other results in the area of cognitive performance and software development. A lot of previous work is about long term effects in cognitive performance while this dissertation investigated the instant influences who are actually easier to change by the developers themselves.
This work demonstrates the possibility to correlate code metrics with it's changing influences that could be used to optimize the processes and environments in software development.

8.3 Future Work

In order to find evidence for influencing factors in software development it is necessary to work with a larger group of participants who produce more source code.

Also, as already mentioned in the Design chapter, this dissertation provides a good base for creating an ecosystem for consonantly providing feedback to developers about their performance and environment. Such a system could also provide feedback to project managers and give suggestions how to improve performance, based on collected analyzed data.

Wearables such as smart-watches, fitness trackers or medical devices are entering the market and are providing even more information about the developers behavior, health and possibly more.

Furthermore, long term studies with more participants would help to create more accurate results and with that knowledge help the software industry to improve the quality and help future developers to be aware about their code quality.

More research work is also needed. As by now, many influences in cognitive performance are only discovered in experiments reasoned with theories but rarely scientific facts. In order to find more influences, a deeper knowledge is necessary to fully understand the human brain and cognition.

Appendix A

Abbreviations

Short Term	**Expanded Term**
AES	Advanced Encryption Standard
API	Application programming interface
APK	Android application package
CBC	Cipher Block Chaining
CSS	Cascading Style Sheets
GUI	Graphical User Interface
GPS	Global Positioning System
HTML	HyperText Markup Language

Short Term	Expanded Term
IQ	intelligence quotient
IOS	(originally) iPhone Operating System
JSON	JavaScript Object Notation
KLOC	Thousand lines of code
LOC	Lines of Code
PSP	Personal Software Process
PKCS	Public Key Cryptography Standards
REST	Representational State Transfer
RSA	Rivest-Shamir-Adleman
SAD	Seasonal Affective Disorder
SHA	Secure Hash Algorithm
SMS	Short Message Service
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience
XML	Extensible Markup Language

Appendix B

Source Code

- Android Application for gathering the data

  `https://github.com/MiChrFri/Dather`

- Java Application for decrypting the data

  `https://github.com/MiChrFri/Decryptor`

- Python toolset for formatting the results

  `https://github.com/MiChrFri/AnnaLize`

- Website and Backend

  `https://github.com/MiChrFri/frickmDE`

Appendix C

Programming task

The programming task for the crowd Experiment.

C.1 Palindromes

C.1.1 Question

Generate a palindrome with the maximum possible amount of characters from an input string.
Count the amount of characters from the input that you didn't use in your palindrome then add 65 to the result and convert the result to the respresending ASCII character

What's a palindrome?

A palindrome is a word which reads the same from left to right and right to left such as anna or racecar

Input structure

a String with characters from a-z and whitespace

The first line indicates the number of test cases

The following lines are the individual test cases

C.1.2 Example

input

2

hello my world

amazing code

Explanation Testcase 1

The largest palindrome you can create from the characters:

hello my world

lohol

These are 7 leftover characters:

emywrld \\char.count = 7

When we add 65 we get 72, which is a 'H' in the ASCII table

65 + 7 = 72 \\char:'H'

Explanation Testcase 2

The largest palindrome you can create from the characters:

amazing code

ama

These are 8 leftover characters:

zingcode \\char.count = 8

When we add 65 we get 73, which is a 'I' in the ASCII table

65 + 8 = 73 \\char: 'I'

Result

HI

Appendix D

Participant data

These are the results of the measurements from Experiment one for each participant.

D.0.1 Graph descriptions

Accelerometer

The accelerometer graph shows the changes of the three axis values over the time of the experiment.

Noise and Light

The graphs of the noise and light level show the changes of the measurements (y-axis) over time (x-axis). The the thicker horizontal line indicates the average value and two lines, one below and the other one on top of the average is the range of the standard deviation and is treated different in the evaluation section.

D.1 Participant 1

D.1.1 Date & Time

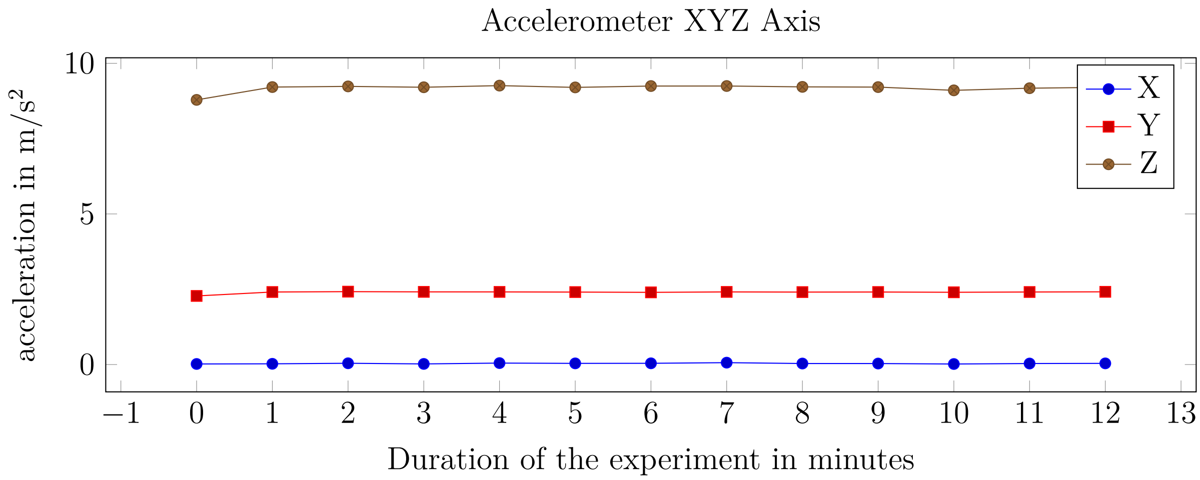
2016-07-28	
Start Time	End Time
12:52:44	**13:04:45**
Duration	
00:12:01	

Table D.1: P1: Date and Time

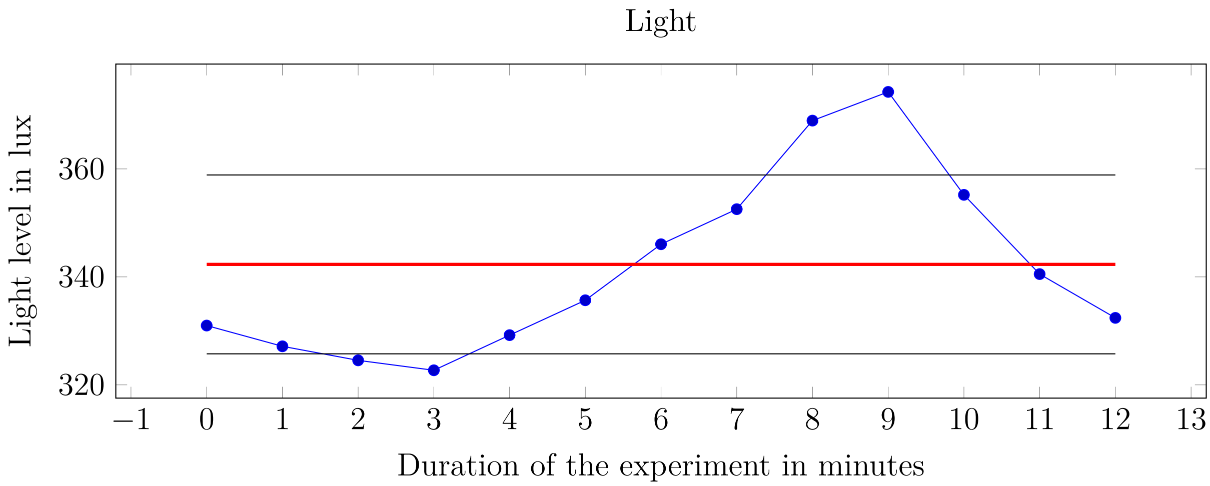
D.1.2 Questions

- ✓ Are you a Student?
- ✗ Did you work in a team?
- ✓ Did you listen to music?
- ✗ Did you feel tired?
- ✓ Did you enjoy the tasks?
- ✓ Did you give all you attention to the tasks?
- ✗ Were you distracted during the tasks?
- ✗ Did you feel stressed
- ✗ Do you think the tasks were easy?

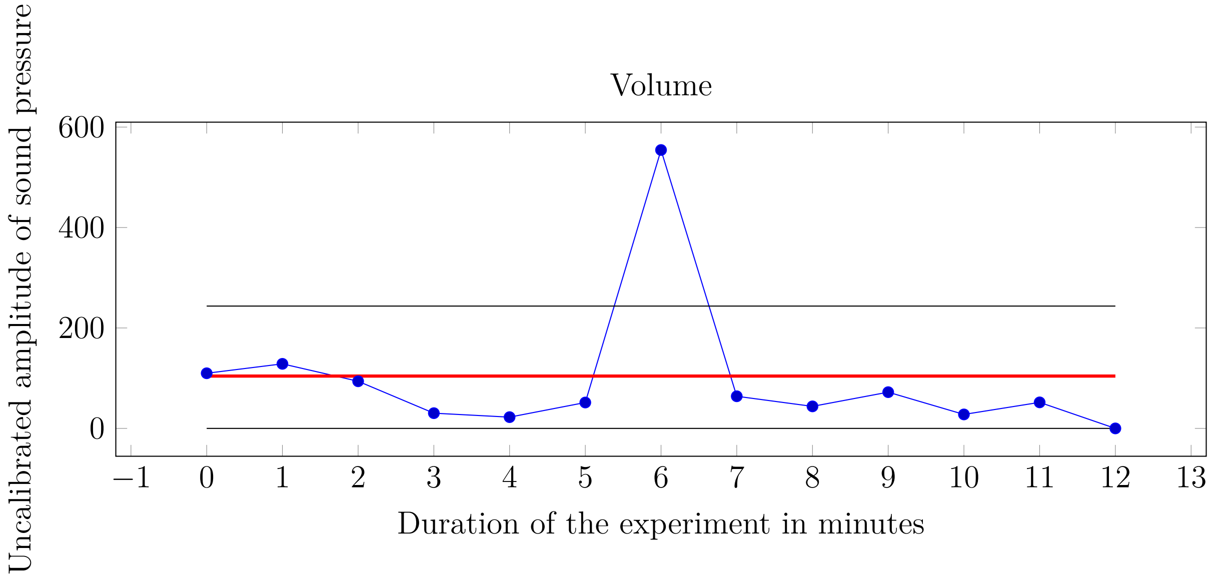
D.1.3 Accelerometer



D.1.4 Light Level



D.1.5 Noise Level



D.1.6 Location

No data gathered

D.2 Participant 2

D.2.1 Date & Time

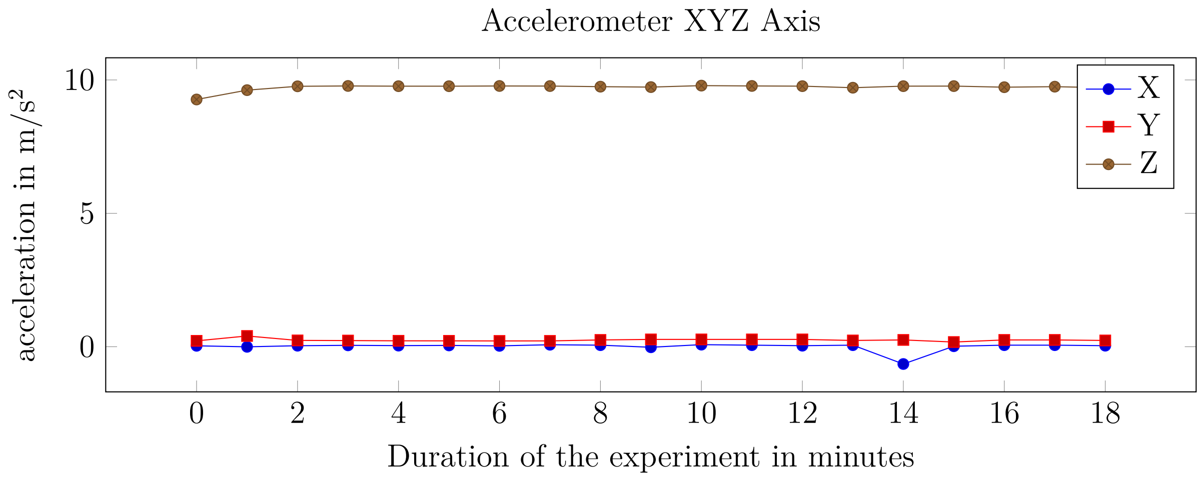
2016-08-03	
Start Time	End Time
12:23:50	**12:42:23**
Duration	
00:18:33	

Table D.2: P2: Date and Time

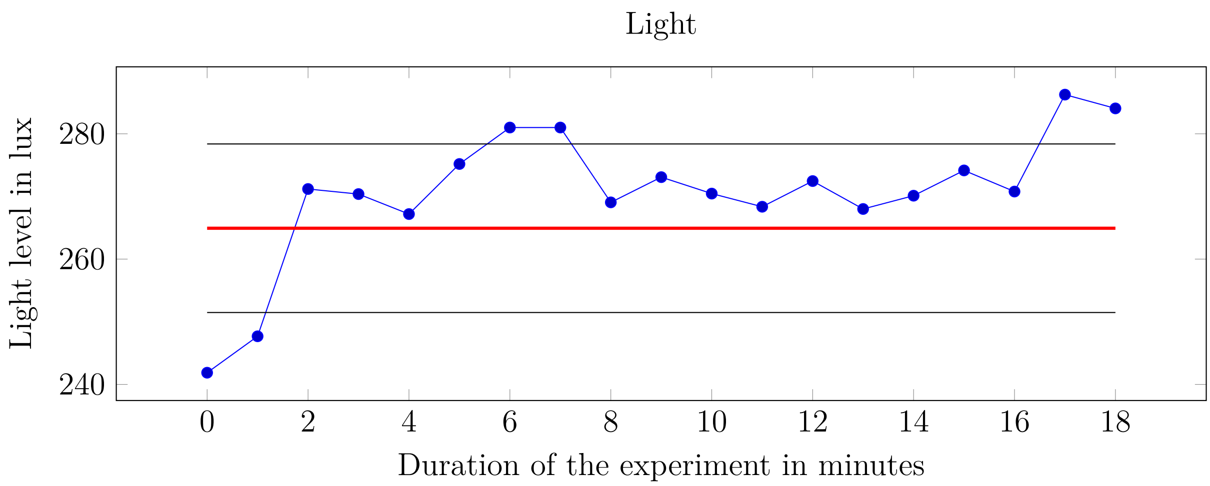
D.2.2 Questions

- ✗ Are you a Student?
- ✓ Did you work in a team?
- ✓ Did you listen to music?
- ✗ Did you feel tired?
- ✓ Did you enjoy the tasks?
- ✓ Did you give all you attention to the tasks?
- ✗ Were you distracted during the tasks?
- ✗ Did you feel stressed
- ✓ Do you think the tasks were easy?

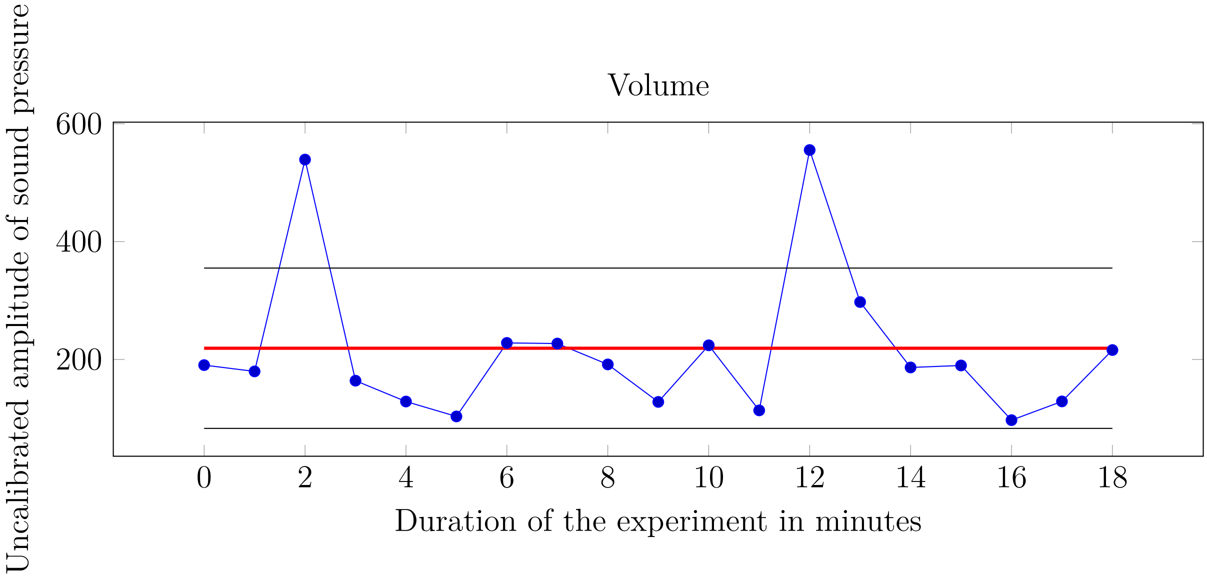
D.2.3 Accelerometer



D.2.4 Light Level



D.2.5 Noise Level



D.2.6 Location

minute 0 : -3.6881917, 40.4579957 minute 1 : -3.68815341053, 40.4579801474) from minute 2 : -3.6881432, 40.457976)

Madrid, Spain

D.3 Participant 3

D.3.1 Date & Time

2016-08-03	
Start Time	End Time
15:38:54	**15:56:12**
Duration	
00:17:18	

Table D.3: P3: Date and Time

D.3.2 Questions

✓ Are you a Student?

✗ Did you work in a team?

✗ Did you listen to music?

✓ Did you feel tired?

✗ Did you enjoy the tasks?

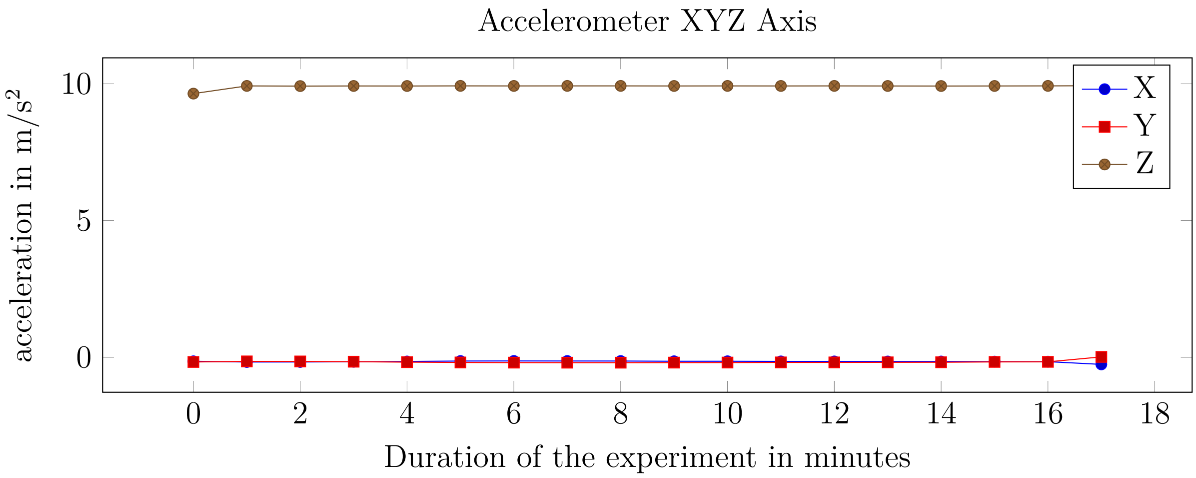
✗ Did you give all you attention to the tasks?

✗ Were you distracted during the tasks?

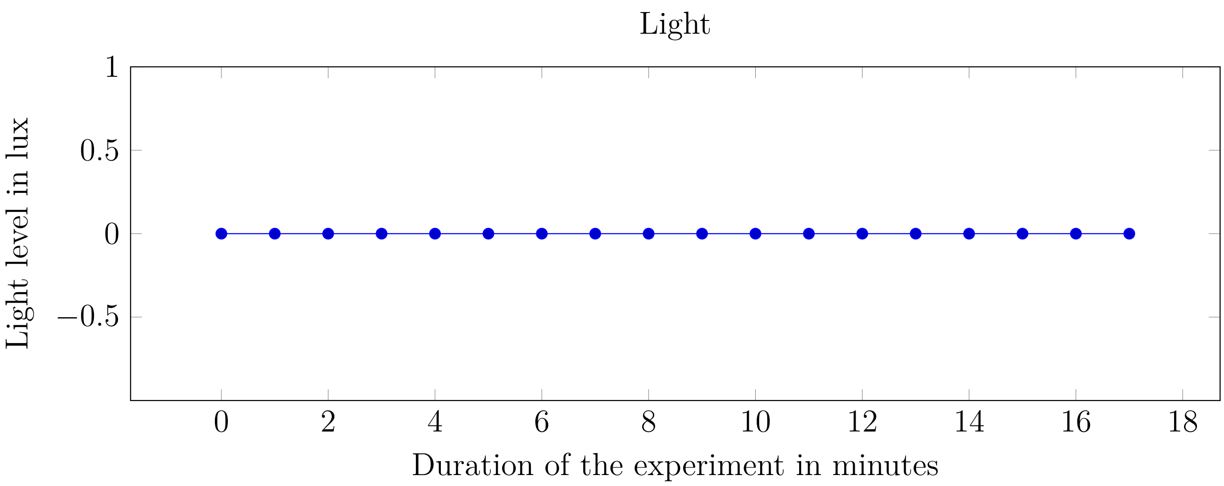
✓ Did you feel stressed

✗ Do you think the tasks were easy?

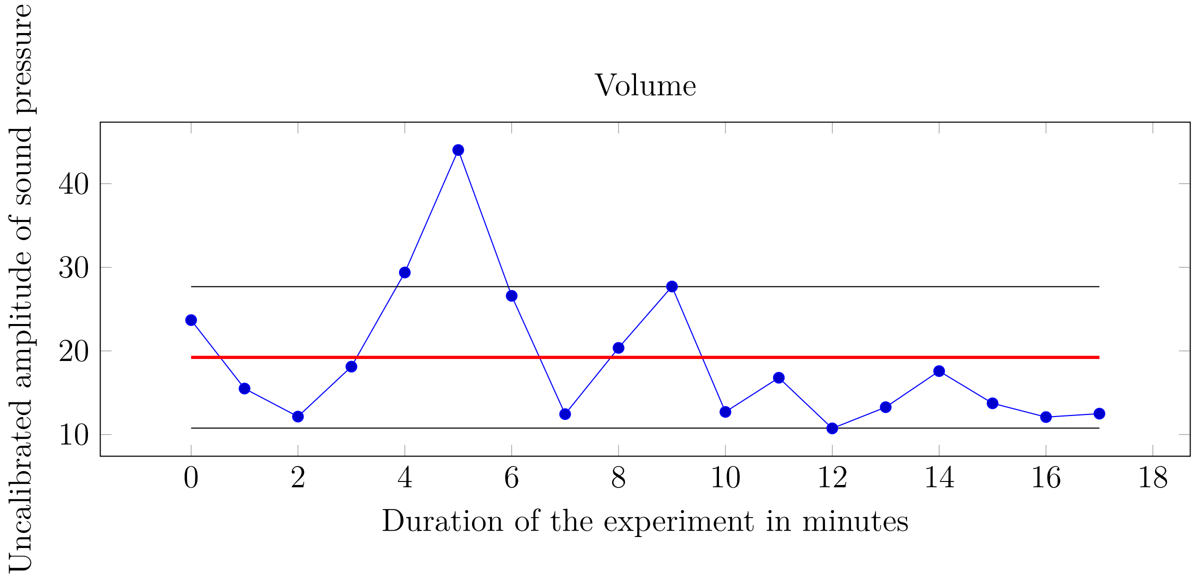
D.3.3 Accelerometer



D.3.4 Light Level



D.3.5 Noise Level



D.3.6 Location

No data gathered

D.4 Participant 4

D.4.1 Date & Time

2016-08-04	
Start Time	End Time
10:20:13	**11:20:41**
Duration	
01:00:28	

Table D.4: P4: Date and Time

D.4.2 Questions

✓ Are you a Student?

✗ Did you work in a team?

✗ Did you listen to music?

✓ Did you feel tired?

✓ Did you enjoy the tasks?

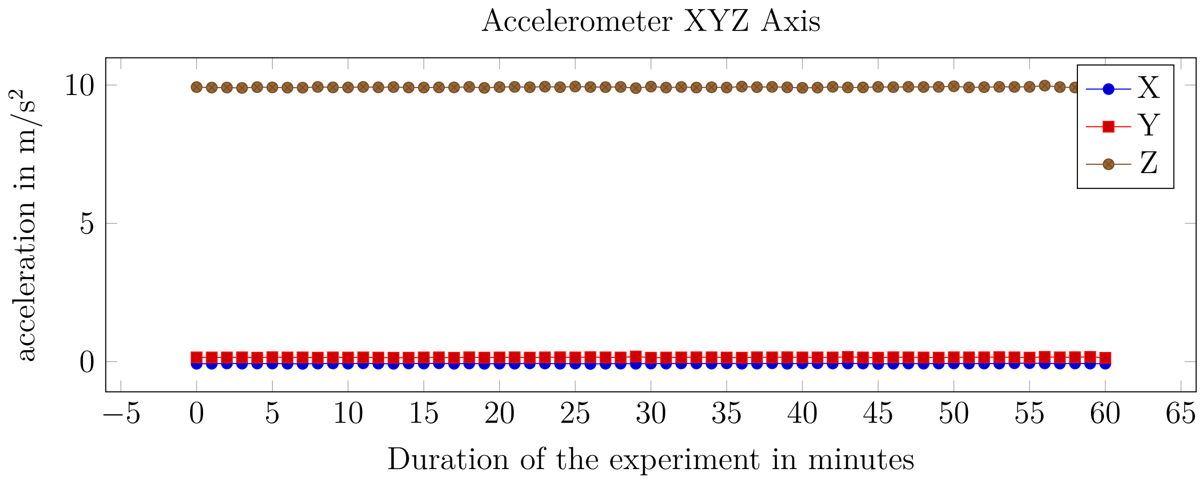
✗ Did you give all you attention to the tasks?

✓ Were you distracted during the tasks?

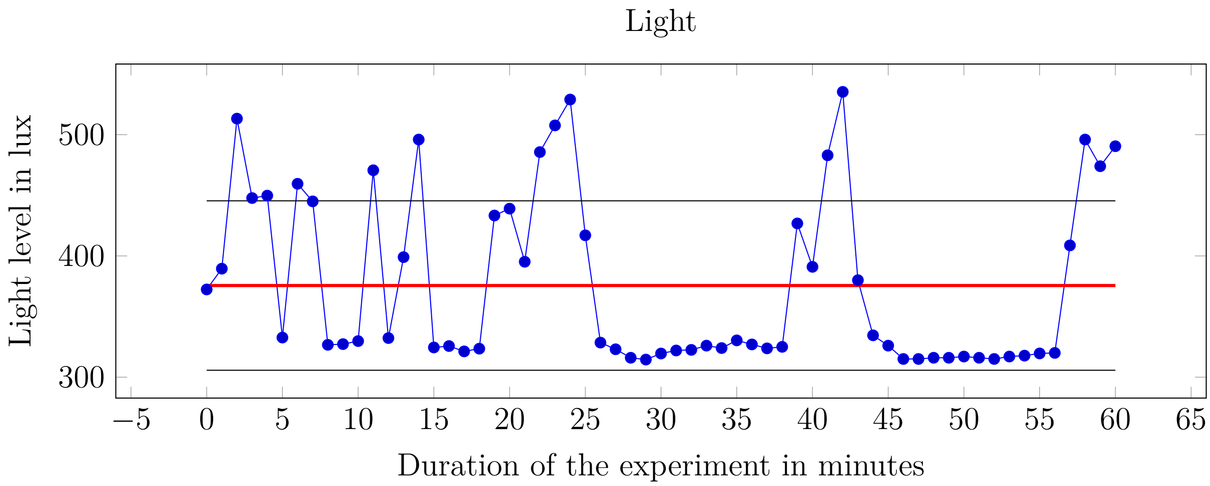
✓ Did you feel stressed

✗ Do you think the tasks were easy?

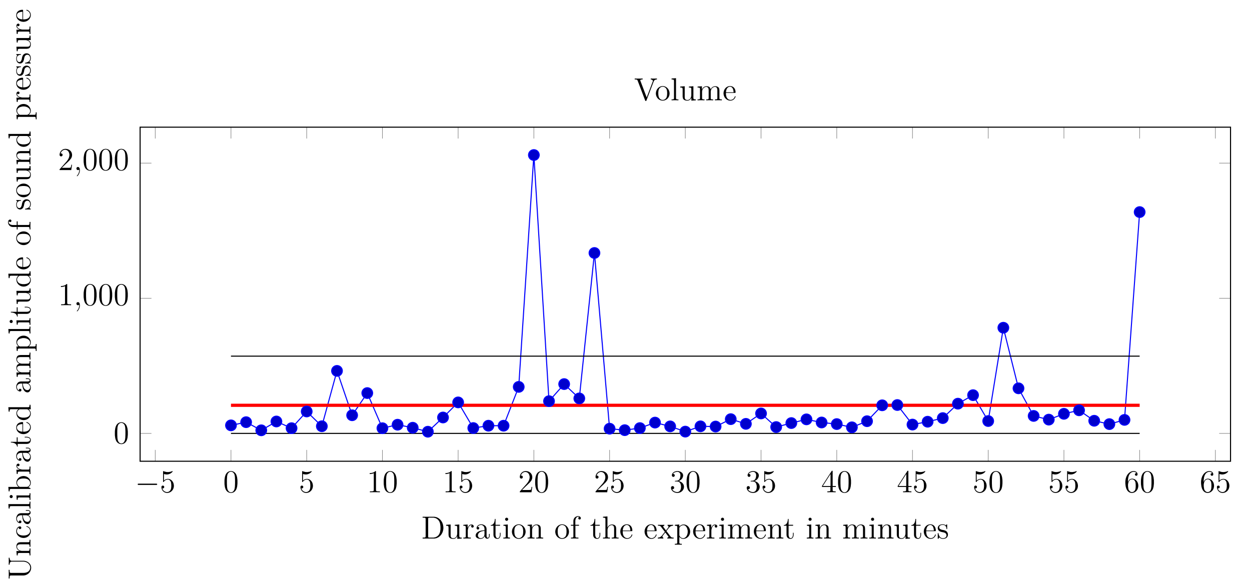
D.4.3 Accelerometer



D.4.4 Light Level



D.4.5 Noise Level



D.4.6 Location

53.3437734, -6.2510318

Dublin, Ireland

D.5 Participant 5

D.5.1 Date & Time

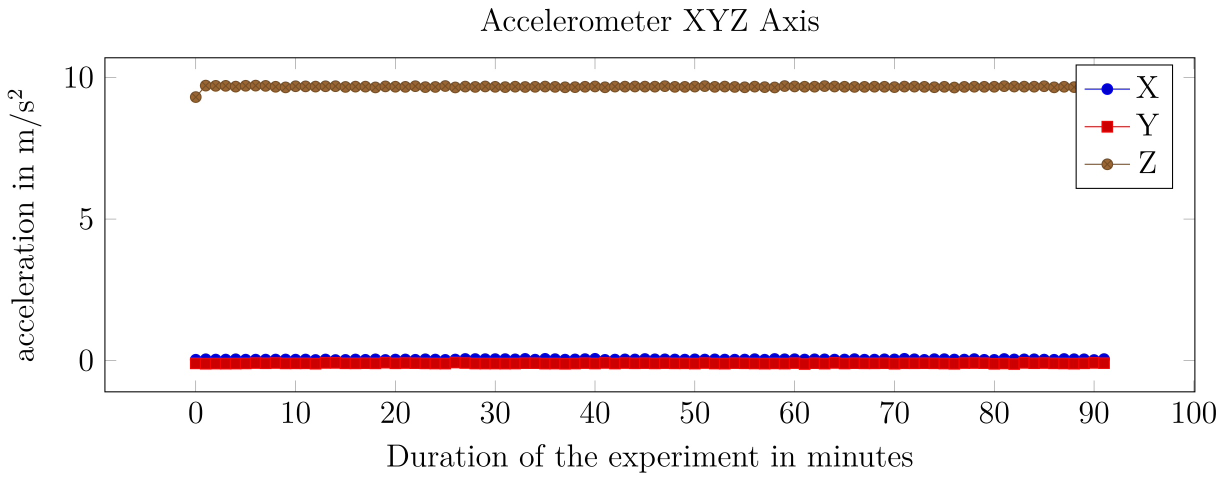
2016-08-09	
Start Time	End Time
15:49:28	**17:20:58**
Duration	
01:31:30	

Table D.5: P5: Date and Time

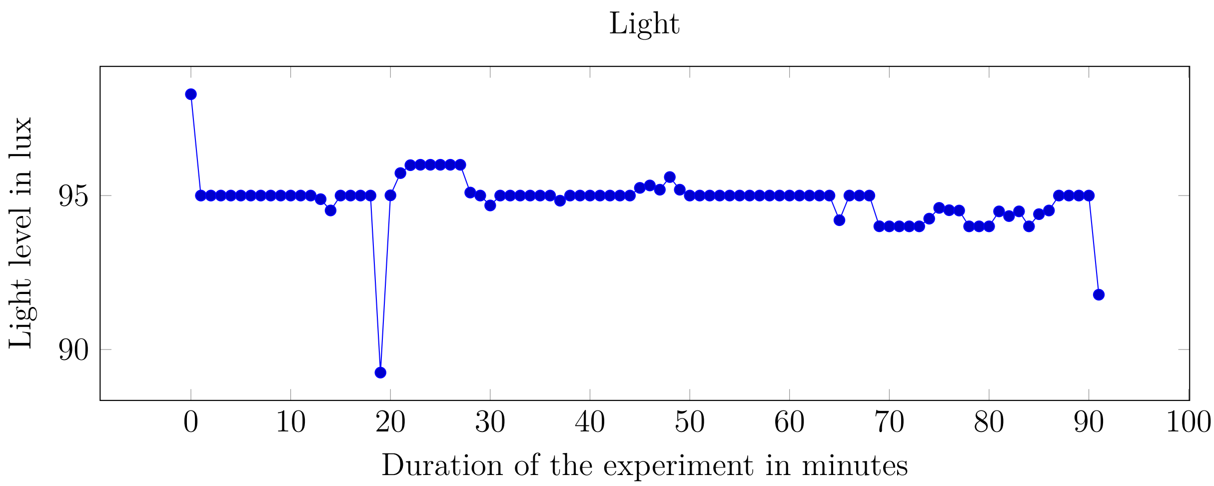
D.5.2 Questions

- ✓ Are you a Student?
- ✗ Did you work in a team?
- ✗ Did you listen to music?
- ✗ Did you feel tired?
- ✓ Did you enjoy the tasks?
- ✓ Did you give all you attention to the tasks?
- ✗ Were you distracted during the tasks?
- ✗ Did you feel stressed
- ✗ Do you think the tasks were easy?

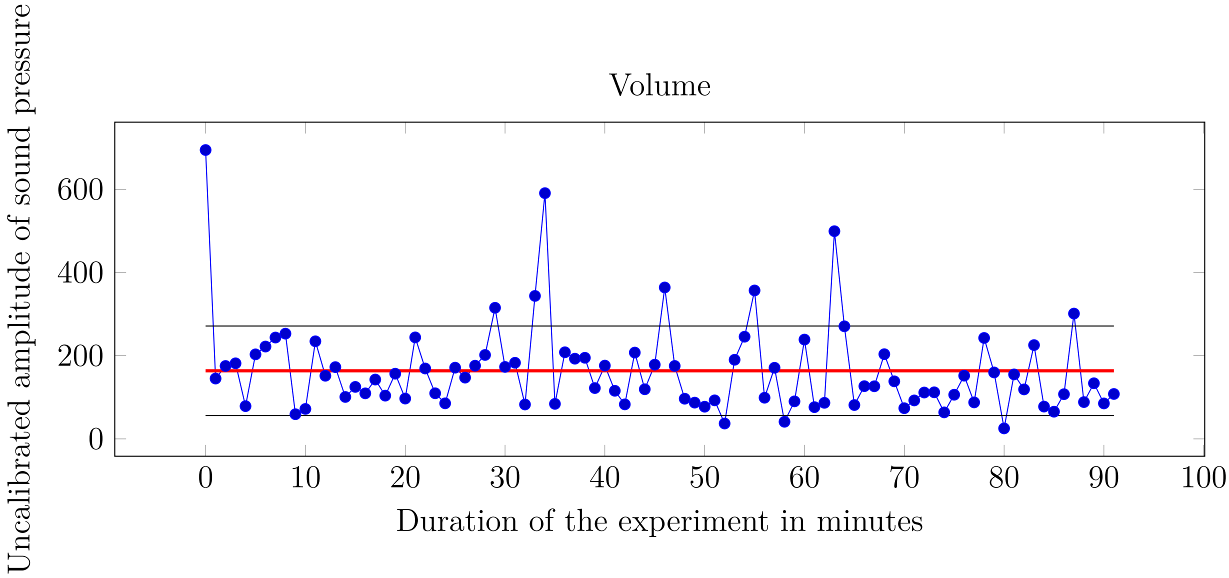
D.5.3 Accelerometer



D.5.4 Light Level



D.5.5 Noise Level



D.5.6 Location

-6.250537, 53.3437789

Dublin, Ireland

Appendix E

Individuals extended Data

These are the times of the measurements from the individual Experiment one for each scenario.

E.1 Coffee

Experiment with Caffeine										
Condition	Duration									
Measurement	1	2	3	4	5	6	7	8	9	10
No Coffee	20:43	21:36	13:14	23:02	09:32	12:18	15:05	09:49	10:07	08:41
Coffee	12:39	22:32	09:00	26:34	14:23	16:21	25:18	16:16	14:47	18:15

Table E.1: Cognitive Performance with Coffee

E.2 Music

Experiment with Music										
Condition	Duration									
Measurement	1	2	3	4	5	6	7	8	9	10
No Music	13:40	12:16	09:11	11:07	15:19	09:47	11:32	09:08	16:00	23:33
Heavy Metal	30:12	15:51	14:42	15:46	19:47	09:51	11:01	12:34	09:58	13:23
Classical	20:52	09:45	21:14	13:01	09:06	19:34	08:02	10:36	19:58	15:03

Table E.2: Cognitive Performance with Music

E.3 Running

Experiment with Running										
Condition	Duration									
Measurement	1	2	3	4	5	6	7	8	9	10
Before run	12:28	11:38	12:29	20:23	08:35	16:10	14:15	11:46	13:32	20:41
After run	14:27	10:29	15:29	08:03	08:53	09:51	10:45	13:28	08:41	08:31

Table E.3: Cognitive Performance with Running

Appendix F

Ethics Information

The following pages contain the 'Information Sheet for Prospective Participants' and the 'Informed Consens Form' from the ethics application.

TRINITY COLLEGE DUBLIN

INFORMATION SHEET FOR PROSPECTIVE PARTICIPANTS

This research is part of my dissertation which is a requirement of the MSc Mobile and Ubiquitous Computing within Trinity College Dublin.
Within this research we try to identify and analyse influences on the learning quality of writing computer software.

Participation:
In order to participate you must be aged 18 or over and legally competent to supply consent.
The provided information about this research and the consent form must have read by you or had read to you.
You must have had the opportunity to ask questions and all your questions must have been answered to your satisfaction. You also understand the description of the research that is been provided to you.
You agree that your data is used for scientific purposes and you have no objection that your data is published in scientific publications in a way that does not reveal your identity.
In the unlikely event that illicit activity is recorded, this will be communicated to appropriate authorities.
Your data is being gathered and securely transmitted to a server of the researchers/research team.
You may stop electronic recordings at any time, and you may at any time, even subsequent to your participation have such recordings destroyed (except in situations such as above).
No recordings will be replayed in any public forum or made available to any audience other than the current researchers/research team.
You freely and voluntarily agree to be part of this research study, though without prejudice to your legal and ethical rights.
You may refuse to answer any question and that you may withdraw at any time without penalty.
Your participation fully anonymous and no personal details about you will be published.
The proceeding is at your own risk if you or anyone in my family has a history of epilepsy.

Context of research:
Learning how to program and thinking like a programmer can be a very different process for different people. It seems that some students have more problems and struggles than others. Also more experienced programmers sometimes write better code and sometimes create less quality. This dissertation investigates metrics of environmental influences in the cognitive processes which are responsible for the code quality and performance in the process of software developing. The programming skills are measured by automated code analysis software and then correlated with gathered data during the development process. The built in sensors and of the participants mobile phone such as the location, microphone, accelerometers etc. are being used to gather information about the environment and context. Therefore it is required for the participants to install and start the provided android application on their device.

Benefits for the participants:
The participants will get feedback and information about their individual results and the code quality compared to other participants. They can also request the results of this study and the influencing factors in the quality of the results. This knowledge can help to increase their own code quality and optimise their working environment.

The procedure of this study:
Each participant needs to install the data gathering application on their cellular devices will explicitly asked to allow the application to collect the relevant data. The required permissions as defined by the Android application model are: accessing the location of the user, using the microphone, using the internet connection of the device and to get information about the network state. The granted permissions only effect this specific application. It doesn't influence any other application of the device or the operating system.

Afterwards the information sheet for prospective participants will be displayed within the application. In order to go on, the user need to confirm that he/she understood the information and agrees to the procedure. Starting the gathering process in the app will trigger a periodically collection of data until the user stops the process. Beside the gathering process, no data will be collected.
After working on programming projects, the participants may be asked to answer some additional questions through the application. After completing and submitting the gathered data, the participant can uninstall the application. The deinstallation is managed by the Android operating system and removes the application and all the gathered data from the device as well as all the granted permissions. Once the data has been actively submitted by the user, it will remain on the server even when the app is deinstalled on the mobile device. In order to delete the data on the server, the user must request the deletion via email.
The user is required to upload the programming code on Github and grant access to the working repository in order to do the analysis.

Withdraw from study:
The user can withdraw from the study any time without naming any reasons. In order to delete all the data that have been collected until that point the user needs to delete the Android app on his/her mobile device and inform the researchers via email in order to delete the data from the server.

The expected duration of the participants involvement:
First, the participant needs to download the app and initially enter information about person, university and experience. Before starting the programming task, the student needs to open the application and log the start time. After the submission the app asks the participant additional occasional questions. The estimated time for the initial setup is not more than 5 minutes. Afterwards the data is constantly being gathered during the development process. The submission of the particular datasets after the gathering takes less than a minute.

The gathering process:
The gathering process starts as soon the user actively starts the gathering within the Android application. The data is been gathered until the user hit the stop button. The gathered data will be saved encrypted on the hard disc of the mobile phone during the gathering process. After finishing the gathering, the user needs to explicitly tap on a button to send the data to the server. Without the active sending action user, no data will be transmitted at all.

Data to be gathered:
In order to find correlations between the code quality and environmental influences, we need to gather as many information about the environment as possible. We want to detect the ambient noise by using the microphone of the mobile phone. We will only save the noise value without audio files. None of the cameras will be used for collecting photos or videos, neither will we access any private data from the users mobile device.
We will also collect motion information about the mobile device itself, the data from its light sensor. In order to collect additional information about the environment, we gather the location and current time of the mobile device which we could use to get the local weather information. Some devices have additional sensors for detecting environmental information such as` temperature or humidity that we also might use in this study.

Usage of gathered data:
The data is exclusively used to for this study and will not be provided to anyone else then the research team. The data will be used to create behaviour models and identify interesting patterns.
(e.g. We expect to find correlations between exigent tasks and loud noisy in the working environment).
The collected data during the study will be anonymised and the transmission over the internet will exclusively be in an encrypted form and just used for statistical analysis purpose. The data will also be stored encrypted and a not traceable identifier will be used to keep the data anonymous.

Participants can be assured they will not be mentioned by name or any other unique identifier and their data will be handled with the most possible discretion.
We also guarantee no direct quotations and contextual appropriateness.

Rights of the participant:
Every participant has the right to withdraw and to omit the individual collected data, without penalty.
A requested withdraw would lead to deletion of the entire data of the participant which was collected related to that study.
Participants also have the provision to receive an individual debriefing after participation.
The participant has the right to request a digital copy of the dissertation after its completion.

TRINITY COLLEGE DUBLIN

INFORMED CONSENT FORM

LEAD RESEARCHERS:
Michael Frick, Stephen Barrett

BACKGROUND OF RESEARCH:
We try to identify metrics, how environmental influences can change the progress of improving programming skills of Computer Science students in order to provide feedback to the students. The feedback consists of correlation of environmental indicators with code performance, and anonymised comparative results with with the other participants based on categorised patterns during the experiment.

PROCEDURES OF THIS STUDY:
Each participant needs to install the data gathering application on their cellular devices will explicitly asked to allow the application to collect the relevant data.
Afterwards the information sheet for prospective participants will be displayed within the application. In order to go on, the user need to confirm that he/she understood the information and agrees to the procedure. Starting the gathering process in the app will trigger a periodically collection of data until the user stops the process. Beside the gathering process, no data will be collected.
After working on programming projects, the participants may be asked to answer some additional questions through the application. After completing and submitting the gathered data, the participant can uninstall the application. The deinstallation is managed by the Android operating system and removes the application and all the gathered data from the device as well as all the granted permissions. Once the data has been actively submitted by the user, it will remain on the server even when the app is deinstalled on the mobile device. In order to delete the data on the server, the user must request the deletion via email. The user is required to upload the programming code on Github and grant access to the working repository in order to do the analysis.

PUBLICATION:
This study is part of my Master dissertation and will be published by Trinity College Dublin.

DECLARATION:

- I am 18 years or older and am competent to provide consent.
- I have read, or had read to me, a document providing information about this research and this consent form.
- I have had the opportunity to ask questions and all my questions have been answered to my satisfaction and understand the description of the research that is being provided to me.
- I agree that my data is used for scientific purposes and I have no objection that my data is published in scientific publications in a way that does not reveal my identity.
- I understand that if I make illicit activities known, these will be reported to appropriate authorities.
- I understand that my data is gathered and securely being transmitted transmitted to a server of the researchers/research team.
- I understand that I may stop electronic recordings at any time, and that I may at any time, even subsequent to my participation have such recordings destroyed (except in situations such as above).
- I understand that, subject to the constraints above, no recordings will be replayed in any public forum or made available to any audience other than the current researchers/research team.

- I freely and voluntarily agree to be part of this research study, though without prejudice to my legal and ethical rights.
- I understand that I may refuse to answer any question and that I may withdraw at any time without penalty.
- I understand that my participation is fully anonymous and that no personal details about me will be published.
- I understand that if I or anyone in my family has a history of epilepsy then I am proceeding at my own risk.

I have received a copy of this agreement.

PARTICIPANT’S NAME:

PARTICIPANT’S SIGNATURE: Date:

Statement of investigator’s responsibility: I have explained the nature and purpose of this research study, the procedures to be undertaken and any risks that may be involved. I have offered to answer any questions and fully answered such questions. I believe that the participant understands my explanation and has freely given informed consent.

RESEARCHERS CONTACT DETAILS:
stephen.barrett@scss.tcd.ie
frickm@tcd.ie

INVESTIGATOR’S SIGNATURE: Date:

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] Jeff Bercovici. Slack is our company of the year. here's why everybody's talking about it. `http://www.inc.com/magazine/201512/jeff-bercovici/slack-company-of-the-year-2015.html`, 2015. accessed on 15.07.2016.

[4] Erran Carmel. *Global software teams: collaborating across borders and time zones*. Prentice Hall PTR, 1999.

[5] Tsun Chow and Dac-Buu Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software*, 81(6):961–971, 2008.

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

[7] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pages 1277–1286, Lyon, France, 2012. ACM.

[8] 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.

[9] 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.

[10] Denzil Ferreira, Vassilis Kostakos, and Anind K Dey. Aware: mobile context instrumentation framework. *Frontiers in ICT*, 2:6, 2015.

[11] 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.

[12] 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, Dallas, Texas, USA, 2011. ACM.

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

[14] W Hayes and JW Over. [a]the personal software process (psp): An empirical study of the impact of psp on individual engineers. Technical report, º Technical Report, CMU/SEI-97-TR-001, Software Eng. Inst., Pittsburgh, 1997.

[15] Lars Heinemann, Benjamin Hummel, and Daniela Steidl. Teamscale: Software quality control in real-time. In *Companion Proceedings of the 36th International Conference on Software Engineering*, pages 592–595, Hyderabad, India, 2014. ACM.

[16] Charles H Hillman, Kirk I Erickson, and Arthur F Kramer. Be smart, exercise your heart: exercise effects on brain and cognition. *Nature reviews neuroscience*, 9(1):58–65, 2008.

[17] 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.

[18] Watts S Humphrey. *A discipline for software engineering*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[19] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[20] Sirkka L Jarvenpaa and Dorothy E Leidner. Communication and trust in global virtual teams. *Journal of Computer-Mediated Communication*, 3(4):0–0, 1998.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] Panu Korpipää, Miika Koskinen, Johannes Peltola, Satu-Marja Mäkelä, and Tapio Seppänen. Bayesian approach to sensor-based context awareness. *Personal and Ubiquitous Computing, publisher: Springer-Verlag*, 7(2):113–124, 2003.

[28] 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, Athens, Greece, 2010. IEEE.

[29] Anthony Liguori, John R Hughes, and Jacob A Grass. Absorption and subjective effects of caffeine from coffee, cola and capsules. *Pharmacology Biochemistry and Behavior*, 58(3):721–726, 1997.

[30] 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.

[31] Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå. A teamwork model for understanding an agile team: A case study of a scrum project. *Information and Software Technology*, 52(5):480–491, 2010.

[32] 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.

[33] Mary Beth Pinto and Jeffrey K Pinto. Project team communication and cross-functional cooperation in new program development. *Journal of product innovation management*, 7(3):200–212, 1990.

[34] Devin G Pope and Ian Fillmore. The impact of time between cognitive tasks on performance: Evidence from advanced placement exams. *Economics of Education Review, publisher: Elsevier*, 48:30–40, 2015.

[35] Balasubramaniam Ramesh, Lan Cao, Kannan Mohan, and Peng Xu. Can distributed software development be agile? *Communications of the ACM*, 49(10):41–46, 2006.

[36] Linda Rising and Norman S Janoff. The scrum software development process for small teams. *IEEE software*, 17(4):26, 2000.

[37] Robert D Rogers and Stephen Monsell. Costs of a predicatible switch between simple cognitive tasks. *Journal of experimental psychology: General, publisher: American Psychological Association*, 124(2):207–231, 1995.

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

[39] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90, Palo Alto, California, USA, 1994. IEEE.

[40] Thomas Schöps, Torsten Sattler, Christian Häne, and Marc Pollefeys. 3d modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. In *3D Vision (3DV), 2015 International Conference on*, pages 291–299, Lyon, France, 2015. IEEE.

[41] Rory Sobolewski, Richard B Reilly, Simon Finnigan, Paul Dockree, Kate O'Sullivan, and Ian H Robertson. Monitoring of cognitive processes in older persons. In *2009*

4th International IEEE/EMBS Conference on Neural Engineering, pages 132–135, Antalya, Turkey, 2009. IEEE.

[42] 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.

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

[44] 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.

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

[46] Jie Xiang, Junjie Chen, Haiyan Zhou, Yulin Qin, Kuncheng Li, and Ning Zhong. Using svm to predict high-level cognition from fmri data: A case study of 4* 4 sudoku solving. In *International Conference on Brain Informatics*, pages 171–181, Beijing, China, 2009. Springer.

[47] 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.