A Wearable System for Acquiring Data on Subjective Experiences

Tomasz Rafał Kamiński

Kongens Lyngby 2016

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

Investigating human behaviour has long been an important research area in the fields of psychology, health care and social studies. Collecting data about human activities has become easier with methods such as Ecological Momentary Assessment and with development of modern technologies, but is still laden with the necessity of user interaction.

The goal of this thesis is to minimise the burden of human interaction required for manually recording data on their subjective experiences. The thesis describes the theoretical background behind the concept of self-tracking and discusses the motivation for investigating new input methods and developing the proposed solution. It explains the design and implementation process of the wearable system prototype and illustrates its usability on the basis of continuous evaluation and performance tests. Informal evaluation results are discussed and supported with explanation of designs evolution leading to the point when satisfactory qualitative feedback was achieved.

Performance test results show that the proposed wearable system succeeds in reducing the user interaction time by approximately 30% on average, compared with alternative mobile application. Additionally, the data collected during the continuous evaluation illustrates location and activity time patterns among participants.

In the end, the thesis presents possible directions of future development, discussing both high-level functionality concepts and minor usability improvements.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfilment of the requirements for acquiring a M.Sc. degree in Digital Media Engineering.

The thesis describes the design and implementation of an Android wearable application for manual collection of data on subjective experiences.

Lyngby, 04-June-2016

Tomasz Rafał Kamiński

Acknowledgements

This thesis was prepared as a continuation of the pre-project conducted in Fall 2015 during the *02830 Advanced Project in Digital Media Engineering* course at DTU. Both the pre-project proposal and the topic of this thesis were suggested by my supervisor Jakob Eg Larsen. I would like to express my gratitude to him for his guidance, feedback and engagement throughout working on this thesis.

Furthermore, I would like to thank Thomas Blomseth Christiansen for his contribution in continuous testing and data collection during the whole development process.

Special thanks are due also to my friends from the DTU Volley club, my colleagues at Exiqon A/S, and my friends and family in Poland, who supported me during my work and willingly contributed to the evaluation process.

Finally, I wish to thank Martine Kam, Denis Kirchhübel and Loïc Turquety for their patience with my English skills while proofreading this thesis.

Contents

Chapter 1

Introduction

In the recent years, the notion of *Quantified Self* has become widespread with the availability of and possibilities offered by mobile applications and wearable sensors. With multiple types of information being tracked, including movement patterns, sleep cycles, mood changes, body symptoms, etc., the act of self-tracking shows great promises in health applications. However, designers and developers have primarily focused on the passive sensing of activity. Mobile applications and wearable devices are currently capable of automatically monitoring multiple aspects of human behaviour, while the possibilities in active sensing (manual tracking of phenomena) have received far less attention.

Knowledge of human activities in their natural environment is crucial in many fields, especially health care, personal wellness and productivity, but poses certain challenges. Traditional methods based on retrospective self-reporting miss the temporarily dynamic user processes and are prone to memory biases[SS03] due to poor event details recall and the so called *parking lot compliance*[SS03, SSS+03], where subjects tend to fill the necessary data shortly before submitting results. Similarly, information collected during laboratory experiments is laden with ecological validity concerns, meaning that the behaviour of subjects might differ between the experiment setup and natural environment. Thus, it was concluded by J. Froehlich et al.[FCC+07] that it is crucial to collect the data as close to the tracked event as possible. Ecological Momentary Assess-

ment (EMA), described by J. M. Smyth and A. A. Stone[SS03], is one of the approaches developed to handle this issue.

EMA relies on time-scheduled reporting with short intervals or event-triggered reporting. Both approaches intend to eliminate the biases at a cost of increased user involvement, including long time and inconvenience of required interaction. Consequently, the compliance of this method has been relatively poor[SPD09]. Although the ways of capturing data for EMA have changed drastically with technological advancement, this issue remains to be solved. From using paper diaries, through PDAs, mobile phones and contemporary smartphones, the need of user interaction has been the most troubling factor. This thesis intends to explore a possible solution to it.

1.1 Thesis Goals

In this thesis, the design and development process of an Android mobile and wearable application for EMA is described. Its aim is to create a simple and user-friendly way of momentary data capture, which would result in higher compliance rate. To achieve that, the project intends to minimise the time and effort needed to collect the data. The thesis also discusses the ways of representing the collected data, its correlation to different data types, security issues and user attitude. The expected results and findings include quantitative measures of the user interaction time and qualitative feedback on the usability of the proposed solution. Furthermore, correlation between self-reported and automatically collected data is expected to be identified.

1.2 Contributions

Recent years brought a rapid development of wearable devices. Smartwatches, smart wristbands, smart glasses, Bluetooth buttons - all are aimed at changing the way the users interact with technology. Even though they all posses certain benefits and drawbacks, their influence on contemporary *human-computer* interaction cannot be denied[Bin03]. As this thesis is aimed at utilising the features of smartwatches, a set of design considerations must be mentioned. To begin with, the screen size and resolution bring certain limitations to the input recording. Not only are some data types difficult to handle (e.g. text input, long option lists), but also the app selection can be a troublesome process. Furthermore, the risk of collecting erroneous input due to accidental interaction

or, again, small screen size is quite high. The reduced functionality of smartwatches (e.g. lack of speakers) can also pose certain challenges, e.g. with respect to providing feedback.

The goal of this project is to develop an Android-based wearable system for collecting data on subjective experiences. The design of the wearable application will aim at reducing the data collection barriers as described by Li et al.[LDF10]. Design considerations based on other publications and own experiments will also be taken into account. The main focus will be put on reducing the necessary interaction time with the device. Due to the specific nature of the smartwatch as an input device, the simplification of the input types and input method will be introduced, together with a confirmation mechanism for some data types. The application will make use of the recently enabled custom watch face feature of *Android Wear* devices[Wel15]. In order to provide contextual data, the application will also utilise the built-in GPS sensor.

A smartphone companion application will be developed together with the wearable application, which will serve the role of the configuration manager. It will provide the users with the possibility to define their own input type and configure the display and notification settings. It will also offer visualisations of recorded data and offer the ability to correct past input. One crucial point of concern is to ensure the flawless synchronisation between the wearable and mobile devices - both when it comes to sending configuration to the wearable application, and receiving the data recorded by it. Additionally, the app designs should be intuitive and present all information in a simple and easy-to-comprehend way.

In order to illustrate the usefulness of the solution, three different tests were conducted throughout the whole process.

- *Quick-and-dirty* and *thinking-aloud*[VW00] methods were used to evaluate the design and UX of the application during the iterative development process.
- Performance tests were conducted to compare to different mobile-based approaches with respect to interaction time and error rate.
- Finally, a pilot experiment on three test subjects was used to gather continuous qualitative feedback, provide long-term data for analysis and estimate the compliance rate.

It is finally important that the applications offer undisturbed and smooth experience. Any flaws or delays would actually create new barriers for the user, instead of removing them.

1.3 Thesis Structure

This thesis is structured as follows. Chapter 2 presents the two main categories of related work and provides examples for both. Chapter 3 discusses the self-tracking issues in depth and how this project addresses them. Chapter 4 describes in detail the design process conducted throughout the thesis. Chapter 5 covers the major implementation issues faced during the development, including optimisation, location tracking, permissions and compatibility. Chapter 6 presents the evaluation methods used during this thesis project, together with a description of additional applications developed to support this process. Chapter 7 discusses the collected results, their meaning and influence on the design and development process. Chapter 8 describes potential future development to both improve the current version of the application and handle the data collected with it. Finally, Chapter 9 concludes the thesis in relation to its goals.

Chapter 2

Related Work

As mentioned in the *Introduction* chapter, collecting in-situ data regarding human actions can be extremely relevant in behavioural medicine, wellness and productivity improvement, or education (understanding study patterns), but it poses certain challenges. The Experience Sampling Method (ESM) introduced in the late 1970s was the first attempt to solve the issues related to laboratory experiments and surveys[SPD09] and a predecessor of later methods such as Ecological Momentary Assessment (EMA). Generally, the method is designed to make users document daily events by asking them about certain aspects of their life (such as mood, food consumption, physical activity, etc.)[CBBM+03, HSC07]. ESM evolved over time, from *paper and pencil* to *computerised methods*, which utilised mobile devices such as PDAs, mobile phones or contemporary smartphones. In the following sections, research works related to experience sampling and momentary assessment are presented, followed by the presentation of contemporary tools facilitating these methods.

2.1 Experience Sampling Method Research

Numerous Experience Sampling Method (ESM) studies have been conducted in the recent years, using both *paper and pencil* and *computerised methods*. Smyth and Stone rationalised the use of methods such as ESM and EMA by eliminating

the retrospective biases and increasing the ecological validity of collected data, e.g. by eliminating *the white-coat hypertension* effect[SS03], which corresponds to much higher blood pressure readings taken in a laboratory or medical setting, compared to a natural environment. Their study about coping with stressful daily events showed significantly different results between the EMA-based group of subjects and the traditional recall group, proving the importance of assessing the psychological processes in a natural environment. Similar studies, including the variability of asthma symptoms and momentary assessment of pain in chronic pain patients supported this observation.

The necessity of using technological innovations in ESM research was discussed by Michael R. Hufford et al.[SSS+03]. In the study, they compared the *paper and pencil* diary self-reporting to electronic diaries. According to the participant reports the compliance of both solutions was over 90%. However, the actual compliance of the paper diaries was close to 11% (which means that participants falsified 88% of their diary cards) while the actual compliance of the electronic diaries was only 1% lower than the reported one. The electronic diaries did not suffer from so-called retrofitting of data and the *parking lot compliance*, common for traditional *paper and pencil* methods. These findings, supported by a large number of publications[DGF+95, SPG+96, SHH+97], confirmed that technology-based electronic diaries are more reliable for collecting momentary data in natural environment. Using mobile or wearable apps for the same purpose is just one step further. One example of utilising the mobile technology was presented by Rajan Vaish et al.[VWC+14] and involved replacing the standard lock screen of the smartphone with simple environment-related question that encouraged short bursts of user contribution. The motivation for using those *micro-contributions* was to lower the threshold to participation in crowdsourcing. Among other findings, they concluded that reducing concerns about necessary input time and effort are key steps to volunteerism of users.

Smartphone-based ESM was incorporated by several recent studies regarding health-related issues. Yingling Fan et al.[FCLD12] conducted research on how travel behavior impacts health and whether interventions can increase travel-related health. Although no significant results were discovered, a similar approach was used by Jason D. Runyan et al. in their study on time-spending patterns of undergraduate students over a 14-week period[RSB+13]. A clear pattern of increasing study activity with semester progression was identified. Furthermore, participants using the self-reporting application admitted that it increased their awareness of daily activities.

A separate set of studies concentrated on finding correlations between the self-reported data and information automatically captured by built-in sensors of mobile phones or wearable devices. Jon Froehlich et al. developed a system

for *in situ* tracing and capturing called *MyExperience*[FCC+07]. Their system supported logging of over 140 event types and was tested in three field studies.

- The mobile phone charging study intended to recognize motivation for charging the battery and user preferences of doing it. Froehlich's findings showed that users tend to charge their phones when they still contain 68% of battery left, mostly because of data synchronisation need or simply a habit.
- The SMS usage study, utilised the SMS-triggered surveys and the GPS sensor. During the study, almost one third of SMS messages were sent in motion, usually because of answering a received message, convenience of inability to talk loudly.
- The last study aimed at finding correlation between ratings of places and the frequency of visits or travel time. Mentioned correlation has been found for selective places, e.g. bars.

AndWellness, a system developed by John Hicks et al.[HRK+10] for similar purpose integrated a number of system components to collect, return and analyse participant data. It was used in three case studies, all of which combined the self-reporting surveys and mobility data collection. Those studies measured the emotions and behaviour of cancer patients; the diet, stress and exercise of young mothers; and sleeping patterns of involved participants. An important conclusion coming from all three cases indicated that experience sampling systems can benefit from capturing contextual metrics. Other interesting findings included the necessity of reducing the burden of the user to absolute minimum and that the captured data can be momentarily-biased due to single point in time events. The same approach was used by Sean T. Doherty et al.[DLC14] for tracking human activity and well-being associated with visits to a protected park area. The GPS-based location tracking, combined with ESM surveys and a follow-up assessment survey showed significant potential for using smartphone-related technologies in capturing activity patterns. The psychological data assessment also confirmed the benefits of capturing data in subjects' natural environment.

2.2 Available ESM and EMA Tools

Currently, there are multiple smartphone-based applications that facilitate the ESM process from the survey creation, through scheduling, to providing end users with necessary notifications. They are meant to reduce the setup time

	OS	Mobile application	On-screen widgets	Wearable application	Survey creation	Survey scheduling	Server-side interface	SDK for sensor usage
movisensXS	Android	✓			✓	✓	✓	
ESM Capture	Android	✓			✓	✓	✓	
MetricWire	Android, iOS	✓			✓	✓	✓	
AWARE	Android, iOS	✓			✓	✓	✓	✓
funf	Android	✓					✓	✓
TapLog	Android	✓	✓		✓	Event-triggered		
Medopad	iOs	✓	-	✓			✓	

Table 2.1: Comparison of available ESM and EMA solutions for mobile platforms.

and costs of the research. Among numerous application available, a few specific are relevant to the present project.

One group of relevant tools include mobile apps for experience sampling alone. Applications such as **movisensXS**[1], **ESM Capture**[2] or **MetricWire**[3] are available for Android (all three) or iOS (*MetricWire*) devices and support server-side survey creation, scheduling and push notifications. In most cases, the surveys need to be created from predefined elements and can be triggered by fixed time intervals, randomly or by specific events.

Applications such as **AWARE**[4] and **funf**[5] go one step further. They provide their own API for integrating with built-in sensors such as accelerometers, GPS, Bluetooth, etc., providing contextual data to self-reported information. The main drawback of utilising these frameworks is the need of at least basic programming knowledge, which means that the simplicity of use from researchers' point of view is reduced.

[1] https://xs.movisens.com/
[2] http://esmcapture.com/
[3] https://metricwire.com/
[4] http://www.awareframework.com/
[5] http://funf.org/

TapLog[6], on the other hand, is an application for EMA, but in simplified form. It does not allow the creation of sophisticated surveys, but aims at reducing the interaction time when reporting events. Binary, numerical and text inputs are allowed, together with respective set of screen widgets offering the possibility of creating event report in a single click.

Finally, **Medopad**[7] is a modern British company that developed anapplication for EMA, that utilizes the possibilities offered by smartwatches, namely the *Apple Watch*. The application offers scheduling of simple surveys that are displayed and answered using solely the watch interface. It is currently used in a pilot study lead by *King's College Hospital in London* and receives very enthusiastic feedback, despite the noticed high cost of the hardware[Xan15].

[6]https://play.google.com/store/apps/details?id=com.waterbear.taglog&hl=pl
[7]http://www.medopad.com/

Chapter 3

Analysis

Quantified Self and *lifelogging* movements intend to incorporate technology into the process of capturing various aspects of daily life, including states, performance and subjective experiences. With the development of technology, this process has been greatly improved in the recent years. Mobile phones and wearable devices not only provide automatically captured data from various built-in sensors, but they have also made the self-tracking process much easier. The *always-on* and *always-worn* nature of smartphones, smartwatches, etc. provides the researchers and users with up-to-the-minute indication of behavioural habits[HRK+10]. Such information shows significant promise of utilisation in personal medicine.

This chapter introduces the concept of self-tracking with focus on Experience Sampling Method (ESM) and Ecological Momentary Assessment (EMA). It also presents the stage-based model of *Personal Informatics* and discusses the EMA in its context. Finally, it describes the high-level concept of a solution to cope with the identified EMA data collection barriers.

3.1 Self-tracking

Measuring the aspects of everyday life and understanding the dynamics of processes that influence them is a key interest of *lifelogging*. In the last 60 years, many works[EM76, Bra69, Z+80] have documented the studies on two methodological approaches to experience sampling - the time-budget research (investigating the time allocation of daily activities) and the psychological impact of everyday situations. However, there are two main issues related to these methods. First of all, they were based on retrospective self-reporting, which did not provide direct information about momentary states and actions of the participants, and were subject to memory biases. Secondly, they were often conducted in laboratory environments, the setup of which could not ideally reflect the real life situations. Experience Sampling Method (ESM) is an attempt to solve those issues and provide relevant data on the psychological dynamics of everyday activities. As described by Csikszentmihalyi and Larson[CL14], it aims at detecting variations in self-reported data by collecting data on daily activities, social interactions, psychological states, events, etc. To achieve that, it relies on the concept of time-scheduled or event-triggered self-reporting where users reflect on the activities and their context in real life situations, also meaning that the reflection itself happens close to the action it is describing.

As far as academic research is concerned, ESM has been widely used to provide data on how users change over time (e.g. Ilies and Judge's study on personality, mood and job satisfaction[IJ02]), what influences the change[STC+00], and finally, how aggregated information on individual experiences can provide a view into group patterns[MJK+12].

ESM, among other methods such as *self-monitoring* (which focuses on behaviours and ambulatory monitoring), is incorporated into Ecological Momentary Assessment (EMA) - a notion first defined by Stone and Shiffman in 1994[SS94] that intends to unify different research aims and foci under one methodology. That concept implies that EMA is more a set of guidelines and potential methods, than a research method itself. Shiffman et al.[SSH08] describe a couple of common factors that characterise EMA in general:

- **real-world data collection**, meaning that the users capture it during everyday activities and assuring the *ecological validity* of collected samples;
- **real-time data collection**, which stands for the *momentary* aspect of EMA, eliminating possible memory biases;
- **strategic scheduling of data collection**, which can be based on event occurrence (event-triggered recordings) or specific time pattern;

- and **repeated assessment**, providing multiple samples spread over time and a variety of situations.

Data collected using EMA techniques can be used for characterising individual differences, describing natural history, assessing contextual associations and documenting temporal sequences[SSH08], which provides great promises for the application in assessment and intervention in treatment. This field, however, is yet to be widely investigated, which this project intends to facilitate.

3.2 Personal Informatics

In general, the *self-tracking* as a process falls under the *Personal Informatics* domain. It represents the concept of reflecting on personal activities, making self-discoveries and behaviour changing based on detailed knowledge about the user[CLL+14]. Such knowledge provides potential benefits such as stimulating self-insights[HS93], supporting self-control[LRB03] or encouraging beneficial behaviours[SD77]. Personal Informatics systems provide tools for data collection, analysis and visualisation, and are currently aimed at data analysts, computer scientist, health and productivity enthusiasts or simply people interested in knowing themselves[CLL+14]. As described by Epstein et al.[ECB+14], their motivations for self-tracking can be very diverse and includes curiosity, social aspects (sharing and reminiscing), improving health or other aspects of life, and finding new experiences.

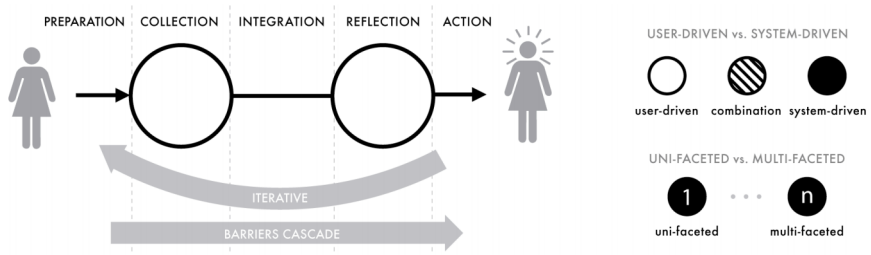


Figure 3.1: Stage-based model visualisation proposed by Li et al.[LDF10].

Interviewing 68 of such self-tracking enthusiasts allowed Li et al.[LDF10] define a stage-based model of *Personal Informatics* system in 2010. They describe five stages of collecting user data.

1. **Preparation** occurs before the actual data-collection. It refers to users' motivations, choosing the type of collected information and the way of doing it.

2. **Collection** stands for the process of capturing the data. Any observations regarding personal data can be reported, including, but not limited to thoughts, behaviour patterns, interactions, and environment. Various frequencies of collection are also possible, both time-scheduled and event-triggered.

3. **Integration** forms a bridge between the *collection* and *reflection* stages. It represents all the automated and manual actions that need to be taken to process the collected data. This can include simple aggregation, sorting, as well as correlating different kinds of input to provide a broader view.

4. The **Reflection** stage refers to the presentation of the collected data (processed or not) to the user, e.g. in a visualised form. It should allow the user to understand the feedback and make a decision based on it.

5. **Action**, as a final stage, stands for all activities performed by the users and based on the insights acquired throughout the process.

For each stage, Li et al. identified a specific set of barriers the users encountered during their study. Reducing those barriers is the key point to maximising the effect of the self-tracking process. Since the focus of this project is mainly limited to the *collection* stage, the following barriers have to be considered for the proposed solution.

- **Tool-related barriers** - participants reported that inability to access their personal computer, or even a mobile phone, at a certain time was the main problem. The observations of Li et al. indicate that the tool used for data collection should be as handy as possible.

- **User-related barriers** - many users lacked time to record the input at a particular time or simply forgot to do so. Hence, the designers should put effort in reducing the necessary interaction time. A reminder functionality built into the collection tool could also be relevant.

- **Data-related barriers** - collected data can often be subject to vague estimate (e.g. the weight or amount of food or drink) or subjective measure with no standardised data entry method. Although those issues are hard to solve in the *collection* stage of self-tracking, at least providing the users with guidance can improve the proposed solution.

3.3 EMA in Context of Personal Informatics

Some daily-life aspects do not pose any difficulties when it comes to tracking, especially considering the functionality currently offered by electronic devices such as sensors, smartphones or wearable devices. However, there are numerous data types that cannot be captured automatically. Ordinary events (sneezing, coughing, itching, etc.), consumption of food or drinks, properties of environment (such as attire), mood changes and all kinds of subjective experiences (pain, energy level, etc.) are impossible to record without manual interaction of the user. Not only are they all source of primary information about the user, providing insights into their daily habits, but also provide important contextual data helpful for extensive analysis. There are numerous scenarios to investigate correlations between habits and fitness/health/well-being, e.g. mood changes and the consumption of alcohol or tiredness. Unfortunately, manually collecting such data creates new challenges, including the necessary user interaction.

EMA is in general laden with specific barriers. First, and probably most important of them is the necessary time and interaction by the user. Capturing even a simple daily event requires users to invest their time and attention in the tracking process, which can complicate everyday routines (e.g. self-tracking at the workplace or during physical activity). It also enforces the need for the tracking apparatus being available and accessible at all times. Even the best *pen-and-paper-based* solutions or mobile applications will pose interaction challenges if the user is involved in some natural activities such as driving a car, cooking or gardening. As mentioned in the *Introduction* chapter, the recording should occur as close to the event as possible, hence those limitations raise validity and compliance concerns.

Both *pen-and-paper* and digital *self-tracking* solutions seem to present same difficulties. However, digital solutions offer much more possibilities with respect to providing contextual, automatically-captured data, such as geographical location. Furthermore, they save the interaction effort necessary to record information such as time of the event. Additionally, aforementioned *always-on* and *always-worn* nature of contemporary digital devices reduces the difficulty of having the apparatus available. It is arguable whether the need of keeping the electronic device charged at all time is more cumbersome than carrying the analog solution; however, since the digital devices usually serve more functions than just being a self-tracking tool, it is safe to assume that it is in users' best interest to ensure that the device is charged. Finally, when it comes to the simplicity of use of digital solution, a lot of attention has been given to the matter by Stone et al.[SSS+03]. They compared the user feedback gathered from two groups involved in an ESM study - one using paper diaries and the other using electronic diaries. No significant difference was observed when it comes

to comments regarding the ease of use and many participants actually favoured the digital solution.

Hicks et al.[HRK+10], in their EMA-based study, intended to solve some of these issues by providing the users with extended manual and automated customisation of the tool used for data collection. Some test groups could have individually set up the reminder patterns for requested surveys, others were only asked to fill in questionnaires when they first moved their phones in the morning. All of those features were introduced to maximally reduce the burden of the collection process on the user side. Hsieh et al.[HLD+08], on the other hand, proposed two possible ways of increasing compliance of the experience sampling research - providing feedback to the user and facilitating data collection. Although the first approach reaches already to the *Reflection* stage of *Personal Informatics* system, the second is strictly related to the *collection* process itself.

It is also important to realise that bursts of interaction with mobile devices, both expressed as a time and as a number of necessary actions, are becoming shorter with development of new technologies. Notification systems, gesture customisation, application shortcuts, etc. are all aimed at reducing the necessary *human-device* interaction time. Thus, it is crucial that the self-tracking applications adjust to this tendency. When it comes to event-triggered reporting, the number of daily occurrences of said event can reach tens of samples. With so many possible recordings, simple smartphone-based methods of reducing the *collection* barriers are not enough. The *Twitch Crowdsourcing* tool[VWC+14] is one example supporting this statement. Its solution, based on the collection of data via smartphone unlocking mechanism, provided increased user compliance, but still did not manage to eliminate all issues. Hence, using different mobile devices is a potential solution.

3.4 Conceptual Solution Model

Introducing digitalised solutions to EMA, or *lifelogging* in general, is a perfectly natural evolution of the research procedure. Lisa and Daniel Barrett[BB01] document a study where *pen-and-paper* collection method is replaced by mobile devices, specifically *palmtops*. They point out the device flexibility and control over the procedure as key advantages of such a solution, also mentioning the improved compliance, reduction of human error and contextual characteristics as additional benefits. Although programming, setup, maintenance and actual use of the devices are identified as potential issues, it has to be noted that the popularity and ease of use of contemporary mobile devices (smartphones) has

increased significantly since 2001. Further studies[VWC+14] do not mention such issues explicitly anymore. As mentioned in the *Available Tools* section, applications currently offered for mobile platforms provide numerous possibilities for self-tracking and succeed in reducing some barriers encountered by both users (in data collection) and researchers or supervisors (in configuring procedure). However, more can still be done with respect to that matter. The introduction of wearable devices presents completely new possibilities. Such devices, including bands, watches, glasses, etc. can not only provide contextual data for recorded samples, but also serve the role of the input devices itself. Their *always-wear* nature offers possibility to reduce the *tool-* and *user-related barriers* of data collection even more. This project is an attempt to implement and document that idea.

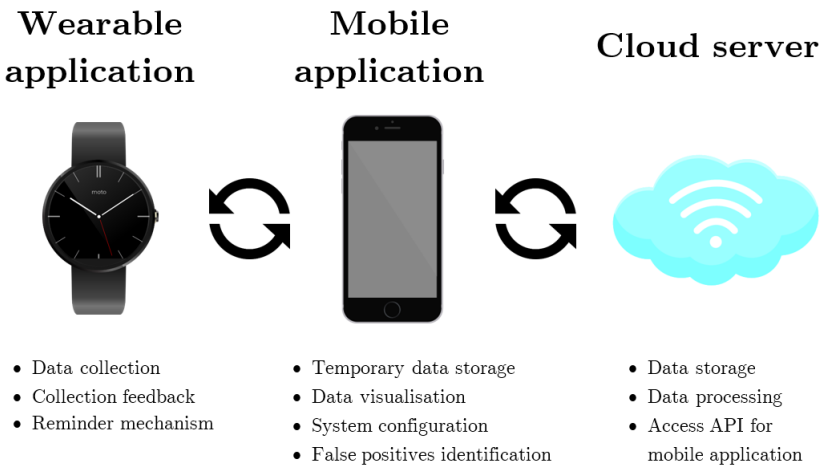


Figure 3.2: The conceptual model for the proposed solution, with specific module responsibilities. Particular system parts are synchronising the data between each other using wireless communication (Bluetooth, WiFi and/or mobile data.

The high-level concept of the proposed solution consist of three system parts, presented in Figure 3.2. The envisioned modularity of the system would allow to implement each of its parts separately and extend the functionality from there.

- **Cloud server** - responsible for the storage of recorded user data in a structured way and providing synchronisation with the mobile application. It also contains the API for accessing the raw or processed data in order

to display or integrate it with external applications. Data processing itself could include e.g. time-based aggregating or pattern recognition. Using a dedicated *cloud server* additionally provides platform independence when it comes to other parts of the system, as long as the format of the data is kept identical.

- **Mobile application** - used for temporary data storage of recorded sample before synchronising them with the *cloud server* and different forms of data visualisations, e.g. time series, spiral plots or geographical maps. Also allowing the configuration of the data collection system, including definition of tracked life aspects and customising the *look-and-feel* of the applications. Finally, providing the possibility to mark captured data as false positives, in case of recording errors.

- **Wearable application** - the actual data collection application, providing the user with a previously configured set of possible samples and the recording feedback. It could also be used as part of the reminder mechanism, requesting the user to collect the data according to specific time schedule or after long periods of inactivity.

Chapter 4

Design

This chapter describes the design process of the wearable and mobile applications introduced in the conceptual design in the *Analysis* chapter. It provides information about general challenges application developers face and describes the design evaluation process. The design overview and details are split into two sections corresponding to either of the applications. Within these, the overview of the application is provided, followed by the description of feature-based design iterations, design considerations, guidelines and qualitative feedback received during the evaluation.

4.1 Design Process

One of the most important phases of developing a mobile or wearable application is the design process. The limitations caused by the screen sizes and resolutions, the variety of the hardware devices, and finally the need to fit in with the predefined system *User Interface* (UI), all pose challenges for the designers and developers. The necessity of making the application highly interactive and intuitive adds to the complexity of the task.

In order to cope with all those constraints, an iterative design model has been applied to the design process of the solution proposed in this thesis. According

to the chosen approach, the whole system was divided into two separate parts - the wearable application for the smartwatch and the smartphone companion application. To make efficient use of the resources available for the project, the design process was split up in subsequent milestones. As the wearable application is main interface for the data collection, it was given a priority with respect to available time resources. Only after a satisfactory design had been achieved for the wearable application, the works on the mobile interface started. For each application, the designs were developed and implemented together with a specific set of features. Then, they were evaluated and revised, integrating an early stage of the next milestone's functionality.

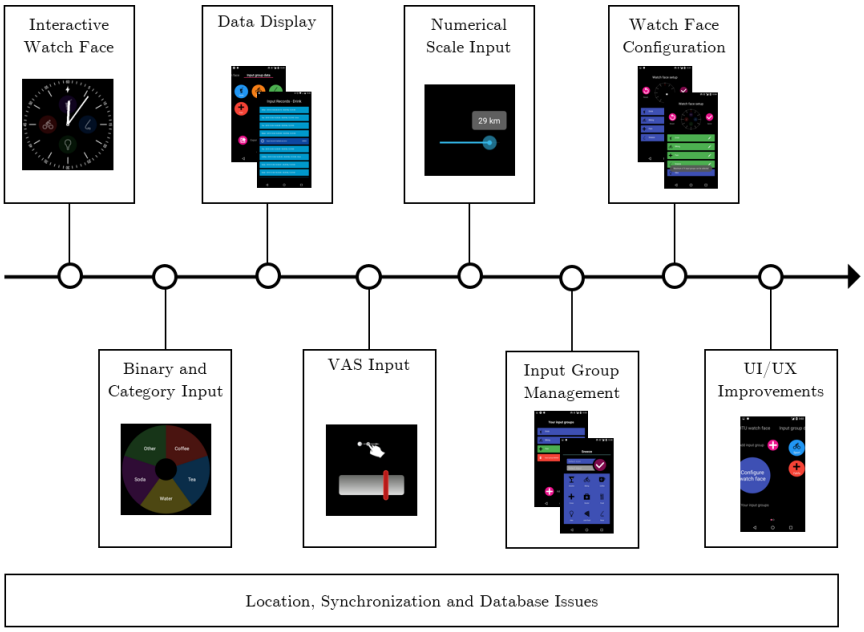


Figure 4.1: Timeline of the design process with respect to prioritisation of features.

Within every feature iteration, the designs were also subject to change and revision. Informal feedback was collected from members of author's social and professional network based on design mock-ups. Furthermore, the early (alpha) versions of the interface were deployed in the form of application prototypes to two test users - this thesis' supervisor, Jakob Eg Larsen (JEL), and Thomas Blomseth Christiansen (TBC). Throughout the period of between 13 and 21 weeks, they continuously used various versions of the applications, providing feedback on the UI. They configured the prototypes according to their needs and actually used it for daily self-tracking. This allowed them to provide con-

tinuous feedback on the UI, with ability to reflect on previous versions and suggest improvements. Apart from that, the *quick-and-dirty* and *thinking-aloud* methods were used on numerous test subjects (members of author's social and professional network) to provide additional *ad-hoc* feedback.

An additional challenge to be faced when designing and evaluating mobile applications is the necessity to provide an interactive prototype. An observation made during the evaluation process suggested that users are much more keen on providing useful feedback when they can actually work with the mock-ups, not only observe static graphics. That is why, even during the *quick-and-dirty* phase, the graphical mock-ups were wrapped in a mock-up interaction frame (*PoP*[1] application).

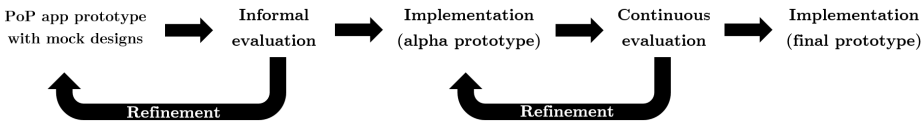


Figure 4.2: The design process applied to every feature of the prototype. Two stages of evaluation included informal *ad-hoc* testing with users from the author's social network and continuous evaluation throughout the long-term user feedback provided by JEL and TBC.

4.2 Wearable Application

Designing a wearable application is a challenging process, especially for unexperienced developers. Sarah Patrick in her article on *clutch.co*[Pat30] gathers a lot of insights regarding this process and names several possible considerations to be taken into account by developers:

- understanding the use case and maximising simplicity of operations performed on the watch;
- limiting the time of interaction with the device - both when it comes to time and number of touches;
- avoiding the overlap of functionality with the companion application;

[1]Prototyping on Paper - https://popapp.in/

- technical limitations of the devices, including weak hardware (processing power, built-in memory), lack of animations, limited gesture handling, and poor battery life;
- small size of devices and screens;
- and time-intensiveness due to extensive troubleshooting.

Many of those challenges were encountered during the development of this project. Additionally, it is important to acknowledge the fact that *Android design principles*[2] are not always in agreement with other guidelines such as the *I have a Big Thumb* rule[3]. Taking into consideration both the system UI standards and the specific needs of the application target group had to be addressed.

4.2.1 Overview

The original idea for the interface assumed utilising the recently introduced possibility of creating custom and interactive watch faces for Android-based smartwatches. The *Bits Watch Face*[4] application served the role of the proof of concept for that vision. The use of the watch face, instead of a standard *Android Wear* application, originated from the intention to reduce the user barrier as far as the interaction with the device is concerned. Providing the functionality directly on the main screen of the smartwatch eliminates the need to start the application through the menu. Precisely speaking, it eliminates two unnecessary touch gestures.

Created design mock-ups included different types of potential input methods, including binary, category, zooming-scale, Visual Analog Scale (VAS)[5], smartwatch keyboard (*Minuum*)[6] or even voice recognition. During early stage of evaluation some of them were rated as too complicated or confusing (zooming scale, smartwatch keyboard), which is in agreement with the *simplicity* design consideration mentioned by Patrick[Pat30]. Thus, the decision to start with the simplest types of input and then proceed to others was taken.

The final designs include an interactive watch face with input buttons. Depending on the input type (binary, category, VAS, numerical scale), the buttons

[2] https://developer.android.com/design/wear/principles.html
[3] https://www.airpair.com/android-wear/posts/android-wear-tutorial-comprehensive-introduction
[4] https://play.google.com/store/apps/details?id=com.ustwo.watchfaces.bits
[5] http://www.physio-pedia.com/Visual_Analogue_Scale
[6] http://www.minuum.com/

record the sample directly (binary) or launch a new input screen with additional options.

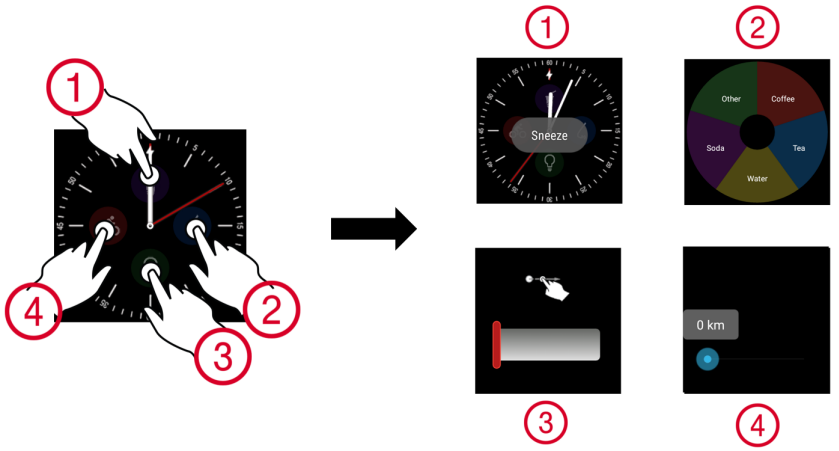


Figure 4.3: Smartwatch watch face interaction flow on example with four buttons for four different input types. The user can directly record a *binary* sample (1) or launch one of the input screens - *category* (2), *VAS* (3), *numerical scale* (4).

The input screens (Figure 4.5) provide the option to do the recording using different gestures (tap for category, swipe or tap for scale input types), exit the screen by swiping right and confirm the input value by tapping on respective confirmation icon (only for scale input types). This approach allowed further reduction of the interaction barrier. Depending on the type of input, any sample can be recorded by between one (for simple binary type) and three (scale types with input confirmation) tap gestures.

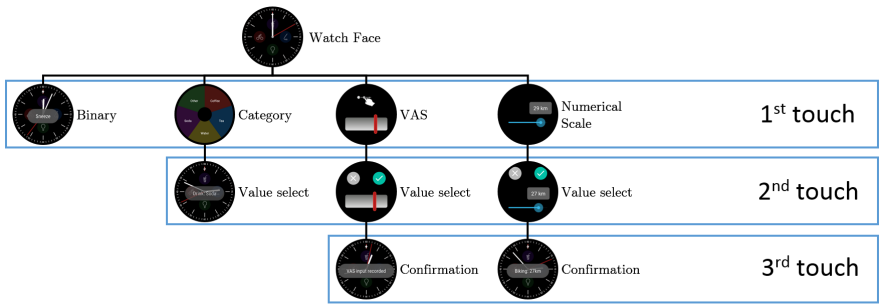
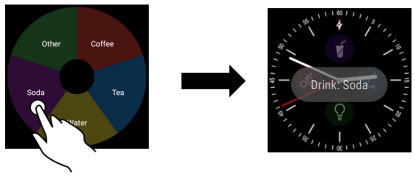
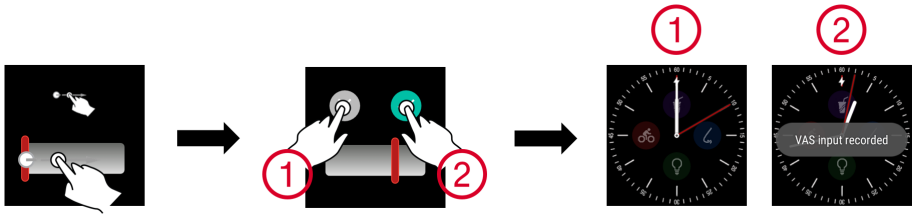


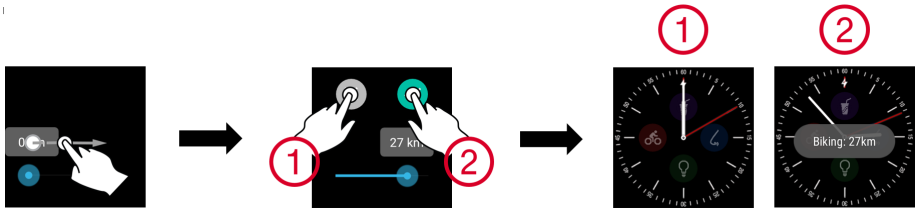
Figure 4.4: User interaction barrier with respect to number of gestures necessary to record a sample.



(a) Category input interaction flow. The user can select one of available values within the category. No input confirmation is required.



(b) VAS input interaction flow. The user can select a value of their choice and then confirm it (2) or reject the input and cancel the recording (1).



(c) Numerical scale input interaction flow. The user can select a value of their choice and then confirm it (2) or reject the input and cancel the recording.

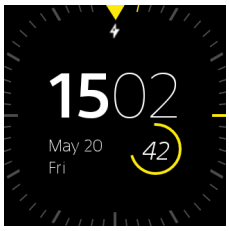
Figure 4.5: Smartwatch input screen interaction flows. For each input type (category, VAS, numerical scale), the user is present with interface allowing him to specify the value for capturing. After the sample is recorded, a confirmation feedback is displayed in the form of a *Toast* message containing the saved value.

The considerations, steps and additional details regarding the design of the respective wearable application features are provided in the following subsections.

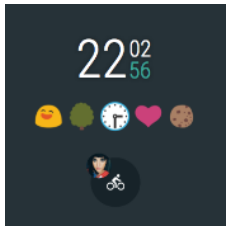
4.2.2 Interactive Watch Face

As far as the watch face itself is concerned, both the digital and analog clock designs were considered. The analog version was chosen as it allows easier and more structured incorporation of input icons. Digital watch face designs usually

place the hour display in the middle of the watch face in order to maintain the primary watch function. In case of smartwatches with square displays, it still leaves enough space for input icons, but in round displays the amount of free room becomes very scarce. Some watch faces, like *Together* (see Figure 4.6b) available for *Android Wear* actually limit the size of the time display and, as a consequence, incorporate the icons on the screen, but such approach raised serious issues regarding the robustness of the input and the potential number of accidental samples.



(a) One of the stock *Sony Smartwatch 3* watch faces. The amount of space free from time display does not allow comfortable positioning of input icons.



(b) *Together* watch face. Small time displaying and dense icon positioning raise usability issues.

(c) Classic *Bering* watch design[7]. Inspiration for the icon default positions.

Figure 4.6: Watch face design considerations, including currently available watch faces and standard analog watch designs.

The icons required consideration themselves. Their placement on the screen was inspired by the design of classical analog watches that incorporate stopwatches or different time zone features as complications. It also allowed to use the exact same design for both round and square devices and received only positive feedback. In the *Android Wear 2.0: Developer preview tour*[8], released May 18th, a similar design approach is used to implement official exemplary interactive watch face.

Furthermore, as far as their look-and-feel is concerned, according to the initial feedback, younger and more tech-savvy users favoured very minimalistic and elegant designs (Figure 4.7b), without any background colours, while older testers preferred the visible distinction of the enabled input field behind the icon (Figure 4.7a). Fortunately, such details can be solved by a customisation mechanism. However, the input field size itself caused a bigger problem. It had to offer a balance between being big enough to press easily and small enough to avoid the accidental or wrong recording.

[8]https://www.youtube.com/watch?v=8gLwk8o9LW0

Finally, following user requests, a recording feedback was introduced in the form of a simple *Toast* message combined with vibration pattern.

(a) Icon designs preferred by older users. The icons contain coloured background that corresponds to the enabled input field.

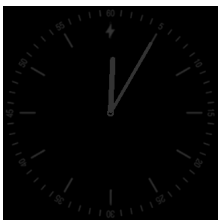
(b) Minimalistic icon design. The design provides more elegant experience, but lacks information about the input field.

(c) Feedback provided after registering the user input.

Figure 4.7: First iteration of watch face design - icon design considerations and input confirmation.

Further testing resulted in a noticeable number of false positives, captured while the smartwatch was in the ambient mode, similar to the sleep mode available on smartphones. The input icons in this state, despite being greyed-out, were still kept active. Users tended to register new samples while trying to simply wake up the device screen. In the final version of the prototype, the icons were removed from the ambient mode and additional validation of watch face state was introduced (see *Implementation* chapter for details).

(a) The ambient mode of the watch face with input icons active.



(b) The ambient mode of the watch face with input icons completely removed.

Figure 4.8: Second iteration of watch face design - ambient mode modification.

Additional hardware-based design consideration resulted from the feedback of a continuous test on a *Motorola Moto 360* smartwatch. This specific model is

[8] http://www.beringtime.com/index.php?id=21&L=2&ArticleId=32139-202

laden with bottom part of the screen being inactive, also known as the *chin* or *flat tyre*. In order to limit potential problems with this issue, the bottom icon position has been given least priority when it comes to placing on the screen. This approach ensures that only in case of using four input icons, the input region can be affected.

(a) Icon positions with high priority of bottom position.

(b) Bottom position set to lowest priority

Figure 4.9: Second iteration of watch face design - icon positions prioritisation.

4.2.3 Category Input

The *category* input was a second type of input considered for the application. Two possible designs were investigated - the select wheel and the carousel input. Test users, however, were not convinced by the carousel version due to big number of options that are not visible to them. They could not know how much scrolling is necessary to reach the desired choice and preferred the much more limited, but clear select wheel (Figure 4.10). With the design type chosen, additional considerations occurred. Because of the possibility of misclicked input and taking into account the *I have a Big Thumb* rule the inner part of the select wheel was blacked-out and made inactive. Furthermore, considering the *I have a big thumb* rule, the number of possible options was limited to only five, as with greater numbers the risk of recording erroneous input was too high.

4.2.4 Visual Analog Scale (VAS) Input

As the name suggests, VAS input is characterised by the user not being aware of the unit or jump of the scale they are using. It is used for assessing subjective values such as pain intensity or tiredness[BSG01]. The original designs of the VAS input featured multicoloured scale following the traffic light patter (from green to red), however, a few objections were raised by the test users. First of all, the VAS scale does not have to be used to illustrate a state where higher values

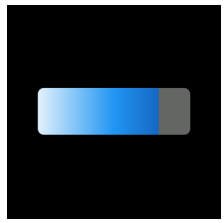
(a) First iteration - category select with maximum number of available options.

(b) Second iteration - the middle part of the select wheel has been disabled to limit input errors.

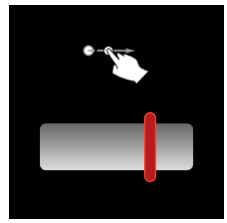
Figure 4.10: Category input design considerations. The select wheel presents all available input options to the user.

are more dangerous or intensive than lower ones, thus, the colouring scheme could be misleading. One of the proposed solutions suggested using different intensity of the same neutral colour (e.g. blue); however, it would not eliminate the intensity issue completely and could actually increase the confusion of the user. That is why, the idea of recreating the very traditional paper-based VAS input originated. With simple grey background and red indicator, the user is left without the suggestion about the input meaning. In order to make it even more clear for the users, additional icon with gesture description was placed above the scale itself.

(a) First iteration - VAS scale with colour gradient visualisation.



(b) Second iteration - VAS scale with single colour gradient visualisation.



(c) Third iteration - traditional VAS visualisation with icon manual.

Figure 4.11: VAS input design considerations

The final placement of the scale itself required some considerations regarding the usability on the round-screen devices, as the scale size could be significantly smaller. However, placing the scale slightly below the centre of the screen reduces its size by less than 10% (215dp vs 195dp) and provides more space in the top part of the screen for input manual icon or confirmation mechanism. To verify the influence of the VAS input size on round smartwatches, a study involving

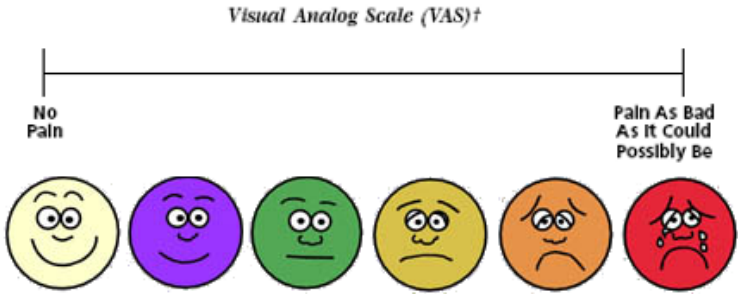
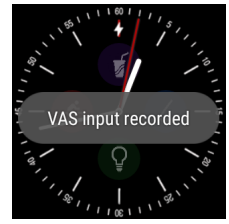


Figure 4.12: Traditional Visual Analogue Scale. Users mark their input on the scale with a vertical line[9].

two groups of users provided with both types of devices could be performed in the future.

Although the users did not raise any complaints or suggestions about it, it is important to mention that the confirmation feedback from early designs had to be altered in later versions. For testing purposes, displaying the actual value recorded by the VAS input gives a good overview of the mechanism. However, the end users should never be able to actually see it. It is in opposition to the core idea of VAS input.

(a) VAS input feedback used during the evaluation phase.



(b) VAS input feedback shipped with the final version of prototype.

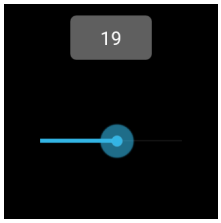
Figure 4.13: VAS input feedback modifications.

4.2.5 Numerical Scale Input

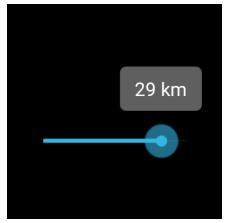
The *numerical scale* input was the last one implemented in the prototype version meant for this project. First designs included a slim progress bar with small indicator used to modify the value. The value itself was displayed above

[9]http://www.physio-pedia.com/Visual_Analogue_Scale

the scale. The users immediately complained about the inconvenience of such solution (due to the small size of draggable indicator) and its lack of coherence with the VAS input designs (which included a much thicker progress bar). After revision, the progress bar itself remained unchanged, but the input value box became interactive. The users could move it along the progress bar, changing the input value and the position of the indicator. That solution not only created a clear distinction between the two scale types offered by the application, but also, according to user feedback, provided a much more intuitive and user-friendly way of recording input. Especially TBC expressed a lot of concerns for the initial design, but did not raise any issues regarding the revised solution.



(a) First iteration - static input value box.



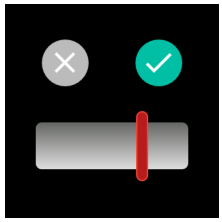
(b) Second iteration - input value box following the progress bar with the unit displayed.

Figure 4.14: Numerical scale input design considerations.

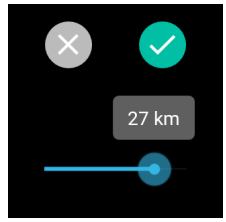
Another issue reported by users after initial evaluation was the lack of an input unit. It caused some confusion when they were using the numerical scale input for tracking two different aspects of their behaviour (e.g. consumption of water and the number of kilometres walked). The unit was incorporated in the later version as a part of the input value box. Furthermore, most of the tracked activities do not need to provide the possibility of recording an empty input (*zero* input in case of the numerical scale). As the definition of the scale range was among the user requests, it was scheduled to be included in the customisation mechanism. Finally, tracking some behaviour issued a problem with the scale size. Capturing the water consumption every time it is drunk required using the millilitre-based scale ranging from 20ml to 200ml, which made it not only hard to use, but also unnecessarily detailed. Thus, a value jump was introduced, allowing the users to select only one out of ten samples ranging uniformly from the minimum to maximum value.

4.2.6 Input Confirmation

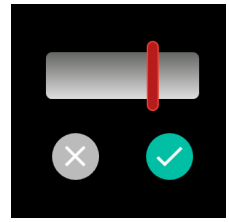
Although the designs for scale inputs were verified, the evaluation resulted in recording numerous accidental or error inputs as user input was captured as soon as the they released their finger from the progress bar or input value box. It was obvious that a confirmation mechanism needed to be introduced. It was designed according to *Android* design guidelines regarding confirmation of user input on smartwatches. An alternative design based on a single *Accept* button was also considered, but was rejected since it would not be consistent with the system UI, and the users found the *Cancel* button in the *two-button* design actually useful for exiting the input screen.



(a) The input confirmation for VAS input.



(b) The input confirmation for numerical scale input.



(c) VAS input confirmation used for testing of inverted positioning.

Figure 4.15: Input confirmation designs.

Designs with switched positions of the confirmation buttons and input scales have also been presented to users. However, they have not yielded considerable results as far as the reduction of error rate or the convenience of use are concerned. At the same time, such solution created additional implementation issues regarding the positioning of the value box for the *numerical scale* input, which eventually resulted in abandoning the idea.

4.2.7 Additional User Input

There were several design consideration raised by the test users that did not result in any design changes, both because of design and implementation issues.

First of all, the lack of dedicated *Exit* option in the input screens caused quite a lot of confusion among the users, especially those who do not use smartwatches on a daily basis. *Android Wear* offers two system-implemented ways of exiting

any application - *swiping right* and *pressing the hardware button*. Those solutions make it possible for the *Exit* button not to be explicitly implemented and provide more flexibility with the design of the application features instead. Due to limited screen capabilities (size, resolution), the number of elements displayed on the screen should be kept to a minimum. Experiments with *exit* dialogue displayed on the *long press* gesture were run in the early stage of designing. Despite initially positive response, this solution proved to be more troublesome when additional input types were introduced - while using the scale inputs, users often launched the *Exit* dialogue instead of moving the indicator or value input box. This problem did rarely occur with experienced smartwatch users, hence an introduction to the system UI should be provided to any first-time users.

(a) System UI *swipe-to-dismiss* mechanism. The user has to swipe screen of the application to close it.

(b) Custom *long-touch* mechanism implemented for evaluation. An *exit* dialogue is displayed and awaits confirmation from the user.

Figure 4.16: Different approaches for exiting the Android Wear application.

Another suggestion concerned introducing the input confirmation or the *Undo* button functionality for binary and category inputs. However, the error rate for these input types is actually very low so adding the necessity of an additional touch was considered a too high interaction cost that could adversely affect the compliance during long-term study. Instead, it was decided to offer the possibility to mark specific input records as errors using the companion application (see *Data display* section).

Finally, some users were not satisfied with the fact that all input types (except for binary) require the launch of additional input screen, which caused slight performance delays. Suggestions about using the advanced gestures (such as swiping, pattern-drawing, etc.) on the watch face were made. Although the idea itself seemed reasonable and attractive, it was quickly rejected due to platform limitations. On the *Android Wear* device watch face, the only accessible gesture type is a single tap, while all others are reserved for the system UI. Technically, it would be possible to add custom gesture recognition, but altering the system UI behaviour is a strongly discouraged practice.

4.3 Companion Application

The companion application was designed following the *Android material design guide*[10], e.g it uses the recommended colour schemes and numerous generic icons. However, additional design considerations had to be taken into account. First of all, the emphasis was put on the simplicity of the designs. The application was supposed to be accessible both by users used to smartphones and technological novices. This approach was appreciated in the evaluation feedback, although some users complained about actual oversimplification. Furthermore, the tab-based navigation design was used for the main screens of the application, utilising swipe gestures and enabling easier access to selective features.

4.3.1 Overview

Mobile application offers several main features, all described in details in the following sections together with design considerations and user feedback.

After launching the application, the user is presented with the main menu that provides access to all functionality via tap or swipe gestures, as shown in Figure 4.17. The combination of both allows to reduce the interaction barrier and provide *one-touch* access to all functionality from the welcome screen without overwhelming the users with the number of available buttons.

The application workflow starts with the creation of the input groups, which can be started both from the main menu and the input group list screen. The input group list screen itself (Figure 4.18) provides the possibility to delete the group or edit its properties. The same input group form (Figure 4.19) is used for both editing and creating new input groups.

Having the groups defined, the users can configure the desired watch face and send it to the wearable application. The watch face configuration screen, presented in Figure 4.20) offers the option to select groups, edit them and preview the changes to the current state of the watch face.

The recorded samples can be accessed via the data display screen (Figure 4.21), which displays the input groups and provides *on-tap* access to detailed recordings. In the detailed view, users can mark the captured data as false positives.

[10]https://developer.android.com/design/material/index.html

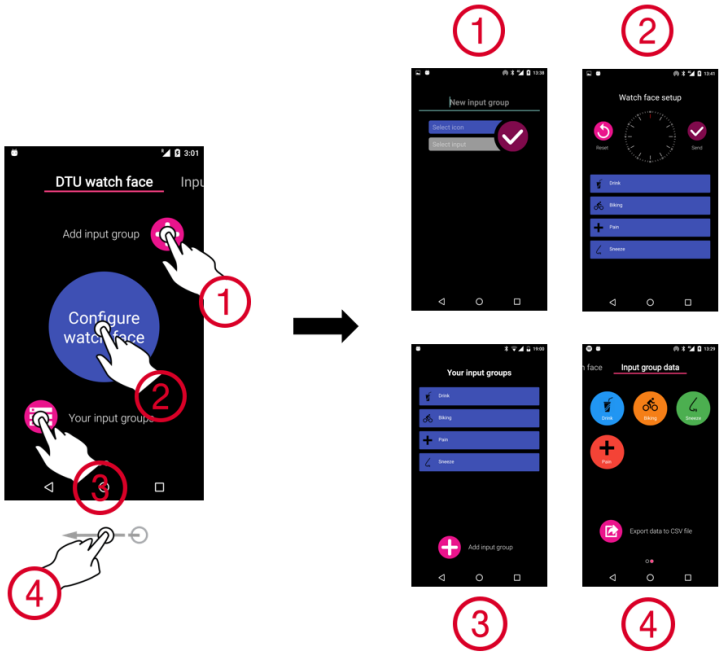


Figure 4.17: Mobile application main menu interaction flow. The user can use tap (1-3) or swipe (4) gestures to proceed to different features - input group form (1), watch face configuration (2), input group management (3) or data display (4).

4.3.2 Data Display

The first iteration of the companion application development included the display of recorded data presented in Figure 4.22. All predefined input groups and all recorded samples were displayed in form of a list and offered the possibility of deleting a sample with a swipe gesture. Early evaluation raised concerns about actually deleting the recordings, since it could seriously affect the results of a study or self observation. The delete functionality was thus replaced by marking individual samples as errors, which would result in changing their background and adding a special annotation. Additional small alterations were introduced to the displayed details of recordings. The date and location data was shortened to fit on the screen and still provide relevant information.

In the next iteration, the users reported confusion between the input group list that provides access to recorded data and the same list used for managing the groups. In order to cope with that, the data access list was transformed into

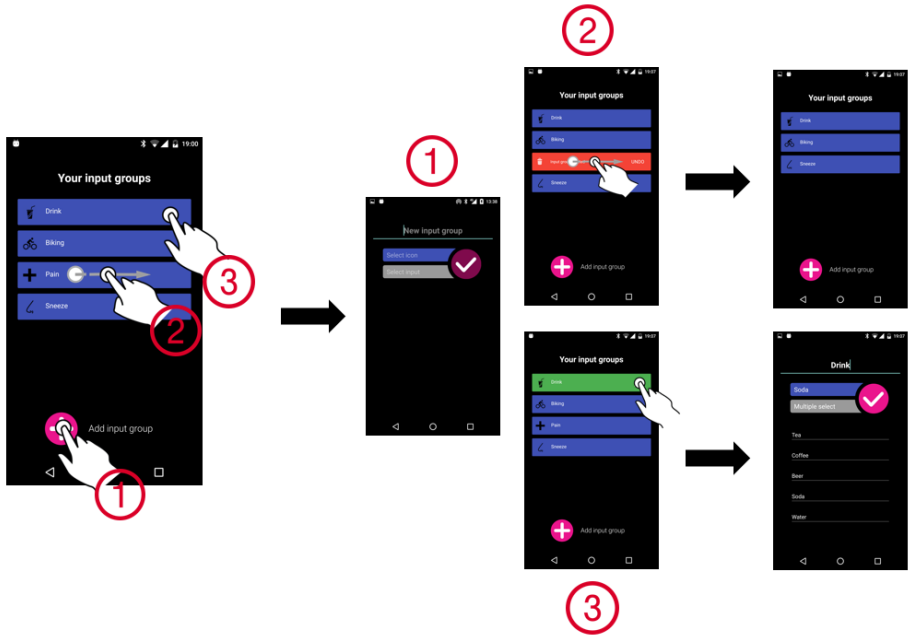


Figure 4.18: Input group list screen interaction flow. The user can select an input group (3) and proceed to the edit form by tapping the *edit* icon; proceed to the input group form directly (1) to create a new input group; or swipe right (2) to delete it.

grid view similar to the one used in native *Android Contacts* application. This approach allowed even better fitting of the groups on the screen while the use of different icons and background colours provided sufficient distinction between them. Additionally, it was pointed out that the *Export* button should contain a text label, similar to the buttons on the home screen, in order to ensure design consistency. Also, without the label, the purpose of the button was not clear to some users.

4.3.3 Input Groups Management

Already the first version of the input groups management (Figure 4.23) consisted of two parts - the list of existing groups (offering selection for edit and delete functionality) and the input group form - and no negative feedback was provided regarding any of them. In the next iteration, the input group form was extended with additional parameters for category and numerical scale input. The users

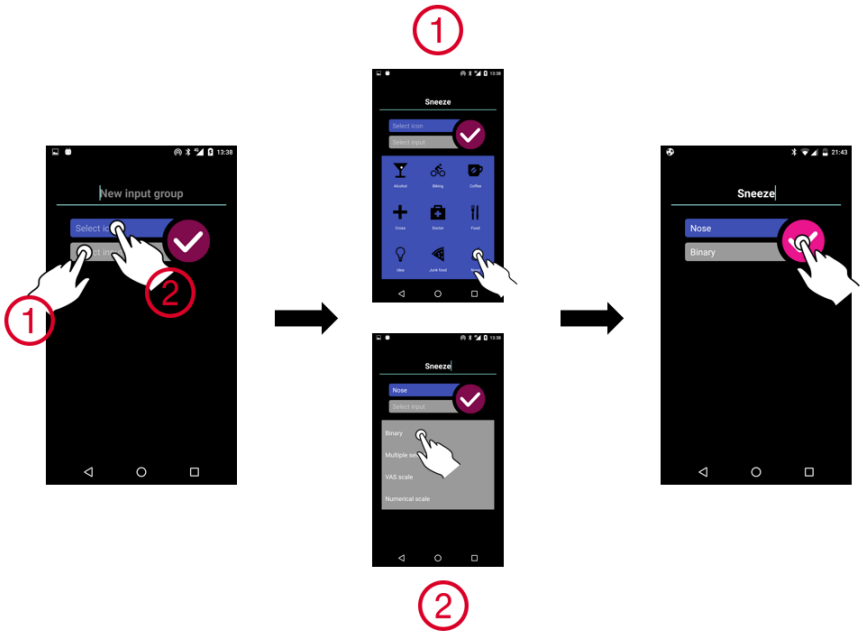


Figure 4.19: Input group form interaction flow. The user is required to provide the name of the input group, the icon to be displayed (1) and the type of the input (2). Additionally, for the *Category* and *Numerical Scale* input types, the user is obliged to provide input parameters - respectively, at least one category value, or the minimum and maximum of the scale, complimented by the measurement unit.

noticed a lack of validation of any of those. Thus, it was important to assure that at least one select option is provided for the category input, and that all numerical scale parameters are set. Furthermore, the minimum and maximum values of the numerical scale range had to be validated with respect to the number format. All feedback for validation errors was provided in the form of *Toast* messages.

Additional consideration that should be taken into account to improve the prototype is the limited number of available input icons. The finite set offered by the current version certainly does not fulfil the needs of all users and all self-tracking aims as the variety of subjective experiences is simply too broad. In order to cope with that, icons with high level of abstraction could be introduced. Potentially, they could represent a whole group of activities or a specific aspect of life, not the specific activity itself.

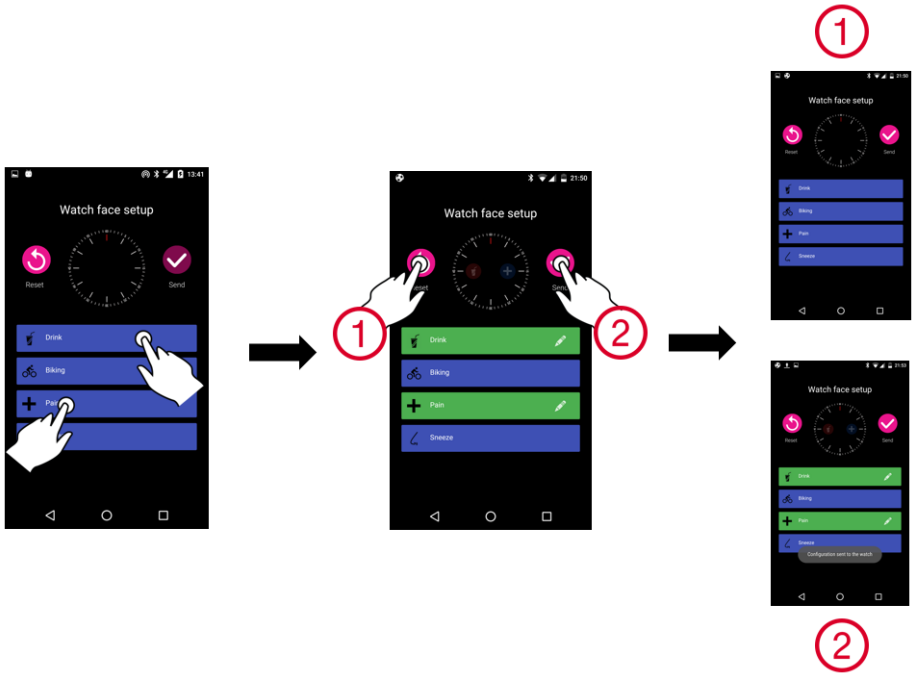


Figure 4.20: Watch face configuration interaction flow. The user can select the input groups by tapping them and then send the configuration to the wearable application (2) or reset the preview (1). In case of successful configuration sync, a *Toast* message is displayed as a feedback.

4.3.4 Watch face configuration

The original designs of the watch face configuration screen (Figure 4.24) included a list of available input groups, which selected in the right order, could be sent to the smartwatch. As noticed, it provided very little feedback about the current and target state of the watch face. The next iteration added a preview of the watch face and a button to reset it. The preview would originally display the current configuration and then offer a live preview while user modified the selection of input groups. Consequently, the *Reset* button could revert the preview to the current watch face state. The evaluation response to those changes was very positive, however, the users could still select more input groups than could actually be used in the configuration. The last iteration provided a solution by limiting the selection to four input groups (a maximum that proved to be useful) and providing a *Toast* message in case of choosing more.

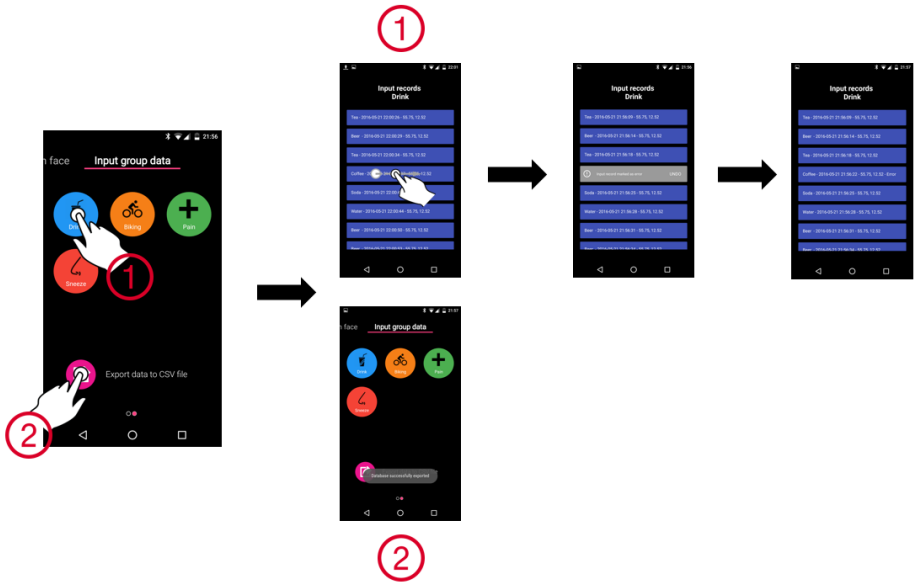
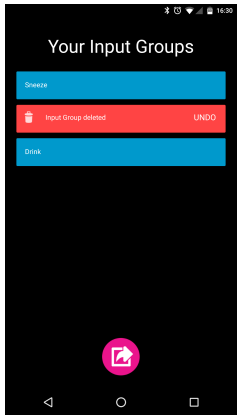


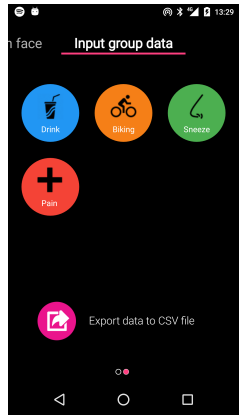
Figure 4.21: Data display and management interaction flow. The user can export the input group samples (2) or view detailed information about recorded samples (1). Individual samples can then be marked as false positives using swipe gesture.

4.3.5 User Experience Revision

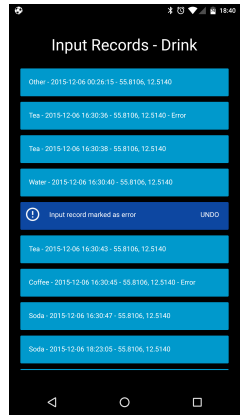
While wrapping up the work on the prototype, the users seemed satisfied with current application designs; however, a consultation with an User Experience (UX) designer revealed potential for further improvements. One very important change that was suggested concerned the navigation between different views of the home screen and described the circle indicators as not intuitive enough. Since designs for different features were tested independently, this functionality lacked proper evaluation during any of the design iterations. To provide better and self-explanatory solution, a title view indicator was added on top of each view instead of standard headers, as shown in Figure 4.25.



(a) First iteration - input groups as a list view with delete possibility.

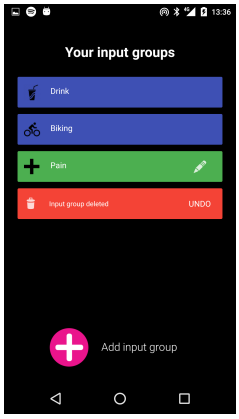


(b) Second iteration - input groups as a grid view.

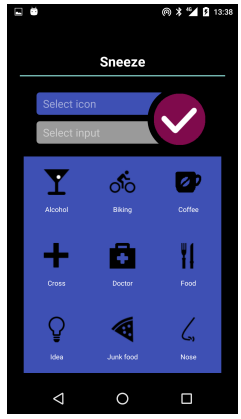


(c) Recorded samples as a list view with error marking.

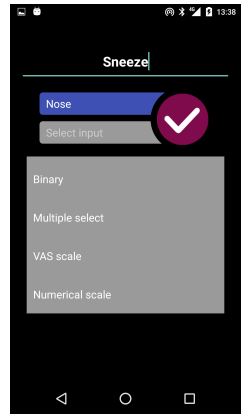
Figure 4.22: Data view designs throughout the design iterations. The grid view with round icons replaced the list view display in order to provide clear distinction between data view and input group management view. The recording details view remained unchanged since first iteration.



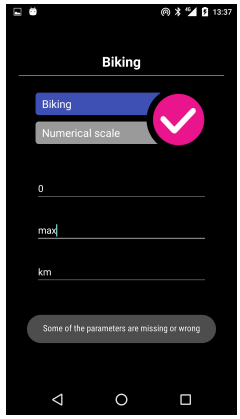
(a) Input group list with edit and delete possibility.



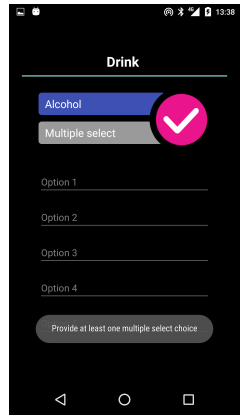
(b) Input group form with active icon select grid view.



(c) Input group form with active input type select grid view.

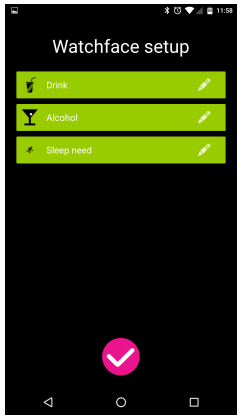


(d) Numerical scale input additional parameters with validation.

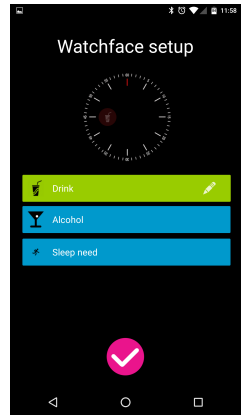


(e) Category input additional parameters with validation.

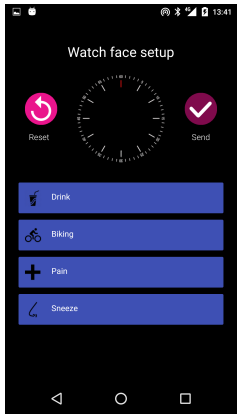
Figure 4.23: Input group management designs, including the input group list and the input group form. The save icon of the form becomes active after the user inputs the name of the group, the icon to be displayed and the input type. The validation of additional parameters (for *Category* and *Numerical Scale* input types) occurs at the tap of the save icon.



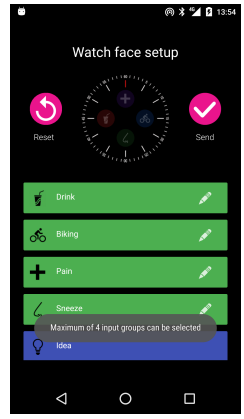
(a) First iteration - input group list.



(b) Second iteration - visualisation of selected input groups.

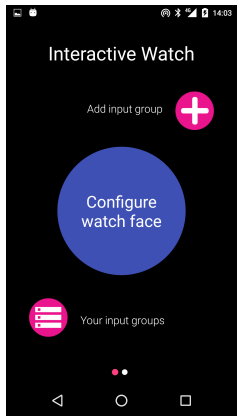


(c) Third iteration - live preview with reset feature.

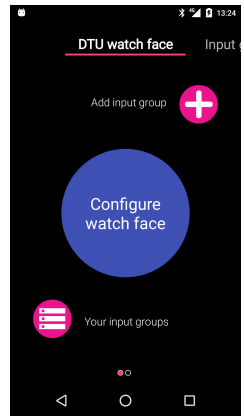


(d) Fourth iteration - input group select validation.

Figure 4.24: Watch face configuration designs throughout design iterations. The designs gradually evolved from the simple list of available input groups, through the live preview of the new configuration, finishing with the preview of current watch face state and the reset functionality. Additionally, the validation of the number of selected input groups was introduced in the form of a *Toast* message.



(a) First iteration - circular tab indicator.



(b) Second iteration - title tab indicator replacing standard titles.

Figure 4.25: Main menu navigation designs. The review of the intuitiveness of the application resulted in changing the navigation designs and introducing title indicators for each available feature tab.

Chapter 5

Implementation

Implementing the designs discussed in previous chapter posed certain challenges throughout the whole process, which have been solved using external libraries, custom implementations or by revising the design concept itself. In this chapter, all the non-trivial issues are described with respect to both smartwatch and companion application. The implementation of features that are not described relies on *out-of-the-box* functionality of the respective platform and follows the official documentation or tutorials

5.1 Wearable Application

The wearable application was developed for Android version 5.0.1 at minimum. It has been tested on *Sony Smartwach 3* and *LG Watch Urbane* devices running *Android 5.0.1* and, for more recent prototype version, *Android 6.0*. As described before, alpha versions of prototype were shipped to the test users and evaluated continuously for a duration of between 13 and 21 weeks. This approach allowed the thorough verification of system robustness and accuracy.

The overall architecture of the system consists of the watch face service used as an entry point to additional input activities. All of these contain dedicated

code for synchronising user input with the companion application via custom synchronisation utility class. Finally, a wearable listener service is implemented to collect configuration data from the smartphone and provide it locally.

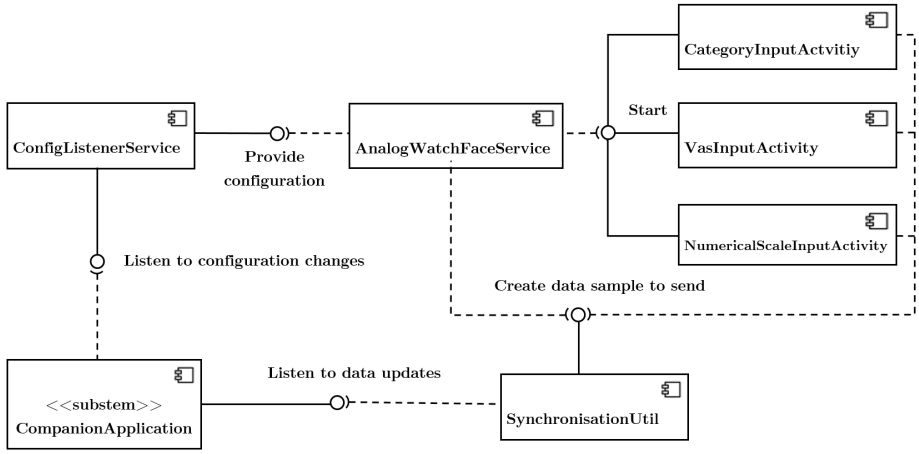


Figure 5.1: Wearable application architecture overview. Solid lines represent provided feature while dashed lines stand for utilising it.

To implement all required functionality of the application and ensure its agreement with the designs, a set of external libraries had to be used. The overview of these is presented below, while the motivation for and details of their usage are discussed in the following sections.

- **Google Play Services** library responsible for communication between both applications and location updates;
- **Android Support Wearable** and **Palette** libraries providing implementation of watch face service and color processing for watch face ambient mode;
- **MPAndroidChart** library used for the *Category* input type implementation;
- and the **SeekBarHint** library code fragments modified for the purpose of *Numerical Scale* input implementation.

5.1.1 Watch Face

Android Wear devices allow the implementation of custom watch faces for smartwatches, like the one described in the *Design* chapter, Section 4.2.2. For the purpose of this solution, the CanvasWatchFaceService was extended in order to draw the background, the watch hands and all input-related icons. Drawing pixels on the low level canvas requires using the screen coordinates of the desired objects. This limitation means that all object sizes and positions need to be scaled according to the screen size. To achieve that, all measurements were taken on the development device (Sony Smartwatch 3) and are dynamically scaled with respect to its specification (280x280 pixels). The *Android developers tutorial*[1] was followed to implement the necessary functionality, including refreshing the screen, handling watch mode changes (burn-in protection, ambient, standard), etc.

The *Android Wear* update published in August 2015 introduced the possibility of adding interactive features to the watch faces. Prior to it, recognising user input on the watch face required a lot of modifications to the core system libraries and was strongly discouraged by *Google* itself. The current version, however, allows to get the screen coordinates of a tap event, which is crucial for the solution discussed in this thesis. The input icons are drawn on the watch face in specific positions, predefined in an *enum* class. This *enum* provides both the position of the icon on the screen and the order in which the positions are drawn. A similar list is defined for the background colours of the icons. Comparing the touch event coordinates and the current icon positions allows the recognition of whether the event happened within a predefined distance (radius) from the position, and defines the enabled input region.

One additional issue related to the watch face behaviour concerned exiting the input *Activities* using the hardware button of the watch. Such action resulted in *hiding* the *Activity* instead of *finishing* or *destroying* it. Trying to record a new input would resume the old *Activity* and put the new one on hold. To prevent that, the default behaviour of *Activity's onStop* method (executed when the activity is being hidden) was changed to actually *finishing* the *Activity*.

5.1.2 Category Input

The category input screen provides a select wheel with predefined input options (as presented in *Design* chapter, Section 4.2.3). Different implementation approaches were considered for that functionality, including dynamic drawing

[1] https://developer.android.com/training/wearables/watch-faces/index.html

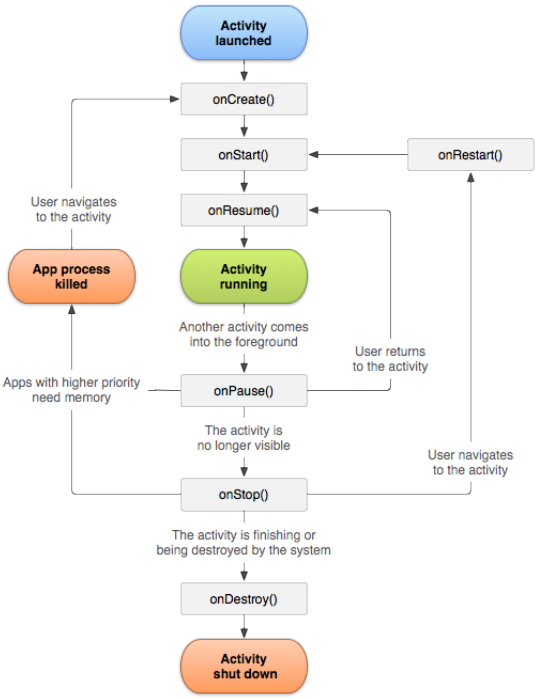


Figure 5.2: *Android Activity Lifecycle*[2] illustrates the difference in execution of *onStop* and *onDestroy* methods.

using canvas, providing a set of backgrounds and covering them with transparent input fields, or just displaying the option names around a circle. Eventually, a non-trivial solution has been identified - implementing a select wheel as a 2D pie chart. Charts are usually used for visualising data and providing user feedback, but for the purpose of this prototype they were used as input mechanism, mostly because of their ease of use.

For that purpose, and to simplify the development process, the *MPAndroidChart*[3] library was used. It provided not only the drawing of the input options themselves, but also high customisation features (including removing the centre of the wheel, colour selection, input feedback, etc.) and the possibility of interaction (returning the index of the option chosen by user).

[2]https://developer.android.com/reference/android/app/Activity.html
[3]https://github.com/PhilJay/MPAndroidChart

5.1.3 Numerical Scale Input

The scale offered to the users for the numerical scale input posed a bigger challenge. Specifically speaking, the draggable input value box did. No *out-of-the-box* solution for that purpose is offered in *Android* and external libraries do not provide enough customisation options to adjust to the smartwatch limitations. In order to implement the designs presented in the *Design* chaper, Section 4.2.5, a custom solution based on the *SeekBarHint*[4] library was developed. The library code used was responsible for displaying the input value box above the *SeekBar* object and following the current progress indicator. However, it had to complimented with gesture recognition for said box.

A mechanism for recognising the type of user gestures (pressing down, moving or releasing a finger) was implemented, followed by redrawing the input value box at the new position. Since the box position had to be defined in pixels, the scaling feature had to be added, as described in the *Watch face* section. Finally, a recognised finger release results in displaying the input confirmation buttons.

5.1.4 Synchronisation

The synchronisation of data between the watch application and the companion application relies on the *Wearable Data Layer* of the *Android* platform. Android offers various ways of data exchange between paired devices. The *Message API* provides a one-way communication mechanism, meaning that it assures no synchronisation between the handheld and wearable application. Hence, it is useful mostly for remote procedure calls (RPC), such as sending start requests to remote activities. Transferring *Assets* on the other hand, is meant for big chunks of binary data, such as images. For the purpose of this project, a third solution was chosen - synchronising the *DataItems*.

DataItems have payload (content) limited to 100kB, which is perfectly enough for the two main synchronisation features necessary in the application - receiving the configuration from the companion application and sending the recorded input. Aforementioned limitation would cause problems with sending the configuration including the icon images; therefore, the image files were included in both applications, allowing the synchronisation only of the image metadata. To assure the consistency of the data sent, the same set of configuration *enum* classes was used in both applications. Those *enum* classes included *input types, icon positions, icon images and colour schemes*. Such approach allowed the synchronisation data to be reduced to a simple set of *string* values.

[4] https://github.com/moondroid/SeekBarHint

The first version of the system used the *DataListener* interface for listening to configuration changes. That approach, however, created a need for the watch face or input activity to be active in order to synchronise data. To avoid that, a dedicated implementation of *WearableListenerService* was introduced that constantly awaits configuration changes. The lifecycle of said service is managed by the system, putting it on hold when no change events are requested by any of the sides. The listener service uses the *SynchronizationUtil* to provide the local storage of the configuration data and the interface for locally accessing it by the watch face service, assuring that the watch face service is served with latest configuration version when activated.

To synchronise recorded input data, the watch face service and all input *Activities* implement the *DataApi*-based code for sending *DataItems*. To avoid code duplication, the *SynchronizationUtil* is used to provide the synchronisation request containing record data (input group, input value, date and location data).

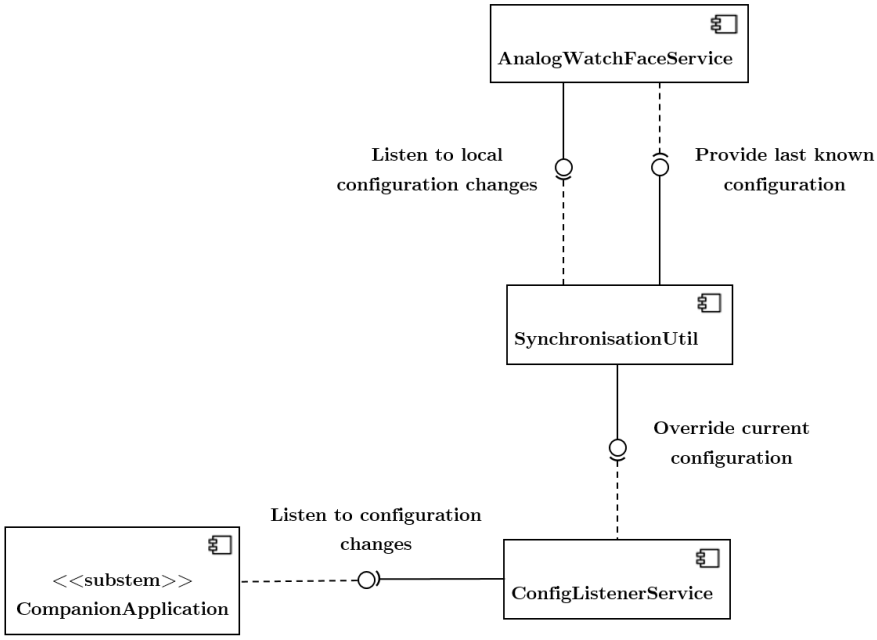


Figure 5.3: Wearable application synchronization mechanism overview. Solid lines represent provided feature while dashed lines stand for utilising it. The *SynchronizationUtil* is responsible for providing the configuration configuration for watch face service at all time.

5.1.5 Location

Note that all following solutions rely on the Google Location Services API, mainly FusedLocationApi.

To provide accurate contextual data, the watch application has to include the location information for every recorded sample. Since most smartwatches incorporate both a WiFi connection and a built-in GPS sensor, collecting this data does not pose special difficulties, except for assuring the right balance between the accuracy of the data and the performance of the system. Moreover, testing of the location feature created additional challenge itself, as some implementations could be verified after extensive (with respect to time and location changes) usage. Consequently, in some cases, problems were reported only after weeks of continuous testing.

The first implementation of the location service relied solely on the *getLastLocation* method, which returns last known device coordinates and failed because of several issues. First of all, it falsely assumed that Android smartwatches contain services or system applications that constantly request device location. Every reset of the device or update of *Google Play Services* (responsible for the *Location Services API*) resulted in resetting the last known location. Accordingly, the location data was simply inaccurate since it could not be updated for hours or even days.

To cope with the location updates, the second iteration included a *LocationRequest* in the watch face service that constantly monitored the device location. Although the location data turned out to be accurate, the battery life suffered drastically. During the first days of testing, the smartwatch battery was drained in less than 7-8h, comparing to its normal lifespan of over 48h. Such results immediately disqualified the concept of constant location updates and even prevented the shipment of the prototype version for alpha testing.

The next version of the prototype limited the location updates. There were two implementation considered for achieving that - both including certain advantages and drawbacks. One was requesting the device location only when recording the input. The record data would have to be cached in the device memory until the location update is received and then sent to the handheld device. Unfortunately, in case of location not being available, the records could be cached for a very long time, which would result in data inaccuracy. Even worse, the watch running out of battery or simply being switched off, would result in losing the cached data. Moreover, caching a lot of records could affect the performance of the application, due to high memory usage. Although those issues could be partially solved by more sophisticated synchronisation mechanism (in-

cluding device state recognition and memory-cleaning policy), the concept was put aside for the sake of simplicity, time and ease of implementation.

The chosen solution relied on the *LocationRequest* used for continuous location monitoring, but with updates requested only when the watch face was in the active state and connected to the *Google Play Services*. Consequently, the request was removed whenever the user was not using the watch. Although the battery life improved significantly and was close to normal usage, the response time of the location update raised certain concerns. Potentially, the time between the watch face wake-up and the user recording the input could be shorter than the update of the location, e.g. due to poor connection. In such case, the *last location* data would be inaccurate, or even non-existing if the watch was recently reset or updated. Nevertheless, this version was selected for extensive testing, the results of which are presented in *Results and Discussion* chapter.

Implementation issue regarding the precision of location were also discussed during the whole implementation process. *Android* offers two types of location updates - *coarse* and *fine*. *Coarse* location relies on the WiFi and/or mobile data connection and provides accuracy approximately equivalent to a city block, while *fine* location uses any available location providers, also including the GPS, and aims at providing the most precise location data currently available (with an accuracy down to a few meters). In the majority of cases this project intends to address, the *coarse* location would probably suffice. However, since using the *fine* location did not significantly affect the performance of the application or the battery life of the smartphone, it was decided to provide more accurate data, at least for the testing phase.

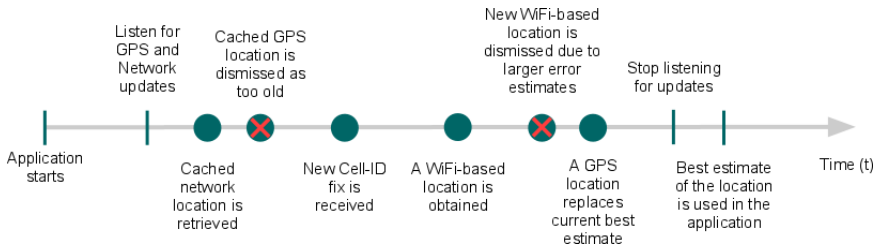


Figure 5.4: Action flow for obtaining *fine* user location using *Google Location Services API*[5].

5.1.6 Permissions

Requesting the device location update requires specific permissions in *Android*. Since the application was originally developed for the 5.0.1 version, they were all

defined in the *wear* module *manifest* file. Standard wearable applications install automatically on the device whenever the companion application is installed on the smartphone, which also means that the included watch faces are instantly available for the users. Unfortunately, since the developed application needed access to the location updates, it did not install until explicitly confirmed in the watch application menu. That also caused the watch face to be invisible in the watch face menu, which caused a lot of confusion and irritation for test users during the installation process. Considering Android's policy regarding permissions, it seemed that there was no solution for that issue.

The issue changed with the 6.0 system update rolling out for *Sony Smartwatch 3* devices in the beginning of April 2016, completing the list of *Android Wear* devices with latest system version[Gho06]. The new permission system defines sets of *normal* and *dangerous* permissions. The former are considered to be less intrusive, are granted for the app automatically and include vibrations, Internet access, checking network state, etc. The latter, however, require the developers to implement explicit confirmation request displayed at run time and handle both the acceptance and rejection of the permission. Consequently, the wearable application no longer required the installation confirmation and was provided with the explicit permission verification at the start of the watch face service. Since the permission request cannot be handled by any service, an additional *Home* activity was implemented, with the sole purpose of asking for permission and displaying a short message manual.

At this point, the implemented solution is neither perfect, nor complete, mostly due to the short time frame since the introduction of the update in April 2016. Although it works correctly if the user accepts the permissions, it lacks feedback when they do not. An additional confirmation message explaining the need for location updates could be introduced in that case. Additionally, since the application continues to work and simply provides no location data, a system status icon indicating lack of location updates should be introduced.

5.2 Companion Application

The companion application was developed for *Android* version 5.0.1 at minimum. It has been tested on *Google Nexus 5* and *Nexus 5X* devices running *Android 6.0*. The alpha versions of the prototype were shipped together with the wearable application to maintain system consistency. Development of this solution created challenges described in the following sections.

[5]https://developer.android.com/guide/topics/location/strategies.html

The back-end architecture of the companion application consists mainly of *Android Activities, Fragments* and *Adapters*, supported by a few features to facilitate the correct behaviour of the whole mechanism and including:

- **SQLite configuration and helper classes** for persisting data,
- a **synchronisation utility class** used to send configuration data to the watch,
- an **export utility class** for providing CSV[6] export of data.

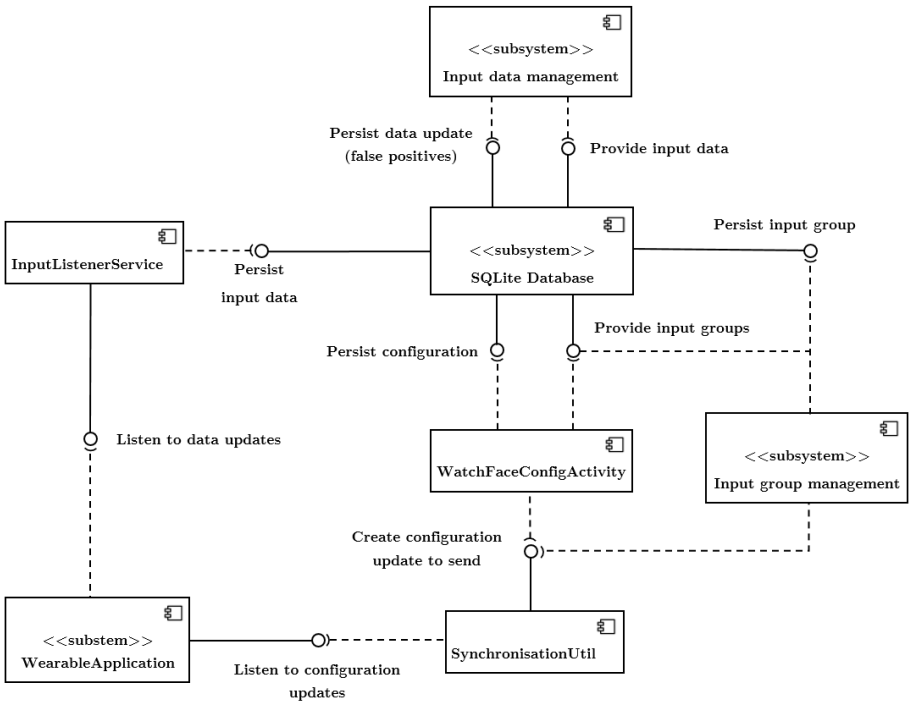


Figure 5.5: Mobile application architecture overview. Solid lines represent provided feature while dashed lines stand for utilising it. Subsystems represent a set of activities and related classes responsible for one system feature.

Numerous external libraries and APIs were used in order to implement the desired functionality. They are discussed in details in the following sections and include:

[6]Comma-separated values that store tabular data (numbers and text) in plain text

- **Google Play Services**, as discussed in the *Wearable Application* section;
- **Android Support RecyclerView** and **CardView** libraries used for displaying records in the list form;
- the **android-swipe-to-dismiss-undo** library providing the implementation for using swipe gestures with the *CardView*;
- and the **ViewPagerIndicator** library simplifying the tab-based navigation mechanism.

5.2.1 Main Menu

The application's home screen, described in *Design chapter*, Section 4.3.1, was implemented using the *Android Fragment* interface in order to provide the tab-based navigation system between the two main parts of the system - the configuration and the data. To guide users through the navigation system, the external *ViewPagerIndicator*[7] library was used, specifically the *TitlePageIndicator* and *CirclePageIndicator* views. The library was chosen due to simplicity of use and the customisation options it offers, but it turned out that it lacks default styles for said indicators. The styles were removed between versions 2.2.3 and 2.3.1 for unknown reasons and had to be restored from the release history.

Using the *Fragments* also affected the use of *RecyclerView Adapters* (see *Listing* section for details. Usually, the data reload for said view happens with the *create* event of the activity, however, it has no effect while using *Fragments* as their state is being preserved. To overcome this issue, the data in such case was reloaded manually with the *resume* event of the fragment.

5.2.2 Listing

Recent trends in *Android* application design suggest using the *card view* design to maximise user experience clarity. To provide that, the *RecyclerView* and *RecyclerViewAdapter* system interfaces were implemented (see *Design chapter*, Section 4.3.2 and 4.3.3). Additionally, the *swipe-to-dismiss-undo*[8] library was added to the project to offer the delete/mark-as-error functionality for swiping gestures. Although it provides very friendly customisation options and ease of implementation, it has not yet been adjusted to specific class changes introduced in *Android API 23* (Android 6.0). To make the code compilation possible, a

[7] http://viewpagerindicator.com/
[8] https://github.com/hudomju/android-swipe-to-dismiss-undo

new version of *SwipeableItemClickListener*, overriding the *onRequestDisallowInterceptTouchEvent* method, was implemented.

Furthermore, to make sure that the *RecyclerViewAdapter* is reloaded correctly after the data set is changed, all form *Activities* starting from *RecyclerViews*, are to send the result information back to their parent. Such information, collected by the *onActivityResult* method, triggers the manual reload of the data.

5.2.3 Watch Face Configuration

The main issue regarding the watch face configuration feature, presented in the *Design* chapter, Section 4.3.4, concerned the live preview of the watch face itself. In the current versions of *Android*, drawing on the screen is handled by the canvas mechanism. Watch face background image and aforementioned configuration *enum* classes were used to draw the representation of the watch face. Analogously to the drawing mechanism described in Section 4.3.4, proper canvas scaling had to be introduced. The original measurements were taken on the *Nexus 5* smartphone and are scaled dynamically accordingly to the recognised device screen size.

To be able to adjust the size of the drawing region, the canvas was implemented in the *CustomCanvasView* class extending the standard system *View* class. Moreover, it made it possible to introduce additional methods for dynamically altering the drawing - e.g. by changing the set of displayed icons or by changing the colour scheme used.

Finally, to avoid unnecessary communication with the wearable application, it was decided to save the current watch face state in the mobile application. Although it would be technically possible to read it every time the configuration activity is launched, such approach assured the simplicity of implementation and allowed additional features to be included in the application (see *Input group form* section).

5.2.4 Input Group Form

The input group form view (see *Design* chapter, Section 4.3.3) offers a few features that required non-trivial implementations. First of all, when the user switches between the input group name input and the icon select input, the active on-screen keyboard had to be closed manually from the code using the system *InputMethodManager* class.

Some considerations were also given to the validation mechanism of additional input parameters. Live validation on every change of focus seemed to be a little excessive considering the simplicity of the form. Hence, two methods validating the whole parameter set were implemented and are called while the user intends to send the configuration to the watch.

Thanks to having the current watch configuration stored on the watch, the automatic synchronisation feature could be implemented in the second iteration of the process. While saving the changes to the input group, its database object ID is compared against the current configuration. In case the edited group is a part of the setup, the introduced modifications are immediately synchronised with the watch and the user feedback is provided in the form of a *Toast* message.

One feature that caused a lot of unexpected problems was the *GridView* displaying the possible input icons. For unknown reasons, it used to change the images of random icons when displayed immediately after filling the input group name text field. Changing the predefined *visibility* property of the *GridView* element from *GONE* to *INVISIBLE* seemed to solve the issue, but its origins remain undiscovered.

5.2.5 SQLiteDatabase

Android offers several storage options out of the box. However, using *shared preference* offers too little functionality for the purpose of the solution and storing data in a file on *internal* or *external storage* would not provide necessary structure for the data. Considering the widespread nature of SQL databases and their availability on *Android*, the choice of using a private database seemed pretty obvious.

For handling database updates and seeding with initial data, a *MySqlHelper* class was implemented as an extension of the system *SQLiteOpenHelper*. In the initial versions of the prototype it was responsible for filling the database with a predefined watch face configuration. However, with the configuration mechanism implemented in next iterations, this functionality became obsolete and was removed. Nevertheless, the extension of the database structure created a need for an update mechanism, which would execute all relevant update scripts after comparing the latest version of the database with the one on the mobile device.

5.2.6 Synchronisation

Analogously to the synchronisation mechanism described for the wearable application, the handheld application uses the *WearableListenerService* in order to provide background data synchronisation. The only non-obvious issue that had to be addressed in its implementation was handling the lack of location data in the incoming *DataItem*. In such case, the service would request last known smartphone coordinates and include them in the database record. To distinguish between accurate location information (originating from the smartwatch) and approximate data collected *post-factum* by the handheld device, an additional field was introduced to the record object. This field, called *locationSource*, would keep one of the *enum* constants describing the origin of the saved location and could provide feedback during data analysis.

Chapter 6

Evaluation Methods

As mentioned in the *Design* chapter, the first line of evaluation for the application iterations consisted of *quick-and-dirty* and *thinking aloud* methods. In the early stage of feature development low-fidelity prototypes were created in the form of graphical designs and interactive mock-ups based on a *PoP*[1] application. It was crucial to provide the test users with a sense of interactivity, even though no business logic was ready at this point. Being able to actually touch the interface and observe results proved to be really encouraging when it came to gathering such feedback from fellow students, friends and family members. This evaluation phase had an informal character in the form of open discussion about concepts, ideas and solutions. The feedback was usually noted down and stored to prepare for the actual implementation.

The alpha versions of implemented features were shipped as an executable application to the project supervisor Jakob Eg Larsen (JEL) and Thomas Blomseth Christiansen (TBC), a technologist and self-tracker with 5+ years of experience within *Quantified Self* field. Throughout the whole development process, they were provided with a new prototype version every two weeks on average. This approach provided continuous feedback combining reflections on earlier versions and potential guidelines for future development. It was usually delivered and openly discussed during weekly meetings of the MSc student group under super-

[1] Prototyping on Paper - https://popapp.in/

vision of JEL. The ability to start a dialogue with students working on projects in the same field was an additional advantage of the approach.

The feedback from both evaluation methods resulted in numerous significant changes to the project. In some cases, they included only minor design consideration or technical approach changes, while in others the scope of the whole solution was revised.

The following sections of this chapter describe the formal performance test conducted with the final version of the prototype and the data collection process involving the continuous test users.

6.1 Performance Testing

In order to verify the usability of the application as an input device, a quantitative study was conducted among a group of 11 people. It aimed at comparing the proposed solution with existing self-tracking input methods, such as the on-screen widgets or mobile application, with regards to the ease of use quantified by the required interaction time.

6.1.1 Participants

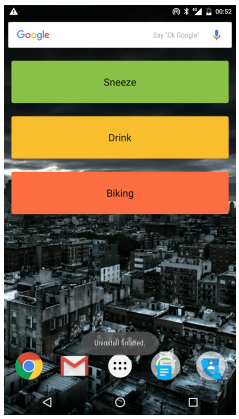
A total of eleven individuals participated in the performance tests conducted at three different occasions. The first group consisted of members of the *DTU Volley* sports club, the second of selected employees of *Exiqon A/S* company and the third of friends and family members. The participants' demographic and educational information is presented in Table 6.1. All subjects were aged between 23 and 56 (but only three of them were over 30), and fluent in either English or Polish. All of them were long-time users of smartphones (mostly *Android*, but also *Blackberry* and *iOS*), but had no experience with smartwatches (except for running watches, which only provide limited user feedback and basic hardware buttons interaction). At most 4 participants could have been considered tech-savvy with their interest in modern technologies and electronic gadgets. All data was collected in April and May 2016.

	Mean	Median	Standard deviation
Age	33.2	26	12.11
Years of higher education	5.6	5.5	1.85
Background (IT/E/O)	3/5/3		
Sex (F/M)	3/8		

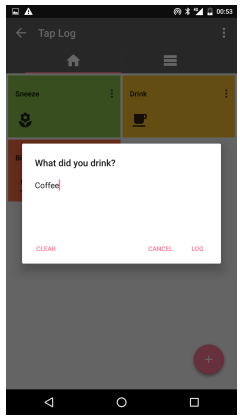
Table 6.1: Demographic data of the participants of the performance test. Abbreviations: *IT* - Information Technology, *E* - Engineering, *O* - Other, *F* - Female, *M* - Male.

6.1.2 Apparatus

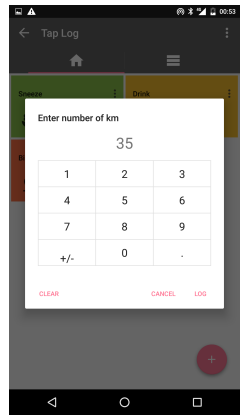
The research required the users to record specific samples using the proposed wearable application, and the widgets or the full version of *TapLog*, a standalone application for the EMA. *TapLog* was selected as its functionality aims to solve a problem similar to the one discussed in this thesis - the user interaction barrier of data collection. It provides the opportunity to record different types of input effectively via a simple and minimalistic UI. It offers the customisable on-screen widget system and supports binary, text and numerical input.



(a) Application widgets



(b) Text input



(c) Numerical input

Figure 6.1: *TapLog* application screenshots with the performance test configuration of on-screen widgets.

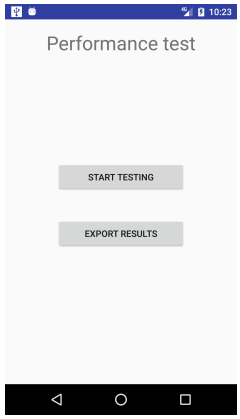
Possible alternatives to *TapLog* could include other tools mentioned in the *Related Works* chapter, such as *movisensXS* or *ESM Capture*. Although their general functionality is closer to the envisioned final state of the proposed solution, they are meant for conducting structured and scheduled studies over longer periods of time. Hence, their level of complexity seemed too high for the purpose of the performance test, which aims at comparing the interaction times only. Other options, such as paper diaries or log books were disqualified as they did not seem competitive enough comparing to modern technologies, as presented by Barrett and Barrett[BB01] or Stone et al. [SSS+03].

To facilitate the performance test, a custom external application was created for *Android* platform. Not only did it provide the participants with guidance throughout the test, but it also captured the times between successive recordings, allowing the comparison of interaction times. The application included an introduction to the whole process and short instructions and proceed buttons for every stage of the test run.

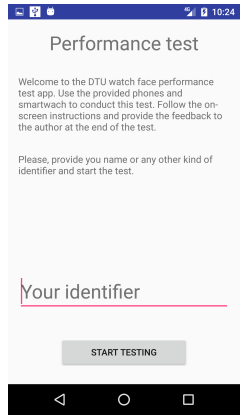
The full setup consisted of the *Android Sony Smartwatch 3* running the wearable application described in previous chapter and including a watch face preconfigured for the purpose of the test, a *Google Nexus 5* smartphone with similarly configured *TapLog* application and on-screen widgets, and another *Google Nexus 5* smartphone running the guidance application.

6.1.3 Procedure

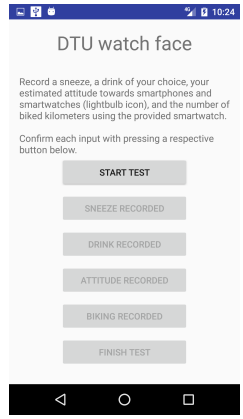
Apart from the guidance provided by the test application, every participant was given an introduction to the process and explanation of the tasks they were supposed to execute. Two trials were conducted for every user and included recording of three inputs using three different input methods (the wearable application, the on-screen widgets and the mobile application). In the first run, subjects were left without any description of the applications and were also allowed to keep the recording phone in front of them. Such approach aimed at testing the intuitiveness of the recording process. The second iteration assumed that the participants are more accustomed to the software, thus, the smartphones were to remain in their usual holding place (depending on participants' habits, e.g. pocket or purse) between individual recordings. Its purpose was to illustrate a real-life situation and concentrate on user-friendliness and ease of use of the solution. The users were not instructed to explicitly mark the error samples; however, the conductor of the test observed their interaction and noted down the numbers of input mistakes made.



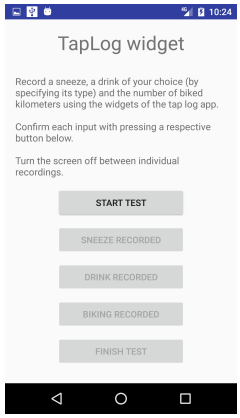
(a) The main menu.



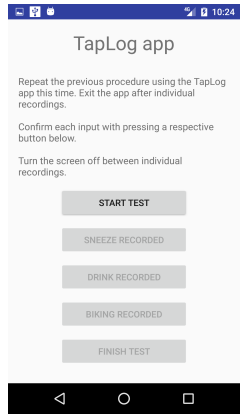
(b) Welcome screen of the performance test, containing the user identifier input field.



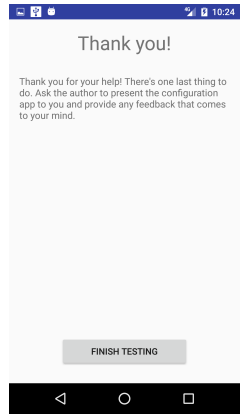
(c) DTU watch face test instructions.



(d) TapLog application test instructions for using the on-screen widgets.



(e) TapLog application test instructions for using the full version of application.



(f) Qualitative feedback view.

Figure 6.2: User guidance application for performance testing. Each screen provides the user with short instructions for the current task and buttons for documenting the progress. The interaction times are recorded with each step of the performance step.

At the end of the test, the first group of users was asked about their qualitative feedback regarding the application. They were presented with the final version of the mobile interface and asked to perform the *thinking aloud* evaluation. At this point, other groups (evaluated later) were presented with a short questionnaire, which contained questions devised from aforementioned qualitative feedback.

The amount of entries collected during the experiment was 11 participants x 2 test runs x 3 input methods x 3 inputs to be captured = 192 samples. These samples, aggregated and averaged, served as a base for the results analysis.

6.2 Data Collection

A very important part of continuous evaluation included data collection. Over a period of 13 to 21 weeks, JEL, TBC and the author were capturing one of the aspects of their daily life - respectively sneezing, skin itching and water consumption. All recorded samples were used at the end of the project to perform a short analysis and test visualisation approaches. It was, however, crucial that no data was lost during the process, which proved to be a challenging task due to the need of updating the application together with erasing the database. That is why the export functionality was introduced in the application. It provided a mechanism to generate CSV files from data samples and save them in the internal storage of the device. This way, the results were easily accessed by the users and shared for analysis.

The data processing included time-series visualisations and location mapping. The first one was expected to provide information about time patterns with respect to various activities and potentially result in behaviour changes. The latter should not only allow identification of user activity regions, but also help recognise geographical regions of increased susceptibility to relevant behaviour factors, e.g. places with concentration of plants being a source of allergy (sneezing).

Chapter 7

Results and Discussion

The formal results collected during this project included the performance test results and data collected during continuous testing. The informal feedback gathered during the design and implementation phase has already been discussed in respective chapters.

It is important to mention that the internal testing can provide feedback regarding the design, the implementation or the performance of the application, but will not give full overview about the behaviour of the users and their interaction with the application on a daily basis. For that purpose, an in-depth experiment with numerous participants would be necessary.

7.1 Performance Tests

The performance test aimed at comparing user interaction time between different methods of registering input - the smartwatch, the smartphone widgets and the smartphone application. Collected results, included in Appendix A, show clear differences in favour of the proposed solution as average interaction times are reduced by approximately 30% on average (see Table 7.1 for detailed information). However, it is crucial to mention that both test runs are laden with specific biases. None of the test users that participated in the performance

test had experience with using wearable devices, but they all were experienced smartphone users. That is why, the unsupervised test run significantly favoured the handheld solution. On the other hand, the second run put the smartwatch in a benign position because of two reasons - the users being already slightly accustomed to the device and the necessity to keep the the devices in their *usual holding* location. The *usual holding* position meant that the users were to put the devices in the place where they would keep them on a daily basis, which turned out to be respectively a wrist for the smartwatch and the trousers pocket or a purse for a smartphone. Of course, one could argue that such placement is not mandatory, but this particular test run intended to just imitate the real life situation. In such, most of the users carry their handheld device on the body (in a pocket, on the belt) or nearby (in a jacket or a purse). Nevertheless, both biases are visible in the results, especially when observing the increase in the time difference between the two test runs.

	***TapLog* on-screen widgets**	***TapLog* application**
1st run	16.3 %	27.2 %
2nd run	34.1 %	44.6 %

Table 7.1: Average interaction time reduction by the proposed solution with respect to the alternatives used in different performance test set-ups. The *TapLog* application was configured to record binary, text and numerical data with three home-screen widget buttons (1st run, see Figure 6.1a in *Evaluation Methods* chapter) or by opening the application and recording input within it (2nd run).

Some considerations regarding the test procedure should be taken into account while analysing the results. First of all, the interaction time seem quite high for recording a simple input. One of the reasons behind that was the lack of focus on the task among participants. Most of them tended to express their comments about the design of the applications and play around with the smartwatch UI during the test runs. Another issue was the response time of the wearable application itself. For the first group of participants, opening the advanced input screens (*Category*, *VAS scale*, *Numerical Scale*) resulted in 1-3s delays. It was later investigated that the problem was caused by the application's debug functionality and was fixed by simply reinstalling the application using the generated APK[1] file. Additionally, this delay was a source of numerous input errors in the first test group as the users made further screen touches when they could not observe any visual feedback.

[1] Android Application Package - a file format used for installing software on the Android operating system

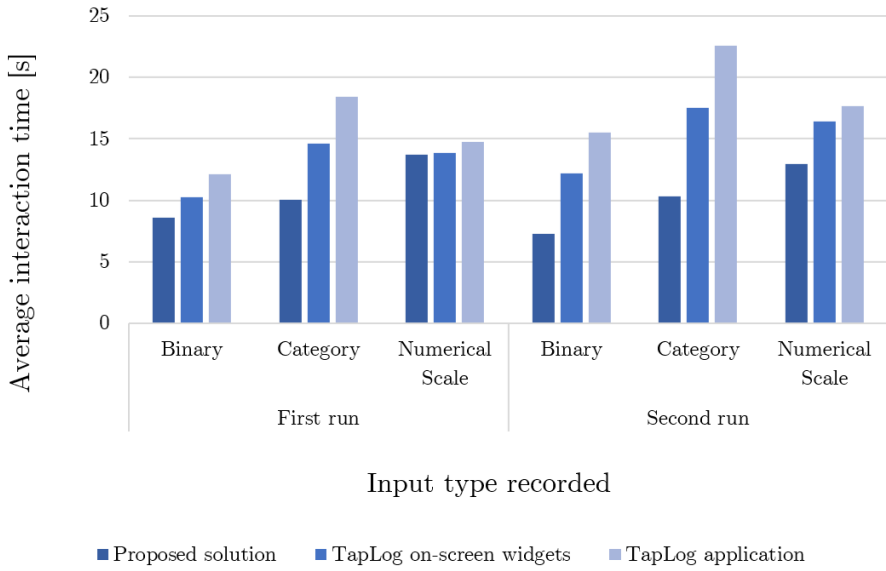


Figure 7.1: Performance test results.

Some observations were also made based both on the interaction times and user behaviour. To begin with, the smartwatch interface proved to be extremely intuitive. Users had issues with the system UI gestures (such as exiting the input screen) before being given some instructions, but capturing the input samples took no exceptional effort. The feedback mechanism based on *Toast* messages and the confirmation buttons seemed similarly appealing. Surprisingly, this was not necessarily a case for the *TapLog* application for smartphone. Although the recording of the sample itself was straightforward, the users lacked explicit confirmation feedback. The application, after saving the record, simply redirected them to the list of recorded samples, even if they registered the input using the widget. Finally, concerns about the *I have a big thumb* rule turned out to be overestimated. The implemented design caused no problems for users that potentially could have been affected.

There was one additional factor measured during the test runs to provide feedback for result analysis. At the beginning of the experiment, the participants were explicitly asked to estimate their level of comfort with respect to modern technology, mainly smartphones and smartwatches, and to record it using the *VAS input* during the performance test. These estimates were supposed to provide contextual data for the actual performance test results (measured in interaction time) and allow the investigation of correlation between both val-

ues. Unfortunately, they yielded no significant results as the recorded estimates seemed to be almost completely random. There are two possible reasons for that. One of them is the lack of stress put on this issue while explaining the purpose of the test. This specific task was incorporated as a part of bigger test run, which caused the users to lose the sense of its significance. The other reason could simply be human psychology. Many users, especially older ones, tended to understate their skills due to personal insecurities.

	Input type		
	Binary	**Category**	**Numerical Scale**
Proposed solution	0.1207	-0.0925	-0.4076
TapLog on-screen widgets	-0.4245	0.1515	-0.0307
TapLog application	-0.4979	-0.3277	-0.3545

Table 7.2: The correlation coefficient between interaction time and estimated comfort with mobile and wearable devices. Subjective user opinion about their skills regarding usage of mobile technology does not directly influence the recorded interaction times.

7.2 Continuous Data Collection

The data collected by Jakob Eg Larsen (JEL), Thomas Blomseth Christiansen (TBC) and the author over a period of few months resulted in creation of three unique data sets and provided a base for exemplary data processing. The exported samples included information about the recorded value, time and location of the capture, the correctness of the input (whether it was marked as a false positive) and the source of the geographical coordinates (the smartwatch or the smartphone). The visualisations and analysis presented further in this section are a proof of concept for possible prototype developments in this direction. The complete set of data consists of 761 samples (109 for JEL, 320 for TBC, 332 for the author) collected between the beginning of January and mid May 2016. Around 3% of them were marked as false positives (unintended recordings).

The first visualisation based on the data sets reflects the distribution of the samples with respect to the month (Figure 7.2) and time of the day (Figure 7.3) they were recorded at. It is clearly visible that the author's compliance with the collection process was decreasing over time. Another interesting observation indicates that TBC started tracking the itching intensively only since the beginning of March.

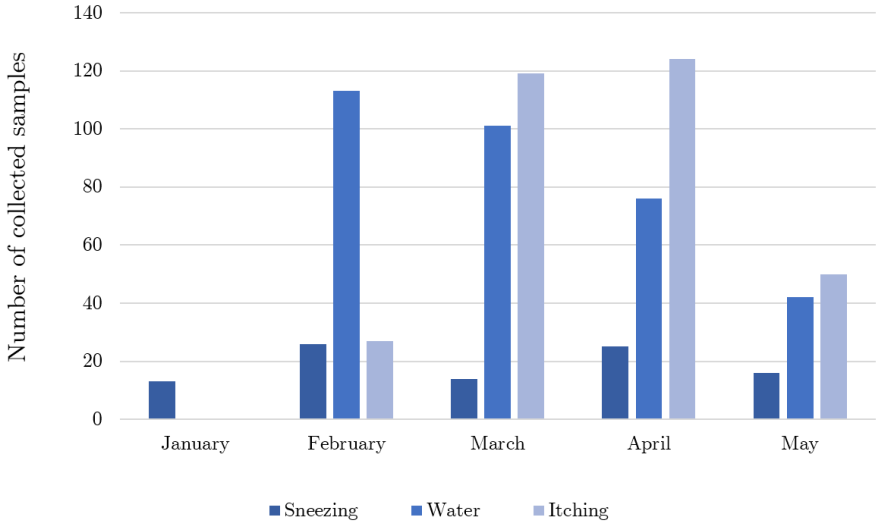


Figure 7.2: Number of data samples collected in respective months.

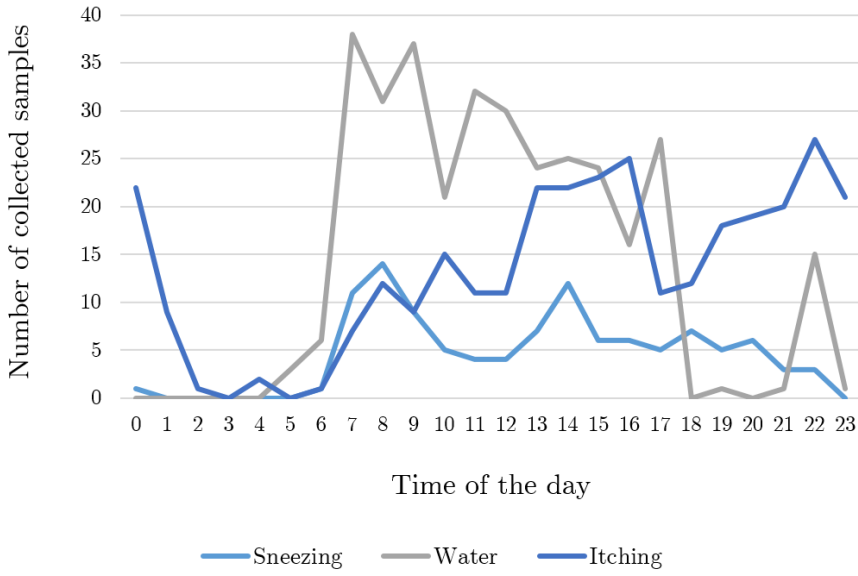


Figure 7.3: Number of data samples collected, with respect to time of the day.

The time-of-the-day visualisation shows more interesting patterns. First of all, although the water consumption data is in general spread quite uniformly during the working hours (which illustrates the fact that the author drinks more water while working), two peaks are noticeable - the morning one, representing the consumption after waking up, and the evening one, standing for water intake during the author's volleyball trainings. Moreover, as far as itching is concerned, the number of samples recorded by TBC tends to increase in the late evening hours of the day. Finally, the sneezing records by JEL present morning (between seven and nine AM) and early afternoon peaks.

The water consumption and sneezing data sets are represented by binary data, however, the itching samples were recorded by TBC using the VAS scale, which allows further analysis in terms of the intensity of the symptom. Looking at the average itching values by months, it is visible that its intensity decreases significantly (by over 60%) between February and May.

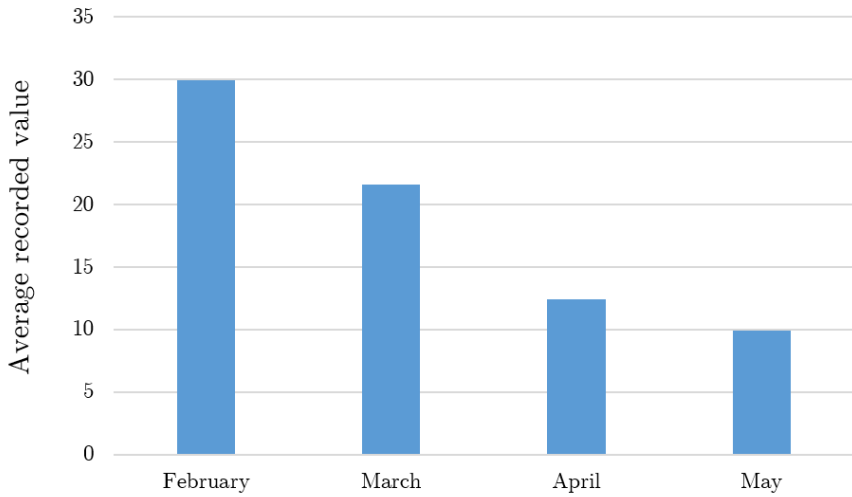


Figure 7.4: The average *itching* values recorded by TBC in respective months. The average values were calculated with all recorded samples, ranging from 0 to 100.

Moreover, the time-of-the-day distribution of the average values shows noticeable peaks in the early morning hours (between four and six in the morning). It is important to mention that the number of records taken during that time is much smaller than in other hours of the day and translates to roughly 2% of all samples. That could indicate that only significant itching was recorded at that time - e.g. one that caused the subject to wake up.

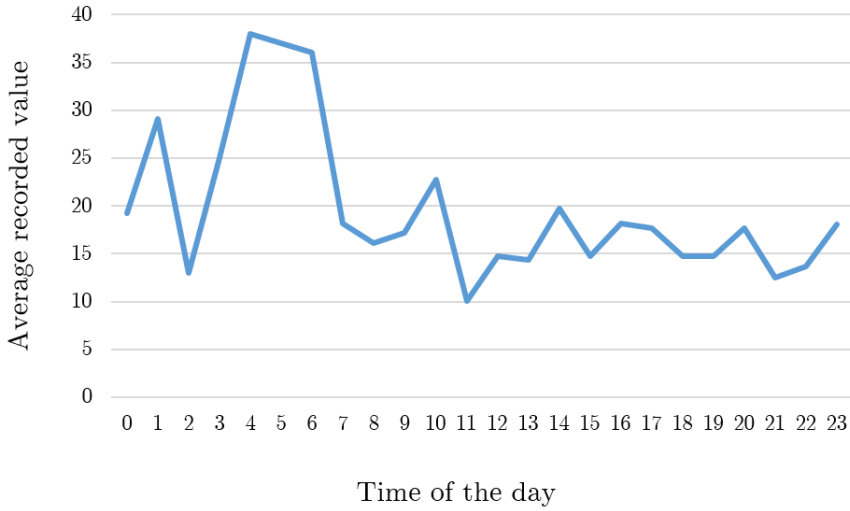
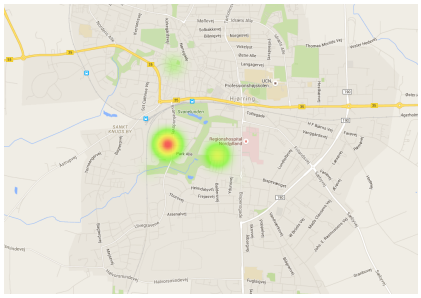


Figure 7.5: The average *itching* values recorded by TBC, with respect to time of the day.

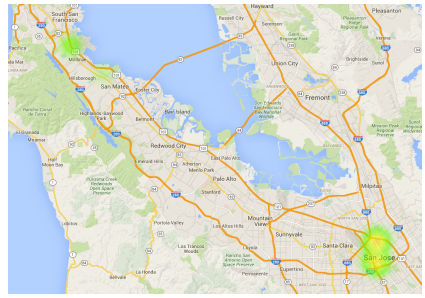
Finally, having the location context data attached to recorded samples, the subjects' location patterns can be analysed. Heat maps presented in Figure 7.6 show the areas of activity of all subjects both during their stay in Denmark, and during their travels. The author's records illustrate his workplace in Trørød, Denmark, two of his living locations (Lyngby, Denmark and Gentofte, Denmark) and the DTU Building 101 (where his volleyball practice takes place). Similarly, JEL's results include mainly his DTU workplace and his home in Hareskovby, Denmark. The heat maps were created using the *Google Maps JavaScript API*[2]. As *Google* provides a similar API to use with the *Android* platform, the presented visualisations can be considered a proof of concept for one potential data display in future versions of the prototype.

Among the collected data some location issues can be observed. The location data can be missing, or inaccurate by 500-2000m for 8-10% of the captured samples. Although the percentage of outliers is quite small, it raises concerns about the location mechanism. The possible sources of this problem have already been discussed in the respective section of *Implementation* chapter and include the lack of available location information or the location update time with respect to the speed of registering input by user. All these reasons increase the need for a more robust location mechanism.

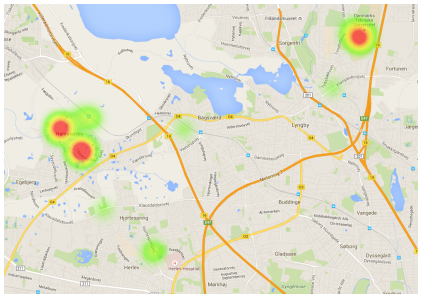
[2]https://developers.google.com/maps/documentation/javascript/



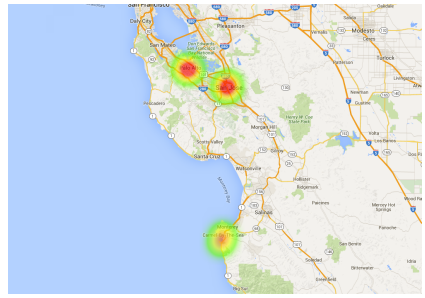
(a) TBC's location patterns in Denmark.



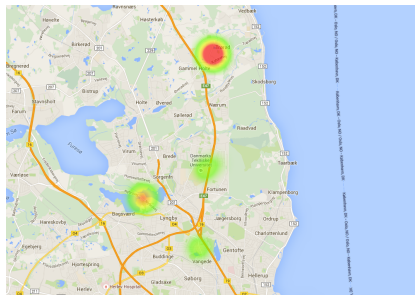
(b) TBC's location patterns during his trip to California.



(c) JEL's location patterns in Denmark.



(d) JEL's location patterns during his trip to California.



(e) The author's location patterns in Denmark.

Figure 7.6: Location patterns identified during continuous data collection, visualised with a heat map. The samples were manually grouped by geographical regions (e.g. the Greater Copenhagen region or state of California) and mapped using *Google Maps JavaScript API*.

7.3 Compliance

One important measure pending extensive testing is the compliance of the proposed solution. Based on the subjective experience of continuous test users (JEL, TBC and the author), it was high. Especially TBC was impressed by wearable application's functionality.

> *"I'd like to tell you that I'm enjoying using the smartwatch UI. I'm tracking something I haven't tracked before: How much the eczema on my hands is itching using a VAS. It works really well. And I've been convinced by the advantages of using smartwatches for this kind of tracking. It's very efficient compared to using a smartphone."*

However, the experience of the test participants is inherently biased in this matter. First of all, as all subjects are involved with the project and are working with *Quantified Self* technology on a daily basis, the opinion is most probably influenced by the urge to actually achieve relevant results. Furthermore, a test group of three people is far from being representative, especially when they represent similar fields, backgrounds and interests. On the other hand, TBC's over 5-year experience in self-tracking with a smartphone cannot be ignored. Positive feedback by a user that worked with competing solutions for that long is a valuable feedback. Nevertheless, a more thorough experiment including specific group of patients from one of the Danish hospitals had been planned, but had to be cancelled due to legal concerns. That experiment could give a much better answer regarding the usability of the prototype. Hence, the compliance evaluation requires additional investigation in the future.

Chapter 8

Future Work

The prototype proposed as a solution in this thesis is a part of an envisioned greater system. Hence, numerous potential points of improvement and extension have been identified via recent user feedback, open discussion and exchange of ideas. Those points include changes to the current implementation and design, improving the interfaces, extending functionality and even providing support for new technologies.

Future work could continue in different directions. The current concept of the application is very general - it allows tracking of arbitrary aspects of everyday routine and provides numerous customisation options. However, it is also possible to choose a different approach - narrowing down and adjusting the functionality for a specific purpose, e.g. medical tracking of symptoms or medication intake. Integration with existing frameworks and providing them with data for analysis could be another option to proceed with.

Another area could involve extending the current input possibilities offered to the users. New wearable devices enter the market at a high frequency, including smartbands, wireless button, finger-attached devices, glasses, even smart clothing or muscle tension sensors. All of them could potentially be utilised for registering user input in some form. The development of the software side of the device also brings new opportunities, e.g. with new touch and wrist gestures being introduced to the system UI. At the same time, porting the application

to the *iOS* or *Pebble OS* platform would increase the reach of the solution.

Considering details, the current prototype itself requires additional work. Many new features, potentially improving the UX, could be introduced. The whole data analysis and representation module has received little attention during the development process and should be extended. The same reflection applies to the compliance support in any form, e.g. a notification system.

Finally, a number of flaws and improvements of current features have been identified and should be addressed.

The following sections discuss those issues in more details, providing general directions and examples for further development. The ideas of high abstraction are discussed first, narrowing it down to current prototype improvements at the end of the chapter.

8.1 Potential Input Types and Methods

Even though the initial response to more sophisticated data types was not very enthusiastic, this possibility should be further investigated. Especially the smartwatch keyboard and voice recognition are worth looking into. *Minuum*[1] is a custom keyboard application still in development and shows impressive promises, while voice recognition could virtually allow to capture any kind of data. The new version of Android Wear (*Android Wear 2.0: Developer preview tour*[2]) is also about to introduce new input options, including the on-screen keyboard. Certainly, the simplicity of use and psychological aspect of both solutions (people might not be convinced to *talk* to their wearable device) will need to verified and taken into consideration.

Another interesting direction of application development would be providing support for different input devices, e.g. Bluetooth buttons. Solutions such as *Flic*[3] or *V.BTTN*[4] offer simplicity of use, impressive portability and, for the latter one, support for gesture recognition. That concept does not end there though as new wearable hardware is being developed rapidly. The *Google Glass* project, despite the initial critical response, still serves a role of a proof of concept for the idea of smart glasses. Recently available forearm band Myo[5]

[1] http://www.minuum.com/
[2] https://www.youtube.com/watch?v=8gLwk8o9LW0
[3] https://www.flic.io/
[4] http://www.vsnmobil.com/products/v-bttn-wearable-bluetooth-le-4-0-device
[5] https://www.myo.com/

already received a lot of positive feedback from both users and press, including *TechCrunch, The New York Times, Mashable, Wearable* and many others. With the development of finger-attached input devices[TY01] and similar ring-like projects like *Ring*[6][Eth14] the possibilities for extension are encouraging.

8.2 Additional features

Among multiple new features that could be implemented in the current version of the prototype, two deserve special attention. Both of them are believed to be important for providing satisfactory user experience and could not be implemented in the current version of prototype only due to scarce time schedule.

8.2.1 Data Processing and Visualisation

With numerous recorded samples, data processing and representation is becoming a pressing issue. The data can be analysed with regards to time and geographical patterns and then displayed as graphs or a map. Provided necessary interactivity, the time-based aggregation graphs could give access to more detailed views, similar to the currently available list view. The possibilities of visualising time series in different forms are endless and should be utilised to provide users with a possibility to reflect on the collected information.

Since data processing could be a resource-consuming process, a *cloud server* solution should be taken into consideration. Such a server could be used not only to analyse the data and provide processed results, but also for the data storage itself. Additionally, it would provide a high-level platform independence. Results gathered from different mobile operating system could be analysed together and then presented to the users as a part of dedicated web application. Such application would be able to offer more sophisticated visualisations and feedback since they are not constrained by the limited screen size.

8.2.2 Reminder mechanism

In order to increase compliance in research, but also to improve the user satisfaction from self-tracking, a reminder mechanism based on system notifications should be introduced. Such notifications, synchronised between the wear-

[6]https://www.kickstarter.com/projects/1761670738/ring-shortcut-everything/posts

able application and the smartphone, should request user activity with respect to recording new samples. Several approaches to the notification scheduling are possible, including, but not limited to, fixed-time scheduling, random-time scheduling and long-inactivity triggers, and should be made available and customisable. Further customisation options would include limiting notifications to specific input groups or time frequencies. Implementing that mechanism, especially the scheduling feature) would actually mean leaning towards the concept of Experience Sampling Method (ESM) instead of Ecological Momentary Assessment (EMA), but should still be discussed in order to improve the usability of the application.

8.3 User Experience Improvements

A couple of UX improvements are also in the agenda for further work. Current shortcomings of the user interface are caused by the urge for simplicity of designs and technical limitations of the smartwatch screen. Some user feedback could potentially improve the UX, but is hard to incorporate into the application without affecting its intuitiveness. Nevertheless, those issues should be discussed and potential alternatives evaluated.

To begin with, more detailed feedback regarding the state and synchronisation of devices would bring additional value. Currently, there is no difference in the mobile application's behaviour depending on whether the smartphone is paired with the smartwatch or not. Users should be informed about a lack of connection to the remote device while using the application. As it would require to be constantly displayed, such an interface would have to take a very discrete form, e.g. an icon with one of the colours from the *traffic light* colour scheme - *red* for the lack of connection, *yellow* for the lack of synchronisation and *green* for full synchronisation, as presented in Figure 8.1.

Similarly, both mobile and wearable application inform the user about synchronisation process when it is started, but there is no validation whether it actually succeeds. Again, the form of such feedback would have to be discussed as it is not desirable to bother the user with consecutive *Toast* messages, nevertheless the information about synchronisation problems would be valuable.

Additionally, there is no need to enforce the chosen colour scheme of the application on all users. The watch face colour, the input icon colours and the colour scheme of the mobile application itself should be customisable. For the latter, the implementation of designs was already concluded (Figure 8.2) and presented to few test users, but no mechanism for changing the schemes was introduced.

(a) Watch face - central circle with alternating colours, dependent on the synchronisation state.

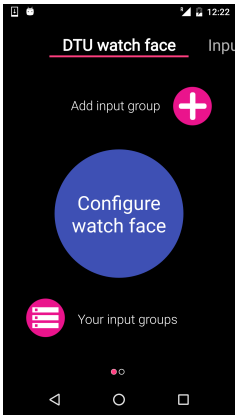
(b) Companion application - synchronisation message on the watch face configuration screen.

Figure 8.1: Potential designs for the synchronisation feedback.

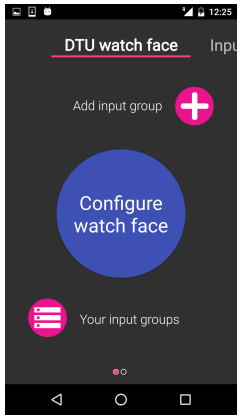
8.4 Late User Feedback

As mentioned in the *Method* chapter, qualitative feedback regarding final version of the prototype was collected during the performance testing. A few new design considerations were suggested, starting with the need for icon position customisation. Users prefer more control over positioning of the selected input group - either via selection of the target position or by the drag-and-drop mechanism. A suggestion to use the watch face preview in the main menu, instead of the *Configure watch face* button was also made.

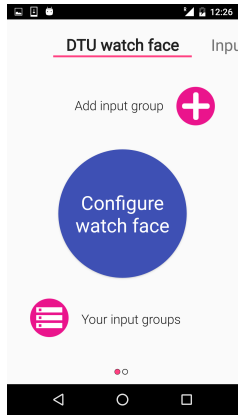
Another group of comments concerned the lack of user guidance in the application. The explanation of parameters necessary to create an input group, the different input types (possibly including graphical examples) and available system UI gestures should be provided. Lack of that feature is acceptable for the current version of the prototype, but not for the release of the application. The form of such instruction mechanism would have to be determined, but possible ideas include e.g. manual screens displayed at the first launch of each application screen.



(a) Custom color scheme



(b) Dark color scheme according to Android design guidelines



(c) Light color scheme according to Android design guidelines

Figure 8.2: Home screen of the mobile application with various implemented colour schemes.

8.5 Bug Fixing

Apart from greater changes, the application contains several small bugs that were encountered in the late phase of testing and could not have been fixed as a part of this project.

- The *Tissue* and *Weight* icon images are not displayed in the *GridView* on the input group form. Instead, they are substituted by duplicated *Coffee* and *Biking* icons.
- After entering the watch face configuration screen and leaving the application, the background of the *Configure watch face* button is changed from *blue* to *orange*. The background turns *blue* again after leaving the application once again.
- Defining two identical input groups is possible, causing potential confusion during the configuration and data viewing process. In such case, both groups would be available for configuration, but the system would assign the recorded samples only to the original one.

Although the bugs seem trivial, the origins of the first two are unknown and would require extensive debugging. All of these should be fixed before any application release is scheduled.

The location issues, mentioned in the *Results and Discussion* chapter, are more pressing. Only after weeks of testing, it became obvious that there are still some inaccuracies in the mechanism. Both minor (500-2000m difference) and major (not registering the change of location at all) occurred for about 10% of samples. Additionally, thorough and time-consuming testing is necessary to identify the reasons for deviations. Potentially, debugging modifications could be introduced in the application, responsible for capturing the times and results of location updates and user interaction. Those data could be compared with the data from other location tracking application, such as *Moves*[7]. Using this method, different approaches, such as waiting for the actual location update, should be implemented and tested. Also, *null* coordinates are still present in the collected data, indicating that the fallback solution (adding location information in the mobile application's data listener service) does not completely solve the problem. The service should actually request location updates itself and not rely on incoming data.

[7]https://www.moves-app.com/

Chapter 9

Conclusion

In this thesis the development process of a wearable system for acquiring data on subjective experiences is described.

It explained the concept of experience sampling and its importance for health care, personal well-being and productivity improvements. It introduced examples of two methods intending to utilise it for research purposes (Experience Sampling Method and Ecological Momentary Assessment) and pointed out the motivation behind this project. The main problem discussed is the user effort necessary for active *self-tracking* and its correlation to poor compliance. Although contemporary mobile devices contributed to the notion of *lifelogging*, the manual data collection has received far less attention than e.g. automated sensors.

The thesis described the research related to the experience sampling, both with respect to the general concept and how it changed with evolution of mobile technology, including mobile and wearable devices. It also outlined the theory behind the ESM and EMA, and how it corresponds to contemporary *Personal Informatics* domain, including its stages and barriers. The conceptual model of solution to reducing these barriers was presented as a combination of wearable and mobile applications with a common server side for data storage and processing. The modularity of the idea allowed the implementation of both applications as a part of this thesis, with a possible server extension in the future.

Numerous design considerations were mentioned and taken into account, including limitations of the devices, official design guides and potential special needs of the users. The design process itself included feature-based iterations with informal evaluations and continuous feedback. The thesis presents the overview of the system, different design stages, possible alternatives, and user feedback with respect to the wearable application and the companion application separately. The description of issues related to implementation of those designs points out encountered problems, evaluated solutions, non-trivial decisions and component diagrams for different modules of the system.

Details regarding the evaluation process itself were also described, including informal design evaluation (*quick-and-dirty* and *thinking aloud* methods), continuous testing and data collection, and finally, the performance test, facilitated by a dedicated test application, and conducted on a test group of various demographic backgrounds. The collected results indicated significant user interaction time reduction in favour of the proposed solution, while captured data itself revealed interesting patterns with respect to location and time-wise behaviour of the participants. Nevertheless, a critical discussion regarding all the findings is also documented.

Finally, the thesis discussed possible directions for further development, both as high level concepts (generalising the solution by extending input types and methods or narrowing down the functionality to specific field) and as improvements and fixes for the current version of the prototype.

To summarise, this project contributes to the fields of experience sampling and momentary assessment by

- a critical discussion of state-of-the-art ESM and EMA and current development in both fields with respect to modern technologies, especially mobile devices;
- investigating the wearable devices as a possible approach to reducing the user barrier to manual collection of data;
- developing a wearable system for said data collection, consisting of *Android Wear* application synchronised with companion mobile application;
- critical evaluation of the developed software as far as the interaction time and effort necessary to acquire the data is concerned;
- and finally, analysing the captured data in context of automatically recorded user location information to illustrate the potential of processing collected data.

Appendix A

Performance Test Results

The results of the performance test conducted on eleven subjects, with two test runs per subject. The collected data include the date of the test, subject's identifier, interaction times in milliseconds, age, number of years of university education, educational background and sex.

Create Date	Name	Wearable application				TapLog widgets			TapLog application			Age	Years of higher education	Background	Sex
		Binary	Category	VAS	Numeric scale	Binary	Category	Numeric scale	Binary	Category	Numeric scale				
2016-04-22 09:34	Harmean	5146	6645	8478	10350	7559	14619	11679	10642	17525	11984	25	4.5	Engineering	M
2016-04-22 09:45	Harmean	7366	11143	12270	21322	9051	21124	12106	16343	26274	18518				
2016-04-22 09:51	Kartoffelsalat	13189	14345	17130	14916	12292	13965	15049	11391	12835	9435	24	6	Business	M
2016-04-22 09:55	Kartoffelsalat	10176	12277	13613	11985	13336	18806	16850	15520	21022	19834				
2016-04-22 10:08	1st Lady	18552	21267	40652	14856	13024	21489	19342	14193	20500	21906	25	5.5	Engineering	M
2016-04-22 10:12	1st Lady	5004	13863	19277	15428	12232	21470	20698	16956	34020	17716				
2016-04-22 10:25	Mr. Pool	5338	610	6369	18561	9546	21637	16556	9319	18482	830	29	10	Engineering	M
2016-04-22 10:29	Mr. Pool	8502	6416	11441	16363	13851	16013	18999	12538	26922	20251				
2016-05-10 12:42	Marek	8108	9639	12034	12780	8269	12024	12603	10773	17784	12695	24	4.5	IT	M
2016-05-10 12:47	Marek	6733	7368	11371	7887	10334	15603	14802	13513	18195	18806				
2016-05-10 13:14	Nana	6589	14187	13970	15129	13192	13187	14185	14046	18486	18776	53	2	Business	F
2016-05-10 13:18	Nana	7795	11131	14802	15204	11268	13849	16836	16447	20875	21162				
2016-05-10 13:35	Vicky	5478	7369	10127	11341	8964	12795	14772	12362	15032	16766	28	6	Business	F
2016-05-10 13:39	Vicky	4829	7040	8171	7751	11794	16269	15831	14745	19152	16844				
2016-05-14 17:49	Drizzt	6668	8879	9702	8150	8666	11861	9119	12099	18232	14764	26	5.5	IT	M
2016-05-14 17:56	Drizzt	5684	9193	9483	8279	12394	17936	13821	13348	15388	4487				
2016-05-14 18:21	Ironman	5641	8463	8766	8330	7472	11586	9222	9891	13285	15421	26	7	IT	M
2016-05-14 18:26	Ironman	4975	6920	9320	8599	9999	14426	12481	14820	17394	14688				
2016-05-15 10:51	ND	8572	10307	16097	19382	11102	13904	10997	15022	24455	19701	56	5	Engineering	M
2016-05-15 10:58	ND	8604	14674	16495	13220	13857	18030	20069	18305	22680	15724				
2016-05-15 11:19	Aga	11063	8872	13603	16929	12941	14013	18924	13430	25859	19696	49	6	Engineering	F
2016-05-15 11:29	Aga	10236	13777	20251	16504	16132	18994	18016	17875	26698	26062				

Bibliography

[BB01] Lisa Feldman Barrett and Daniel J Barrett. An introduction to computerized experience sampling in psychology. *Social Science Computer Review*, 19(2):175–185, 2001.

[Bin03] Philip F Binkley. Predicting the potential of wearable technology. *Engineering in Medicine and Biology Magazine, IEEE*, 22(3):23–27, 2003.

[Bra69] Norman M Bradburn. The structure of psychological well-being. 1969.

[BSG01] Polly E Bijur, Wendy Silver, and E John Gallagher. Reliability of the visual analog scale for measurement of acute pain. *Academic emergency medicine*, 8(12):1153–1157, 2001.

[CBBM+03] Tamlin Conner Christensen, Lisa Feldman Barrett, Eliza Bliss-Moreau, Kirsten Lebo, and Cynthia Kaschub. A practical guide to experience-sampling procedures. *Journal of Happiness Studies*, 4(1):53–78, 2003.

[CL14] Mihaly Csikszentmihalyi and Reed Larson. Validity and reliability of the experience-sampling method. In *Flow and the Foundations of Positive Psychology*, pages 35–54. Springer, 2014.

[CLL+14] Eun Kyoung Choe, Nicole B Lee, Bongshin Lee, Wanda Pratt, and Julie A Kientz. Understanding quantified-selfers' practices in collecting and exploring personal data. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 1143–1152. ACM, 2014.

[DGF+95] HE Drummond, S Ghosh, A Ferguson, D Brackenridge, and B Tiplady. Electronic quality of life questionnaires: a comparison of pen-based electronic questionnaires with conventional paper in a gastrointestinal study. *Quality of life research*, 4(1):21–26, 1995.

[DLC14] Sean T Doherty, Christopher J Lemieux, and Culum Canally. Tracking human activity and well-being in natural environments using wearable sensors and experience sampling. *Social Science & Medicine*, 106:83–92, 2014.

[ECB+14] Daniel Epstein, Felicia Cordeiro, Elizabeth Bales, James Fogarty, and Sean Munson. Taming data complexity in lifelogs: exploring visual cuts of personal informatics data. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 667–676. ACM, 2014.

[EM76] Norman S Endler and David Magnusson. Toward an interactional psychology of personality. *Psychological bulletin*, 83(5):956, 1976.

[Eth14] Darrell Etherington. The ring input device puts gesture control and home automation on your finger, 2014.

[FCC+07] Jon Froehlich, Mike Y Chen, Sunny Consolvo, Beverly Harrison, and James A Landay. Myexperience: a system for in situ tracing and capturing of user feedback on mobile phones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 57–70. ACM, 2007.

[FCLD12] Yingling Fan, Qian Chen, Chen-Fu Liao, and Frank Douma. Smartphone-based travel experience sampling and behavior intervention among young adults. 2012.

[Gho06] Agamoni Ghosh. Sony smartwatch 3: Android marshmallow update rolling out to users, 2016-04-06.

[HLD+08] Gary Hsieh, Ian Li, Anind Dey, Jodi Forlizzi, and Scott E Hudson. Using visualizations to increase compliance in experience sampling. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 164–167. ACM, 2008.

[HRK+10] John Hicks, Nithya Ramanathan, Donnie Kim, Mohamad Monibi, Joshua Selsky, Mark Hansen, and Deborah Estrin. Andwellness: an open mobile system for activity and experience sampling. In *Wireless Health 2010*, pages 34–43. ACM, 2010.

[HS93] J Gregory Hixon and William B Swann. When does introspection bear fruit? self-reflection, self-insight, and interpersonal choices. *Journal of personality and social psychology*, 64(1):35, 1993.

[HSC07] Joel M Hektner, Jennifer A Schmidt, and Mihaly Csikszentmihalyi. *Experience sampling method: Measuring the quality of everyday life*. Sage, 2007.

[IJ02] Remus Ilies and Timothy A Judge. Understanding the dynamic relationships among personality, mood, and job satisfaction: A field experience sampling study. *Organizational behavior and human decision processes*, 89(2):1119–1139, 2002.

[LDF10] Ian Li, Anind Dey, and Jodi Forlizzi. A stage-based model of personal informatics systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 557–566. ACM, 2010.

[LRB03] George Loewenstein, Daniel Read, and Roy F Baumeister. *Time and Decision: Economic and Psychological Perspectives of Intertemporal Choice: Economic and Psychological Perspectives of Intertemporal Choice*. Russell Sage Foundation, 2003.

[MJK+12] Megan A Moreno, Lauren Jelenchick, Rosalind Koff, Jens Eikoff, Cheryl Diermyer, and Dimitri A Christakis. Internet use and multitasking among older adolescents: An experience sampling approach. *Computers in Human Behavior*, 28(4):1097–1102, 2012.

[Pat30] Sarah Patrick. Early development challenges for wearables prompt creative problem solving, 2015-09-30.

[RSB+13] Jason D Runyan, Timothy A Steenbergh, Charles Bainbridge, Douglas A Daugherty, Lorne Oke, and Brian N Fry. A smartphone ecological momentary assessment/intervention "app" for collecting real-time data and promoting self-awareness. *PloS one*, 8(8):e71325, 2013.

[SD77] Clive Seligman and John M Darley. Feedback as a means of decreasing residential energy consumption. *Journal of Applied Psychology*, 62(4):363, 1977.

[SHH+97] Saul Shiffman, Michael Hufford, Mary Hickcox, Jean A Paty, Maryann Gnys, and Jon D Kassel. Remember that? a comparison of real-time versus retrospective recall of smoking lapses. *Journal of consulting and clinical psychology*, 65(2):292, 1997.

[SPD09] Christie Napa Scollon, Chu-Kim Prieto, and Ed Diener. Experience sampling: promises and pitfalls, strengths and weaknesses. In *Assessing well-being*, pages 157–180. Springer, 2009.

[SPG+96] Saul Shiffman, Jean A Paty, Maryann Gnys, Jon A Kassel, and Mary Hickcox. First lapses to smoking: within-subjects analysis of real-time reports. *Journal of consulting and clinical psychology*, 64(2):366, 1996.

[SS94] Arthur A Stone and Saul Shiffman. Ecological momentary assessment (ema) in behavorial medicine. *Annals of Behavioral Medicine*, 1994.

[SS03] Joshua M Smyth and Arthur A Stone. Ecological momentary assessment research in behavioral medicine. *Journal of Happiness studies*, 4(1):35–52, 2003.

[SSH08] Saul Shiffman, Arthur A Stone, and Michael R Hufford. Ecological momentary assessment. *Annu. Rev. Clin. Psychol.*, 4:1–32, 2008.

[SSS+03] Arthur A Stone, Saul Shiffman, Joseph E Schwartz, Joan E Broderick, and Michael R Hufford. Patient compliance with paper and electronic diaries. *Controlled clinical trials*, 24(2):182–199, 2003.

[STC+00] Joel D Swendsen, Howard Tennen, Margaret Anne Carney, Glenn Affleck, Amy Willard, and Amber Hromi. Mood and alcohol consumption: an experience sampling test of the self-medication hypothesis. *Journal of abnormal psychology*, 109(2):198, 2000.

[TY01] Koji Tsukadaa and Michiaki Yasumurab. Ubi-finger: Gesture input device for mobile use. In *Ubicomp 2001 Informal Companion Proceedings*, page 11, 2001.

[VW00] Luuk Van Waes. Thinking aloud as a method for testing the usability of websites: the influence of task variation on the evaluation of hypertext. *Professional Communication, IEEE Transactions on*, 43(3):279–291, 2000.

[VWC+14] Rajan Vaish, Keith Wyngarden, Jingshu Chen, Brandon Cheung, and Michael S Bernstein. Twitch crowdsourcing: crowd contributions in short bursts of time. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 3645–3654. ACM, 2014.

[Wel15] Chris Welch. Android wear watch faces are now interactive and way more useful, 2015.

[Xan15] Dimitris Xanthos. Medopad. *The Lancet Oncology*, 16(8):893, 2015.

[Z+80] Jiri Zuzanek et al. Work and leisure in the soviet union. a time-budget analysis. *Work and leisure in the Soviet Union. A time-budget analysis.*, 1980.