VRIJE UNIVERSITEIT BRUSSEL

Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Making Wearables Fit the User

A Trigger-Action Programming Approach to Configure Wearables

NICOLAS CARRAGGI
August 2017

Promoter: Prof. Dr. Beat Signer
Advisor: Sandra Trullemans
Faculty of Science and Bio-Engineering Sciences

VRIJE UNIVERSITEIT BRUSSEL

Proefschrift ingediend met het oog op het behalen van de graad van Master of Science in de Ingenieurswetenschappen: Computerwetenschappen

Making Wearables Fit the User

A Trigger-Action Programming Approach to Configure Wearables

NICOLAS CARRAGGI

Augustus 2017

Promotor: Prof. Dr. Beat Signer
Begeleider: Sandra Trullemans
Faculteit Wetenschappen en Bio-ingenieurswetenschappen

Abstract

Over the last decades technology evolved from personal computers with limited computing power to mobile computers, such as smartphones and notebooks, that are small and extremely powerful. Nowadays technology is so tiny that it can be embedded in everyday objects, so-called smart devices or things. One type of smart devices, labelled wearables, can be worn on the user's body. For example smartwatches, which have recently gained popularity for fitness and health purposes. Big manufacturers like Google, Apple, Samsung and Fitbit sell various kinds of smartwatches and are exploring other types of wearables. Their small form factor leads to a challenging task of offering interesting functionality in a user-friendly way. The functionality proposed by manufacturers of wearables often does not cover all the possible usages of these devices. To utilise wearables to their full capacity, we present a framework that empowers the end user to configure and personalise their wearables.

Through a literature study we identify the current state of wearables and explore how they can be used for novel interactions. Furthermore, we classify and analyse end-user development techniques. Professional programmers are overloaded with an increasing amount of specific software needs with domain-specific requirements. To overcome this problem end-user development gives the ability to the end user to make their own programs. The end user can, for example, create little programs in the form of if-then rules.

With the Triggered framework, we introduce a new approach on how to use wearables and open up new functionality for wearables in a more personal manner. This framework allows users to configure their wearables by creating rules, such as "*When I start working out in the gym, set my smartwatch in sport mode and start a timer*" or "*When I press button X of my smartwatch while I am at home, turn the TV on or off*".

In order to find the best balance between ease of use, learning cost and expressiveness we propose three different configuration methods for creating rules with our framework. Based on an evaluation and analysis of those configuration methods we define three design implications to enhance our framework. With Triggered we provide a solid, user-friendly and promising framework that enables end users to configure their devices and wearables.

Acknowledgements

First of all, I would like to thank my advisor Sandra Trullemans and promoter Prof. Dr. Beat Signer. They guided me throughout the whole process of choosing a relevant topic idea, executing it and writing this thesis. Thank you for your time and explaining me over and over again how I could improve my work.

Next, I would like to thank Lars Van Holsbeeke for supporting me, giving me great tips, proofreading my thesis and doing a pilot study for the evaluation. Also, I would like to thank all the persons who freed up some time to participate in the evaluation: Kevin Sterckx, Olav Keijzer, Jonathan Van Den Driesch, Thomas Deckers, Nick Marcoen, Maude Van Gyseghem, Manon Vervaeke, Julien Warnotte, Maxime Evrard, Amaury Evrard, Carlo Carraggi, Nico Esposito, Aline Bouton, Guillaume Veldekens and Nicolas Mancini.

Finally, I would like to thank my parents for everything they did for me, giving me the opportunity to go to university and believing in me. Thanks to my brothers Carlo and Julien, my family and my close friends for being there for me and motivating me.

Contents

Chapter 1

Introduction

The immense growth in technology the past decade made us realise that technology has become a huge part of our daily life. It has become so important that the idea of teaching kids very early about technology and even programming is becoming a reality. We learned to think more logically, when we perform some kind of action we know that something will react in one way or another. This understanding of logic rules can be applied to new technology by the end user. The end user can configure and use new technology in a personalised and unique way. For example, the end user can configure their smart home to heat the living room to an ideal temperature when the user comes back home from work.

A new kind of technology which is getting a lot of attention in the scientific and consumer world is wearable technology. Wearable devices can be worn by the user, which allows for new ways of interaction with the user and makes it possible to retrieve data about (or from around) the user. These devices are in direct contact with the human body and require little to zero physical effort from the user. Big companies such as Apple[1], Google[2] and Samsung[3] have already successfully released wearables for customers. Unfortunately, most of these wearable devices are restricted to fulfil one or few purposes. The end user cannot personalise, for example by means of a configuration, how the wearable works.

In this chapter we explain wearables and end-user programming in more detail, introduce the contribution of this master's thesis and finally give an overview of the structure of the thesis.

[1] `https://www.apple.com` [Accessed 22 July 2017]
[2] `https://www.google.com` [Accessed 22 July 2017]
[3] `https://www.samsung.com` [Accessed 22 July 2017]

1.1 Wearables

A long time before the emergence of wearables Mark Weiser presented his vision of the future of computing, called Ubiquitous Computing [61]. Weiser explains that once we learn something, in this case computer technology, we would cease to be aware of it. The computers will be there but they will vanish into the background. Weiser presents a wired and wireless network in which computers, tabs, pads and printers are interconnected and share programs and data. Ubiquitous computers will make information retrieval trivial and will help overcome information overload. Our lives and habits are indeed affected by the fact that we now have access to computing technology almost everywhere and at any time.

Over the years a lot of wired and wireless technologies have been invented and improved. The most common wireless communication technologies are Bluetooth[4], Wi-Fi[5] and 4G. These are serving different purposes but are very important in today's life. Bluetooth and Wi-Fi enable direct short-distance wireless communications between devices. However, Wi-Fi connects a device to a network by means of a router on which other devices can connect as well. Finally, 4G is a long-distance wireless mobile telecommunications technology which connects a device to a network (through a cell phone tower). The largest public network is the Internet. Most networks are connected to the Internet, which means that most computers in the whole world are interconnected.

The combination of the Internet with the fact that computers are shrinking and getting cheaper is leading to the Internet of Things (IoT) [22]. This principle implies that not only computers, but also things are connected through the Internet. Nowadays, there exists computers having the size of a penny, called things, that contain all kinds of sensors that can capture data and share it via the Internet. Other things can react to specific information that is received from the Internet. The most important consequence of the Internet of Things is that our environment is turning into a smart environment. Gubbi et al. [22] define IoT for smart environments as follows:

Definition 1.1.1 (Internet of Things)**.** *Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless large scale sensing, data analytics and information representation using cutting edge ubiquitous sensing and cloud computing.*

The term *cloud computing* refers to a paradigm that promises data storage and processing *in the air*, but in reality the data is stored and processed in

[4] `https://www.bluetooth.com` [Accessed 22 July 2017]

[5] `http://www.wi-fi.org` [Accessed 22 July 2017]

data centres that are connected to the Internet. We can resume that the IoT enables innovative applications, which gradually make our environment and life smarter.

With this understanding of the IoT, we can now look at wearables in particular. Wearables are obviously one category of things in the whole spectrum of IoT. They exist in different forms. The most popular wearables right now are activity tracking wristbands, smartwatches and smartglasses. Samsung and Apple have chosen to create innovative smartwatches that encapsulate multiple sensors, are lightweight and have a beautiful screen. Google chose to develop smartglasses, which also have a multitude of sensors and a little screen in the corner of the user's eyesight. The possibilities of smart clothing are being explored as well, for example clothing that can sense a wide range of characteristics of the users body. Smart clothing can be used for fashion by having the ability of dynamically change what is displayed on the clothing itself [15].

For now, consumer wearables are mainly used for health and fitness purposes and information retrieval in the form of notifications. Living in a time where fitness is seen as very important by the society, consumer wearables are used to track activities such as walking, running, cycling, weightlifting or swimming. This type of wearables work as persuasive technology [19], for example, people acquire fitness wearables to surpass their own results or records. The fact that fitness data can be shared on social media, pushes the user to achieve good results which after the effort can be shared and compared with friends. Thanks to this type of wearables, users can be more aware of their fitness level. People with certain health issues can use wearables in a more informative way, for example in the form of a wearable that informs the user about blood pressure and sugar level.

A major part of the actual use of wearables is notifications retrieval. This again confirms Weisers prediction [61] of a more natural flow of information to the user. A small company called Pebble[6] gained a lot of popularity with their smartwatches that were focussed on notifications. Pebble users get notified via their smartwatch about incoming calls, messages or social media. This gives the user the opportunity of having a quick glance at the smartwatch to know what the notification is about. The user can then decide to ignore and/or dismiss the notification or dive more into the content of the notification on the smartwatch itself or another device.

Although there are wearables that target different goals, these are in the minority. A lot of research has been done to find new usage and ways of interaction with wearables, which is discussed later on in Chapter 2.

[6] `https://www.pebble.com` [Accessed 22 July 2017]

1.2 End-User Development

One of the main goals in Ubiquitous Computing is to achieve autonomous applications that require a few or no user interactions. These applications are capable to adapt their behaviour according to the environment, actions or requirements of the user and are generally called context-aware systems. However, Hardian [23] remarked that users are not ready for fully automated applications as they want to have some control over them. Thus a balance between user control and system autonomy has to be found. Giving control to the user can be achieved by applying end-user programming techniques.

With the rise of Ubiquitous Computing, programming skills are very important to make these devices useful and intuitive for the end user. Learning to program is a very demanding and time-consuming task. Only a small portion of end users have the ability to write programs, but a huge amount of them have ideas for applications that could be created or personalised to improve their lives. End-user programming (EUP), or more generally end-user development (EUD), is giving the end user the possibility to create, configure or personalise computers without needing professional programming experience [45, 51].

A well-known and established EUD paradigm that is applied in Microsoft Excel[7], is called the Spreadsheet Paradigm [1]. A spreadsheet is composed of cells that contain values, references or formulas. This tool gives the ability for any end user to create programs to a certain extent. A user could, for example, use Excel to keep an overview of all expenses paid. Thanks to the concept of formulas, which is derived from mathematics, the user can apply more complex calculations to the data in an easy and efficient way.

Other types of end-user programming have been explored over the years. Visual programming is based on graphically representing program elements such that the end user can easily understand and use these components to make a program. This type of end-user programming is very successful on the Web, using a *You See Is What You Get* (WYSIWYG) approach the end user can build a website. Take, for example, the online web development company Wix[8]. Their online tool allows the end user to drag and drop visual elements to build web pages. Wix also provides templates, in case the user needs inspiration or does not want to start from scratch.

In traditional programming the If-Then(-Else) statement is very important. Based on this statement, trigger-action programming, or more generally rule-based programming, allows the end user to define what triggers a rule and what should happen when the rule is triggered. For example, the commercial online

[7] `https://products.office.com/en/excel` [Accessed 22 July 2017]
[8] `https://www.wix.com` [Accessed 22 July 2017]

service If-This-Then-That[9] (IFTTT) lets the user create or re-use If-Then rules. This way IFTTT users can make links between services and/or devices and create their own mashups.

1.3 Methodology and Contribution

An important characteristic of wearables is their small size. This means that the makers of wearables have to be creative and think of interesting applications and ways of interacting with the user. Wearable devices are not yet used to their full potential. They are still in the early stages of adoption, but the majority of end users do not see the utility of wearables. At the moment wearables have limited capabilities. The maker of the wearable device decides which functionality the wearable offers. The end user is limited to use the wearable as it is intended by the maker, which leads to a lack of interest in the wearable.

The idea that drives the research in this thesis is the exploration of a new approach for using and configuring wearables. Instead of seeing the end user only as the user of the wearable, we think the end user should also be the programmer of the wearable. By combining wearable technology with the knowledge that we already acquired in end-user programming, end-user creativity can light up all the possible uses of wearables. By proposing an interface which conforms to trigger-action programming techniques, the end user can configure and personalise wearables. In order to fully understand this approach, we discuss a potential use case.

Nick is a student and badminton player. He would like that his smartwatch behaves differently according to his location, his activity or a specific moment in a day. On a typical day Nick wakes up, takes his bike to the badminton hall, has a training session, then rides back to his house, walks to the tram stop, takes the tram to school, attends to classes, takes the tram back to his house and by the end of the day goes to sleep. We can classify this information into different locations, activities, moments:

- Location: *house, badminton hall, school*
- Activity: *riding the bike, walking, taking the tram, training, sleeping*
- Moment: *waking up, going to sleep*

When Nick wakes up, he would like to be informed about the weather and see an inspiring quote on his smartwatch. When Nick is at home, one button on his smartwatch should open and close his garage door and another button turns

[9] `https://www.ifttt.com` [Accessed 22 July 2017]

the radio on or off. When he is riding his bike, he wants to able to see his speed on his smartwatch. During his training, Nick would like to be alerted if he exceeds a certain heart rate. While walking to the tram stop, Nick wants to know when the next tram arrives. During classes Nick wants to see when and which is the next class. Finally, when going to sleep, Nick can turn of the lights with a gesture. All these requirements are conditional and can be mapped to If(-While)-Then rules. Nick can now configure his smartwatch with those rules using our proposed framework.

We start by exploring the current state of end-user development and wearables in the form of a literature study. A lot of research has already been done on end-user development, we give an overview and examples of the different types of EUD. On the other hand, we explore a more recent topic called Wearables. We discuss what wearables are, how they are designed, how they are currently used and finally how they could be used in the future.

We present three possible ways of configuring wearables using rules. The first one is a *closed configuration* which will propose predefined rules to the user. The user can choose a rule and eventually adjust values to fit their situation. The second configuration is a *half-open configuration interface*, which means the end user is guided with templates, where the user can choose which devices are used in the rule and more specific details. The last possible way is a fully *open configuration* where the end user has complete freedom and builds the rules from the ground up.

We implement these configuration methods in a framework called Triggered. With this framework the end user can effectively create rules to combine functionality from devices and wearables. Using Triggered, we evaluate these three configuration methods in order to analyse how end users experience them. Finally, from these results we propose three design implications that can be applied on future designs.

We can resume our contribution as follows:

- A literature study on wearables and end-user development
- The Triggered framework for configuring wearables
- An evaluation and statistical analysis of the three configuration methods
- Three design implications for our framework

1.4 Structure

With this brief introduction we explained the actual lack of control that the end user has over wearables and how this could be improved by giving the ability to the user by means of an intuitive trigger-action configuration interface.

This thesis continues by discussing research that has already been done on wearables and end-user programming in Chapter 2. Chapter 3 gives a detailed overview of the different types of configuration that are possible for the end user and how our framework is created. Afterwards, the methodology of our research is explained in Chapter 4. Chapter 5 analyses the results of the evaluation. Finally, we conclude this research and discuss what still can be done in the future.

Chapter 2

Related Works

In this chapter we discuss in more detail existing research. Wearable technology is an innovative technology that brings a lot of improvements to the table, but also demands changes in the user's habits. What are wearables exactly good for? How should they be designed to effectively be useful for any end user? In addition, all the sensors generate enormous streams of data that can contribute to smart environments. Possibilities for giving more control to the end user over computers and programs has been explored for a very long time. End-user development has developed to a major aspect in computing. That is why we will investigate multiple EUD approaches that have been explored and have achieved significant results.

2.1 Wearables

The idea of wearing tiny computers on one's body is not new. Decades ago, some wrist-worn devices have been developed using available technology from back then. Unfortunately those were ahead of their time, too expensive, cumbersome and not pleasant for the eye. But the idea luckily stayed alive and became reality thanks to the advances in technology. Now that the required hardware is a lot smaller and cheaper, all sorts of wearables can be made [54]. Depending on the included input sensors and output, various functionality can be proposed. Smartwatches are already well-known thanks to a significant number of products on the consumer market. But also smart glasses, smart jewellery and all kinds of smart clothing can be worn. As with every new technology, wearables have numerous advantages and constraints as well. An obvious advantage is the form factor. For example, compared to a smartphone, it is less obtrusive and less cumbersome. But more important advantages are the mount location and the continuous connection to the skin. Constraints of wearables are mainly related to their size. In order to be small enough to be worn, wearables have to opt for tiny screens and a limited amount of buttons or touch input.

More generally, wearables often have restricted input and output possibilities. Other logical consequences are the limited computing power and battery capacity. With all these constraints in mind, it is very important to keep wearable applications simple and energy efficient. To resolve the lack of input and output capabilities, new interactions are being investigated in research. Depending on the type of wearable and location on the body, gestures and the human skin are used to interact with the wearable. These will be explained in more detail later on in this chapter.

Wearables have specific hardware and needs, thus the importance of a good Operating System (OS) is vital [14]. A wearable operating system should be specially designed to assure an efficient process scheduling, efficient energy management and other wearable-specific aspects. Depending on the focus of a wearable, the OS can be different. Wearables with a rich display for example, require an OS that prioritises the user interface for a good user experience. Thanks to the evolution of this subset of operating systems, the wearable evolved from a data providing device to an information-providing device. The OS allows for more complicated algorithms and communication with other devices or the cloud, which makes the wearable a more capable device. It is obvious that the success of a wearable is highly influenced by its OS. Apple, Google and Samsung used their knowledge of smartphone operating systems to create their own wearable operating system. Apple's Watch OS, Google's Android Wear, Samsung's Tizen OS or Pebble's Pebble OS are the most common at the moment of writing.

We first dive into more details by explaining the importance and the challenges of design. We then discuss notifications and user attention. Next, the role of wearables in the health and fitness domains is clarified. We then answer why wearables are always associated with the word smart. This is followed by an alaysis of the current wearable usage. After this analysis we discuss new kinds of interaction using wearables. Finally, we briefly introduce cross-device interaction with wearables and we summarise the essence of wearables.

2.1.1 Design

Computers and smartphones are mainly about extensive interaction with the user and continuous output. On the other hand, a wearable is a device that is targeting specific data, directly or indirectly from or by the user, as input and limited output. Depending on the location and the sensors of the wearable, specific data can be collected. A well-known example is a smartwatch that has a gyroscope sensor and a heart rate sensor, this wearable can help to track and evaluate the end user's health and eventually display some key information, for example today's steps.

Making a wearable device brings along a lot of design complications. First of all, it is a device that the user can and wants to wear. Hence, the device must be practical and pleasing to wear. Just as jewellery, the exterior look of wearables is fundamental for its success. This implies a well designed exterior for the device, a good form factor and good materials for skin contact and durability. While designing the exterior, the designer must keep in mind what hardware will be incorporated such the battery and sensors. The hardware of a wearable is often quite limited because of its size and production cost.

An analysis of actual smartwatch users highlights that 75% of the problems reported by users are directly related to the software platform [44]. These problems should be fixed by improving the design of the software on these devices. The design implications proposed in [44], are actually based on the same idea of this thesis, namely that it is essential to know the user's context and circumstances in which the user interaction takes place. By offering customisation options, the end user can improve this detection and specify how the wearable should behave.

Lowens et al. [38] analysed wrist worn devices' (WWDs) current design issues with existing devices and proposed design guidelines for novel WWDs. Based on user feedback collected from multiple websites comments, the authors distinguished key problems with existing consumer WWDs. Those problems, along with their design recommendations, are mainly about exterior quality and sensor accuracy. Unfortunately this is a consequence of the consumer market, to keep the price low for the customers, sacrifices must be made. Apart from this major problem, we do believe that today's available hardware and sensor analysis algorithms could be combined for high-quality wearable devices.

Still, a lot of improvements are possible on the user interface design for wearables. Two major obstacles with WWDs are their physical limitations and the context in which the interaction takes place. To overcome these difficulties, Motti et al. [43] propose a novel design paradigm for each one, which enables quick interactions that require low cognitive efforts. These design paradigms are flexible enough for designers to adapt and refine them to their likings. The first design paradigm is to maximise the use of micro interactions. Micro interactions occur when a user can complete a task such as user input or respond to system output in a short amount of time, with a low level demand of the user cognition and attention. For system output, examples are: variations with vibration, beeps, display, etc. As for the input, tactile and gesture input are the most common. The physical limitations makes it very important to show only essential content. The graphical user interface (GUI) can be designed in such a way, as to what is shown on the screen is only a part of an interaction continuum. The information is presented sequentially while the user navigates through the GUI.

To familiarise young children with wearables and to explore the children's creativity, a modular wearable construction kit ReWear is presented [31]. This proved to be an interesting platform that requires minimal training to create great wearable designs.

We can resume that the designer must find a way to keep interactions with the user as simple as possible, while trying to pass enough information in a short period of time. The user should not be annoyed by an interaction with a wearable device. This leads us to an important aspect of wearables: attention management, which is further discussed in the next subsection.

2.1.2 Notifications and Attention

A possible negative effect of wearables is the extra attention humans need to spend on it. Yet, the additional attention is more subtle and natural compared to other devices like a smartphone or a computer. Actually, in some situations the wearable reduces the attention that was initially needed by for example a smartphone. With a wearable that handles notifications the user can be notified through a vibration or a displayed message on the wearable, which could only require a quick body movement and glance. Without the wearable the user could for example be notified through a smartphone, which in many cases would ask more attention from and distract the user. A video analysis of smartwatch usage actually shows that little damage was done to the user's ongoing activities by the short watch interactions [53].

User attention is a very important aspect of a person's life. One needs to be interrupted by technology as little as possible. Lee et al. [33] present how user attention can be managed by the mobile OS. They state that the OS should predict the importance and complexity of new information for the user in order to decide when or whether the user and their current activities can be interrupted by a notification with that new information. Thus, depending on the current attention consumed by the user, the OS determines the right time to interrupt the user. To evaluate this right time is quite a challenge that needs a lot of information about the user. To make this attention management possible, Lee et al. [33] modelled interactions with the user the same way tasks are managed on a processor with multiple cores. Interactions with the user are scheduled based on a calculated priority. Switching attention from one activity to another also comes with a certain cost, called the *context switch cost*. If the priority of the interaction is greater than that of the current one, then the new interaction with the user can take place. Measuring how much attention a new interaction demands is the easy part. The more challenging part is to know how much attention the user is actually investing. The user could for example be driving, walking or con-

versing. The activity recognition therefore needs to analyse and evaluate data coming from the myriad of available sensors. This attention-aware notification system could be incorporated in new wearable operating systems.

Another intelligent notification mechanism has been proposed by Mehrotra et al. [40]. Based on how the user interacts with notifications, the system proposed in this paper builds an individual-based prediction model for predicting notification acceptance by the user. When the user manages a notification fast enough after its arrival, the system classifies the notification as accepted. Prediction models use user-defined or data-driven learning rules. The user-defined rules performed less than the data-driven learning rules. The data-driven predictor trained with "information-type and social circle" achieved the best result, which confirms that the importance of a notification is often a combination of *from who* and *what*.

These two studies could be combined in an OS that maintains knowledge about a user's social circle. The OS can improve its notification mechanism over time by collecting all the data from the notifications and the user. The long term result could be that, thanks to all the sensors that a user will have on him through wearables and other devices, all the attention could be managed to achieve a more natural co-existence with technology as Weiser predicted [61]. Unfortunately we believe that a user likes to change their habits from time to time. Achieving full predictability is hard because there will always be some randomness in the user's interests and thus notification responses. However, we strongly believe that wearables can have a positive effect on attention management if the notifications are managed in a correct way.

2.1.3 Health and Fitness

More than a decade ago Fogg discussed Persuasive Technology [19]. This principle currently plays a big role in wearable technology. For now, wearables are used very well to motivate healthy behaviour [20]. These devices make use of different sensing technologies to track movement or other activities. The user can view this information in visualisations displayed on the wearable itself, a website or a mobile app. To motivate the end user to be more active, they have to achieve goals and rewards. For example, a wearable can count footsteps and when the user reaches a pre-set goal a reward is won. This way, users get familiar with a healthy lifestyle. This persuasive technology often focusses on motivating users for change or improvement. Unfortunately, for long-term users, motivation for maintenance or more personalised motivation still lacks.

For the end user, reading direct sensor data such as hearth rate can be difficult to understand. To resolve this problem, Tollmar et al. [56] introduced the mobile health mashups system. Instead of showing sensor data values, the end users are receiving summary information based on data from multiple sources.

For example, the user could be informed with the following text: "You walk 80% farther on weekends vs. weekdays (15,000 vs. 8,300 steps)"[1]. Thanks to these one-sentence summaries, end users are more aware of their well-being. The evaluation of the system proved that participants were able to learn new facts about their health. To react on these findings, participants requested that the system would give recommendations based on the collected data. Finally, the authors express the need for health and fitness APIs to support more queries for accessing and aggregating specific types of data such as daily or weekly summaries of the captured data.

Kawamoto et al. [30] demonstrate that some aspects of the the human's lifestyle can be quantified using a single, low cost, non-invasive sensor that is available in wearables. Wrist-based accelerometers can detect various physical conditions such as fever or smoking cessation. The medical sector could see an increase of patients coming with health data, gathered through their wearables [52, 2].

2.1.4 Smart

Wearables are often referred to as smart devices. In reality this means that they are not completely dumb and have computing capabilities. A classic analogue watch is an example of a dumb device. Conversely, when sensors and wireless connectivity are added, the smartwatch can collect data and share it with other devices or services. This makes the dumb watch smarter, in a sense that it can sense and communicate. A wearable is truly a smart device when it can detect changes in the user's context [44]. Based on this context, a wearable can react or behave differently. Yet, detection of context is a complex task. By giving the ability to the end user to configure context-related information, the context sensing can be facilitated and improved.

With the IoT becoming more and more present, we will soon live in smart houses and cities. Wearables will be used as sources for context-aware systems to get more insights of the wearer. Wearables can for instance be used for identification. This can be achieved by means of Near Field Communication (NFC), Bluetooth or other techniques. David et al. [12] propose three possible approaches for contextual mobile interfaces. Every type of interface is based on the relationship with the environment. In these cases, the wearable would just be an element in the contextual system.

In the future, wearables can become smart individually by applying deep learning techniques [32]. Currently deep learning algorithms are rarely directly used on IoT devices because of the limited hardware capabilities. For now, deep learning that is used in mobile devices is mostly achieved by using services in the cloud in order to get smarter. The observations from Lane et al. [32] indicate

[1]Example taken from Figure 1 in [56]

that a basic deep learning approach is possible on the latest mobile and wearable devices. Deep KWS is a simple model that recognises a limited amount of spoken words from audio. This model showed an acceptable performance in execution times on the different target hardware platforms. Unfortunately the more complex models pointed out how easily this limited hardware can be overwhelmed. This means that smart wearable applications will not be responsive enough with more complex models. Nevertheless, all models are suitable for life-logging applications because they are processed at a slower pace. Knowing that mobile hardware improves very fast, we believe that more complex deep learning models will soon be feasible for wearables.

2.1.5 Usage Analysis

Wearables, and mainly smartwatches, have already been used for some years now. Even though those consumer wearables are an emerging technology, a lot of user feedback can already be useful for further development. The user's wearable usage has been discussed using users' comments from popular websites [44], video recordings [53] and an online survey [41]. Although users post comments mainly to express critique, they are useful for identifying problems from the point of view of the user.

Users confirm the importance of notifications over all the other functionality. The always-available nature of wearables means users do not have to go and find the device, which is a big advantage at home. The watch reduces the time spent on a smartphone, which leads to more availability of the user for other activities or people. Because of the acceptance of one-day battery on smartphones and the habit of recharging every single day, smartwatch users are neutral or satisfied with the one-day battery life of their device. However, in general, users expect more than one day battery from their wearable. For example, the one-week battery life from the Pebble smartwatch played a big role in its success. Finally users also regularly mention the lack of personalisation offered by a wearable.

2.1.6 Gestural and Skin Interactions

As discussed earlier, the form factor of wearables makes touch input on the screen more difficult. To improve the ease of use, other ways of interaction have been explored. First of all, the device itself can be used to interact through buttons or by rotating a crown. Interaction on the screen could be improved using special on-screen finger gestures used for example by WatchWriter [21], using the edge [50] or a rotating bezel as used on Samsung smartwatches. Using available sensors in the user's wearable, gestures from the user can be recognised and linked to specific actions. Using the user's skin for input is another

interesting possibility. An elicitation study conducted by Arefin et al. [3], resulted in a user's mental model of smartwatch gestures. These findings and design guidelines can improve gesture implementations in future wearables.

Early wearable gestural research introduced novel mobile and free-hand interaction [42]. Using a tiny projector, colour markers and a mounted camera, Wear Ur World (WUW) is an interface that augments the physical world using hand gestures. A similar approach is presented by Bailly et al. [4] by means of the ShoeSense wearable system that uses a shoe-mounted depth sensor pointing upward at the wearer to recognise arm and hand gestures. Although these systems propose interesting gesture recognition, the user's body and the context in which the system is used must be ideally set up for significant results. The advantage of these systems is the ability of detecting gestures using both hands or arms.

A potentially easier approach for the user, as it only requires an off-the-shelf smartwatch, is presented by Wen et al. [62]. Serendipity brings a promising finger-based gesture recognition, but still needs to improve the accuracy and range of their gestures.

Gestures detected with wearables can also activate interactions with other devices or places. For example, a smartwatch along with multiple gestures can digitally augment an office [5]. In smart spaces, the wearable can communicate with the smart environment and enable new gestural interactions.

The human skin can be seen as an interactive surface that is always available. Skinput, for example, uses an armband that senses finger taps on the forearm and hand [24]. A slightly different approach is proposed with SkinTrack [64], which uses a signal emitting ring and a sensor band. The sensor band can be worn under the smartwatch, or could be integrated in the smartwatch band in the future. Just by wearing the ring, the user's finger touch events on the skin surrounding the smartwatch can be tracked with high accuracy. Weigel et al. [60] achieve touch input via a novel class of skin-worn sensors, called iSkin. This very thin, flexible and stretchable second skin can sense two levels of pressure and is visually customisable.

2.1.7 Cross-device Interaction

Over the years, we have been using multiple devices separately and for different purposes. To overcome the limitations in functionality when we only use one device at a time, cross-device interaction comes into play. With this type of interaction, the user can combine devices to complete tasks and achieve a better experience. For example, in a lot of text editing applications we can find a toolbar with shortcuts to important actions. To free up the screen estate consumed by the toolbar, we can move these shortcuts to the user's smartwatch when the application is open on the main device [8]. Devices can work together

and complement each other, which leads to a symphony of interaction. Nebeling et al. [48] propose WearWrite, a crowd-assisted writing system that gives the ability to the user to continue editing and working on documents using a smartwatch on the go.

Creating these cross-device interactions is a complex process that involves data distribution and synchronisation, as well as device-specific communication. Efforts for helping developers with this difficult task have been explored recently. With the WatchConnect toolkit, developers can rapidly prototype cross-device applications with smartwatches [26]. A scripting language and environment called Weave [9] proposes a set of high-level APIs and abstractions to reduce the developing effort of cross-device interaction.

2.1.8 Summary

Wearables are promising devices that can be useful in any context. We believe that currently only the tip of the iceberg has been explored, thus we expect these devices to become a part of everyone's life in a certain way in the near future. Well-designed wearables will fade in the user's life by improving a multitude of little things that add up to an important difference in user experience.

2.2 End-User Development

Computing technology reached a point where it became a part of everyone's life. We are used to interact with all sorts of computers to retreive, publish or manipulate information. The applications that allow these interactions have been developed mostly by professional software developers. Unfortunately, because of the steep learning curve of programming, these professional software developers are scarce. Most domains and tasks are changing frequently, which leads to complex software requirements. These very demanding requirements are hard to implement by professional developers because of their limited domain knowledge and the slow development cycles. End-User Development (EUD) helps to bridge this gap. Lieberman et al. [35] define EUD as follows:

Definition 2.2.1 (End-User Development)**.** *A set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artefact.*

End-User Programming (EUP) is focussing on the *create* part of EUD. It is the most mature in research and practice of EUD, because creating a program is often the first step in the development process for end users. They have specific needs in their own domains, which they can fulfil by programming their own

applications using EUD techniques. EUD languages or applications aim to reduce the learning barrier in order to reach a maximum of users. To achieve that simplicity for any user to develop software, a trade-off is needed between learning cost, in terms of ease-of-use, and expressiveness, in terms of the scope of application [18]. The first major EUD programming environments were based on the spreadsheet paradigm [1]. By now, spreadsheets have been adopted by almost every desktop computer user, as we explained earlier in Chapter 1.

Today, users have access to data from a huge number of online services. When the end user needs specific information, data from multiple services can be combined. For example, finding a hotel in a certain town through one service, which has received a high rating from another service. These combinations of web-based content and services to create new applications are called mashups. Before the availability of good mashup EUD tools, only expert programmers could create mashups and the end user was only able to use them. However, the available mashups did not support all the needs of all their end users. Marmite [63] is an EUD tool that addresses this problem. Using Marmite, the end user can extract interesting data from the Web, process that data in a dataflow manner, combine it with data from other sources and output the result of the mashup. Marmite also uses the spreadsheet paradigm to view the data tables. This EUD tool showed good results with programming and spreadsheet familiar users, for other users the complexity of the program was still too high.

In practice, EUD has mainly been used with desktop computing. Now that Ubiquitous Computing is becoming a reality, we have to bring EUD approaches to other devices to allow for development of ubiquitous applications. End users should be able to create and customise applications for smart homes, applications that span multiple connected devices or other Ubiquitous Computing environments [49, 25]. The popularity of mobile devices, and computing technology in general, means that the end user has already some general knowledge of how applications work. Users have adopted various new interaction styles with mobile devices. User interfaces have evolved along with the user's knowledge and understanding of computing devices and how to interact with them. For example, by using mobile devices the user learned swiping gestures to scroll through lists, rotating a device, pinching to zoom and other interactions.

Existing EUD tools show promising results and invite for more research in EUD and better EUD tools. EUD tools cover three important parts, as classified by Ibrahim [29]. First of all, the chosen design approach determines which abstraction level will be used in the EUD tool. In a *device-centric* approach, the end user is aware of which devices are available, how they work, decides which ones will be used and finally how they possibly will work together. On the other hand, with the other design approach the presence of specific devices can be hidden to focus more on the goal. This *user-centric* approach focusses on the

behaviour of a program, i.e. what to do when a specific event occurs. Second, the generality targeted by the EUD tool is important. A *domain-independent* EUD tool is general, which means that it is usable for applications in any domain. For example, a EUD tool where the end user can create and manage events for any domain. The advantage is obviously that all end users can use this tool, but probably will not be able to manipulate important details from their respective domains. To give end users more possibilities and expressiveness, EUD tools can choose the *domain-dependent* approach. An example of this could be a football events manager EUD tool. This tool will only apply to end users interested in football, but will for example support adding football teams and other information to an event. Finally, the development interface of the EUD tool has to apply a (or a combination of) EUD paradigm(s). Designing for human-computer interaction is the most challenging and important task in EUD [7]. The prominent EUD interface approaches are natural language programming, programming-by-demonstration, visual programming and trigger-action programming.

2.2.1 Natural Language Programming

One way of making it possible for the end user to program is to let them express their ideas using Natural Language Programming (NLP). In recent times, voice assistants like the iPhone Siri from Apple provide functionality that lets the user interact with a device using voice commands. At the beginning these commands were short and limited, but now those voice assistants are smarter and understand more natural sentences from the user and can react more naturally. This is only a small example of what is possible with NLP. Myers et al. [46] explored how end users express ideas to computers. Based on their studies, they designed a new language and environment called Human-centered Advances for the Novice Development of Software (HANDS). HANDS is an event-based, user-centric and domain-dependent EUD tool that has performed well with children as end users. Liu and Lieberman introduced Programming Semantics [37]. Their work illustrates the principles of programming semantics with the Metafor editor, which converts English sentences to partially specified program code. For example, data structures are recognised from noun phrases, properties from adjectives and functions from verbs.

2.2.2 Programming-by-Demonstration

Another way to avoid writing actual code for a program is by demonstrating how a program should behave to achieve the wanted result. This approach is called Programming-By-Demonstration (PBD), also called Programming-By-Example (PBE). The PBD system can infer a program representing the logic

from the example demonstrated by the user. This inference is either happening fully deductively using Artificial Intelligence (AI) techniques or partially with AI and the help of the user for the rest [10].

To support end users to build context-aware applications for Ubiquitous Computing devices, Dey et al. [16] presented a PDB environment called aCAPpela. The end user can record visually an interaction such as a meeting and later annotate the events that occurred. With this information the end user can train the system to learn from the demonstration.

As with Marmite, the idea of offering the end user the ability to create mashups drove Lin et al. [36] to propose Vegemite. The difference with Marmite, is that Vegemite uses the PBD approach. Vegemite, which extends CoScripter [34], automatically populates tables with data from various sources on the Web and offers query possibilities. The PBD part within Vegemite takes place when the end user goes to the website and demonstrates how the data must be collected. Every step is added to a script along with the arguments representing the input values. These steps can be clicks, text input, pausing and copy and paste. Although the evaluated end users needed a solid session to get used to the PDB environment, all reported to be comfortable with the tool and are interested to use it again.

A problem that occurs with PBD after the program is inferred is representing the result in a comprehensive way to the end user. For example, the user study of Vegemite reports some difficulties due to the user interface. For this reason, PBD is in many instances assisted with natural or visual languages.

2.2.3 Visual Programming

The end user's difficult comprehension of program code can be resolved by representing coding expressions or techniques with graphical elements. Visual Programming (VP) environments offer a way of manipulating these graphical elements to develop a program [6]. In visual programming languages these graphical elements can make use of position, size, colour, shape and relations with other elements to encode a lot of information in a more human-readable form. These VP environments are often used for educational purposes.

The jigsaw puzzle metaphor is a popular VP technique because of everyone's familiarity of assembling simple jigsaw like pieces. This way the end user can connect graphical components. The "Playing with the bits" editor applies this metaphor to support user-configuration of Ubiquitous Computing environments in the home [28].

The publicly launched EUD tool Scratch uses a LEGO bricks metaphor which is similar to the jigsaw puzzle metaphor [55] to allow its users to combine programming statements. The end user can choose graphical "programming blocks" that can be snapped together in different sequences and combinations.

To make syntactic sense, while hiding the syntax complexity, blocks are shaped in specific ways. For example, a forever-loop block suggest, through its shape, that blocks should be placed inside. The end user can easily drag-and-drop blocks from various categories including motion, looks, sound, pen, control, sensing, operators and variables. Scratch has proven to be user-friendly for a first step into programming and has an online social community for end users.

Targeting the development of applications on mobile devices, Puzzle also uses the jigsaw puzzle metaphor [11]. Whereas Scratch helps the end user to learn the basics of real programming, with Puzzle the end user does not have to be familiar with low-level programming constructs to effectively create a program. The jigsaw pieces in Puzzle are an abstract representation of these low-level programming constructs and encourage the user to combine them in an interactive way.

Dey et al. [17] presented iCAP, a rule-based VP environment to visually prototype context-aware applications. The end user can sketch input and output elements that can be used in If-Then rules. The resulting context-aware application can be simulated with the iCAP rules engine, or be used in a real context-aware environment called the Context Toolkit.

2.2.4 Trigger-action Programming

While iCAP is a visual programming environment, the rule-based approach can be classified as another EUP paradigm called Trigger-Action Programming (TAP). The research involving iCAP found that the most common mental model for the end user, is rule-based as in TAP. This EUD paradigm lets the end user specify a *trigger* and an *action* in the form of "IF *trigger* THEN *action*".

Most behaviour that end users would want to implement in a hypothetical smart home can be expressed using TAP [58]. Ur et al. [58] categorised the wanted programming behaviours from their participants in four categories which express if only one trigger and/or one action is needed or multiple triggers and/or actions. 77.9% of automation tasks could conceivably be expressed as rules with a single trigger and single action, while the other 22.1% required multiple triggers and/or actions. The latter can be achieved by using the logical operators AND and OR. The authors found that a trigger generally should contain exactly one event, optionally alongside multiple conditions.

This has been confirmed and investigated in more depth by Huang and Cakmak [27]. A trigger can be an event or a state. A change that happens at a specific point in time is called an event, for example, "*the doorbell rings*". When a state is used as a trigger, it checks if currently a condition is true or false, for example, "*it is raining*". The popular customer TAP platform IFTTT only works with events. To get information about states, IFTTT uses events associated with state-changes. For actions a similar distinction can be made. Actions

can be instantaneous such as "*send email*", extended in time such as "*brew coffee*" or sustained such as "*turn the lights on*". Conjunctions of multiple triggers are possible in the forms of event and state(s) or state and state(s). Combining an event with another event is not possible, because two events happening *at exactly the same time* is impossible in theory and extremely unlikely in practice. An end user could for example ask "*when the sun rises and the doorbell rings, ...*". This combination of events can be rewritten to an event and state as: "*when the sun is up and the doorbell rings, ...*". The authors evaluated multiple triggers and actions with a user-study, which revealed the users' difficulties to understand the difference between events and states. To resolve this problem, the authors propose that the interface could be improved with prompts to warn users, disallowing confusing options, trigger duality and top-level statements.

Ur et al. [59] continued their research of TAP with an analysis of 200000 IFTTT recipes, a synonym for trigger-action rules. The analysis revealed that many different users wrote recipes, but only a few published more than 5 recipes. A few IFTTT end users were especially productive and shared more than 300 recipes. These extremely productive users, which are called "gardeners" in EUD research, make programs for standard problems that are widely used in order to avoid that other users reinvent them over and over again. Unfortunately for now, a minority of IFTTT recipes involve physical devices. Another remarkable observation is that users often create a rule that is actually already created and shared by other users, the most popular recipes are often duplicated. This issue could be related to the interface proposed by IFTTT. In IFTTT the user is not aware of all the existing recipes and the effort needed to find a recipe is sometimes more demanding than creating a new one. The available recipes on IFTTT grew exponentially at a rate of 50% per year including duplicates, this proves that IFTTT is a very successful TAP tool.

To allow for easy reuse of triggers in other rules, Trullemans and Signer [57] propose a multi-layered context modelling approach distinguishing between end users, expert users and programmers. The authors implemented the Context Modelling Toolkit (CMT) that enables the practical use of their multi-layered context modelling approach. The user can define a situation with an If-Then rule and later on use it again on the "IF" side. For example, "IF Person is Bob and Location is kitchen THEN Bob is in the kitchen" can later on be used as "IF Bob is in the kitchen THEN turn radio on"[2]. The definition of situation and context rules is simplified by the authors by means of *templates*. A template acts as skeleton for rules and can be filled in with values when used for a concrete situation or context rule. These templates can be created by expert users in the expert GUI and later on used by end users in the end-user GUI.

[2]Example taken from [57]

2.2.5 Summary

Research in EUD has established a stable ground with multiple interesting approaches. We have discussed EUD tools that target different domains and types of users. An important factor for the future of EUD is to connect tools with the Web and its services, but also with IoT devices. This way the end user can, for example, program their own smart home [13, 39]. The designer of an EUD tool should know the user, know the domain very well and finally accept that user will always make mistakes. To avoid the latter, the end user must be assisted and informed as much as possible. In the future, multi-device interaction will become the norm and must be taken into consideration in the design. Finally, social support will become more and more important in EUD activities on social platforms such as Facebook[3]. Crowdsourcing and social networks need to be further explored, as Nebeling et al. [47] introduced.

[3] `https://www.facebook.com` [Accessed 22 July 2017]

Chapter 3

Triggered Framework

In this chapter we will explain our thought process and demonstrate how we implemented our Triggered framework in the form of an Android application called Triggered that communicates with devices and web services. We have chosen the Android platform because of its popularity, the offered functionality and the developer support of our chosen wearables.

3.1 Introduction

Wearables should be more accessible and configurable for every type of user. We believe that this can be achieved by providing configuration to the user using the Trigger-Action Programming (TAP) approach. Functionality offered by wearables and other devices or web services could be used by the end user as triggers and actions in rules. From the literature study we know that this could be a difficult task for the inexperienced user.

The Triggered framework allows end users to use their devices and web services as input and output for rules. These rules can be triggered when events or state changes occur, which results in actions that are executed. Figure 3.1 shows how the users' devices and web services interact with the Triggered application on their smartphone.

A potential use case could be that the user presses a button on their Pebble Watch while they are at home to start brewing coffee with the coffee machine. The "*Locations*" device sets the "*currently at home*" state on true when the user is at home. The "*Pebble Watch*" device communicates via Bluetooth with the smartphone and triggers the "*button pressed*" event. This triggers the rule and the "*start brewing coffee*" action is sent to the "*Coffee Machine*" device using a web service.

Figure 3.1: Triggered framework

3.2 Configuration Through Rules

To explore how such a system should behave, we present three configuration methods that enable the creation of rules. These configurations differ in ease of use and expressiveness.

3.2.1 Closed Configuration

The *closed configuration* allows the users to choose a rule from a set of predefined rules. These rules consist of specific triggers and actions from devices, which can eventually be fine-tuned through specific adjustable values. This results in a straightforward process of creating rules. On the other hand, it restricts the possibilities of personalisation for the user. It is nearly impossible to foresee every possible rule in the system, thus the user will always have a limited choice of predefined rules. One could generate rules based on possible combinations of events, states and events, but this is very difficult because those combinations should create rules that make sense.

3.2.2 Half-open Configuration: Templates

The *half-open configuration* guides the user through the process of creating and editing rules by means of templates. Templates are more general predefined rules, which are constituted of event types, state types and action types. Types are used to categorise events, states and actions. A type can eventually be a subtype of a more general type. The user has more freedom of choice of events, states and actions that belong to a certain type.

3.2.3 Open Configuration

The *open configuration* offers complete freedom of choice to the user. The user starts with an empty rule and adds triggers and actions, which can be selected from all the available triggers and actions. Creating a rule from the ground up requires some understanding and creativity from the user. The complete freedom of this configuration allows the user to create unique rules that apply to their personal environment, needs and situations.

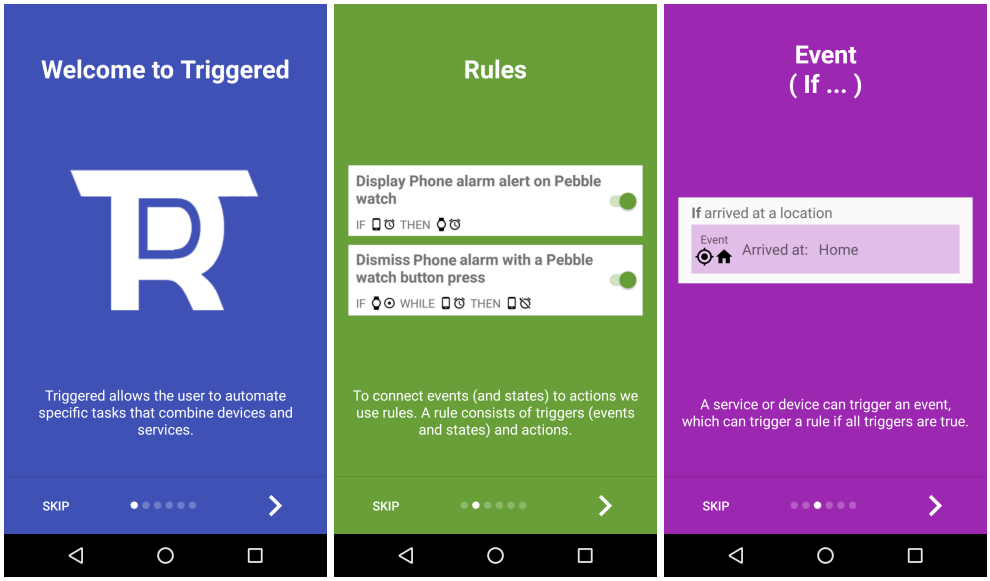
3.3 Application Interface

The application welcomes the user in the form of a quick introduction, which explains the core elements through small examples as can be seen in Figure 3.2. After the introduction the user arrives on the home screen shown in Figure 3.3, which is divided into three parts: the rules overview, the devices overview and the locations overview. The user has quick access to toggles to enable or disable rules and devices. Every rule in the rules overview shows a brief summary of the rule using symbols of the device and event, state or action. This quick summary helps the user to effectively have an idea of what a rule is doing. This visual information provides fast recognition of rules. Finally the user is presented with a button to create rules and locations.

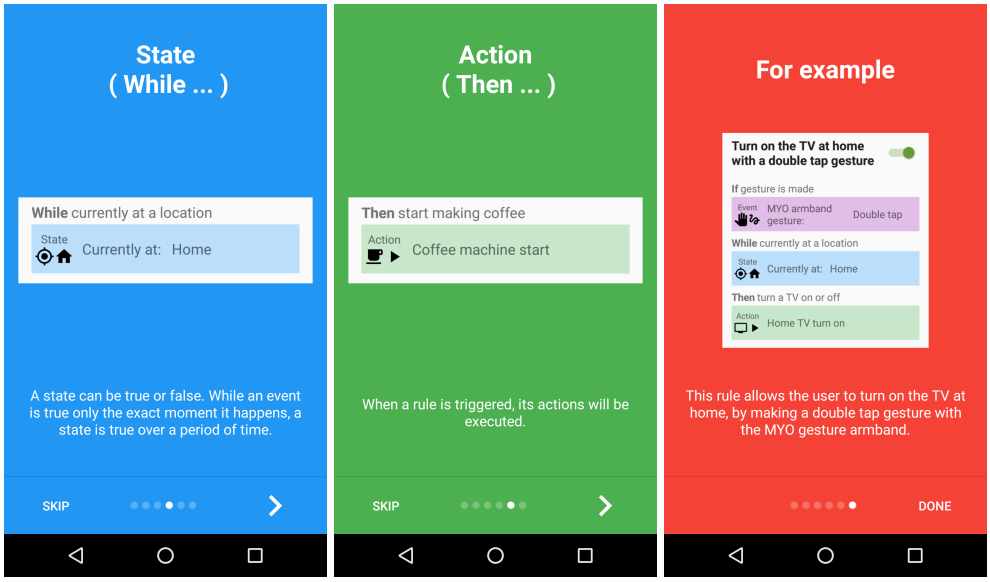
Throughout the whole application, searching functionality is proposed. For now this only filters on text, but this could be extended with device filtering or type filtering. Improvements in the searching functionality would improve the user experience.

3.3.1 Rules

With our application the user can create and edit rules. The lack of knowledge about how users would experience this process of rule creation, lead to multiple configuration methods proposed by our design to achieve the same result.

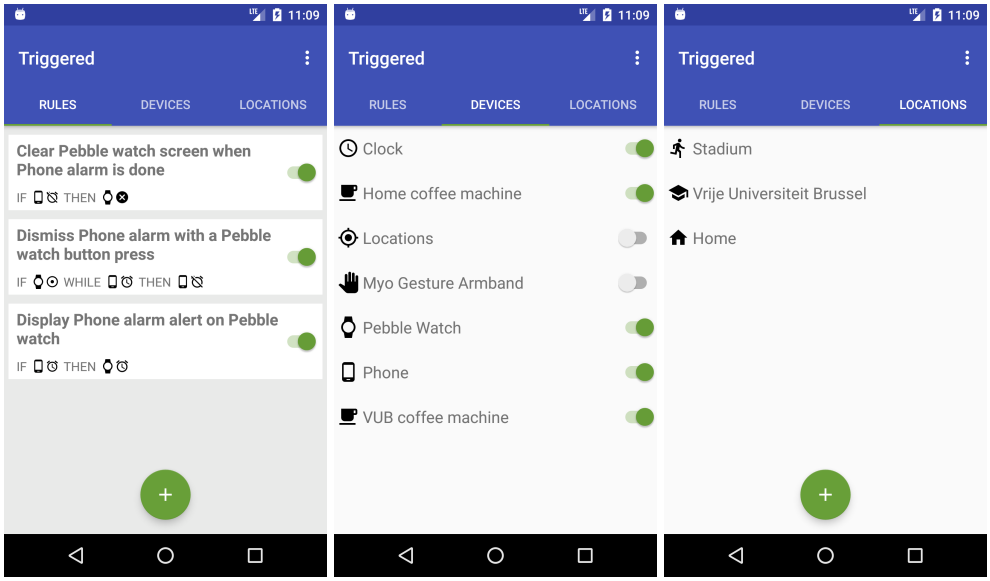


(a) Welcome (b) Rules (c) Event



(d) State (e) Action (f) Example

Figure 3.2: Application introduction



(a) Rules (b) Devices (c) Locations

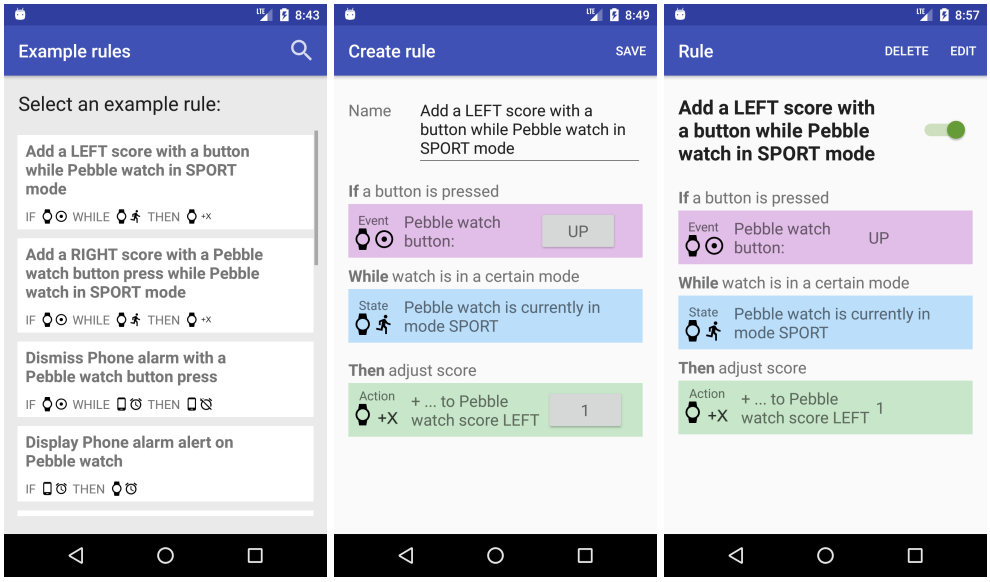
Figure 3.3: Application home screen

As explained before, we differentiate three possible ways of creating rules in our methodology. This is reflected in our application by selecting the preferred configuration in the administration settings. In each configuration, editing existing rules works with the same interface as the creation of rules.

3.3.1.1 Closed Configuration

The first method discussed is the *closed configuration*, involving little to no freedom for the user to edit the triggers and actions of rules. After pressing the create button on the rules overview interface, the user is prompted with a list of example rules as shown in Figure 3.4a. These are predefined rules that combine triggers and actions from specific devices and facilitate the creation process for the user, as no extra thinking or creativity is required. Nevertheless, the user is able to change values used by triggers or actions, for example the exact time in a time event. The list of example rules can be refined thanks to the searching functionality. As with normal rules, every example rule in the list shows a summary of the rule with symbols.

Once an example rule has been chosen, the user is prompted with the create rule interface illustrated in Figure 3.4b. The selected example rule contains a button-press event and a score-modifying action, both offering editable values. Finally the name can be changed as well. When the rule is ready, the user can save it and gets a detailed view of the created rule as displayed in Figure 3.4c.



(a) Example rules (b) Example rule create (c) Example rule details

Figure 3.4: Rule-creation process using example rules

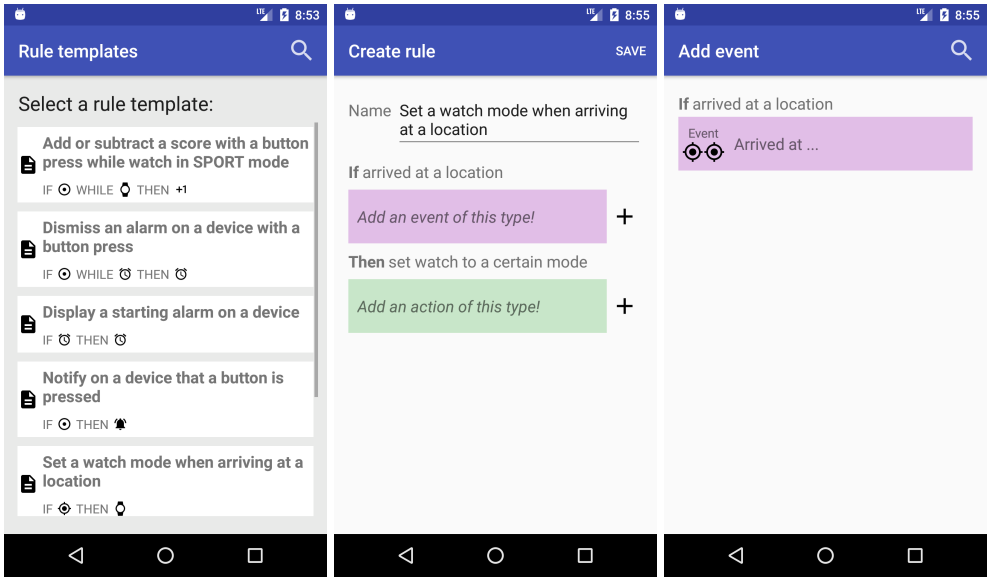
3.3.1.2 Half-open Configuration: Templates

In general a popular way of assisting the user in creating new content is by means of templates. We introduce rule templates, which are templates that represent rule ideas consisting of event-types, state-types and action-types. By using those types we give more options to the user in terms of devices that offer the functionality described by a certain type.

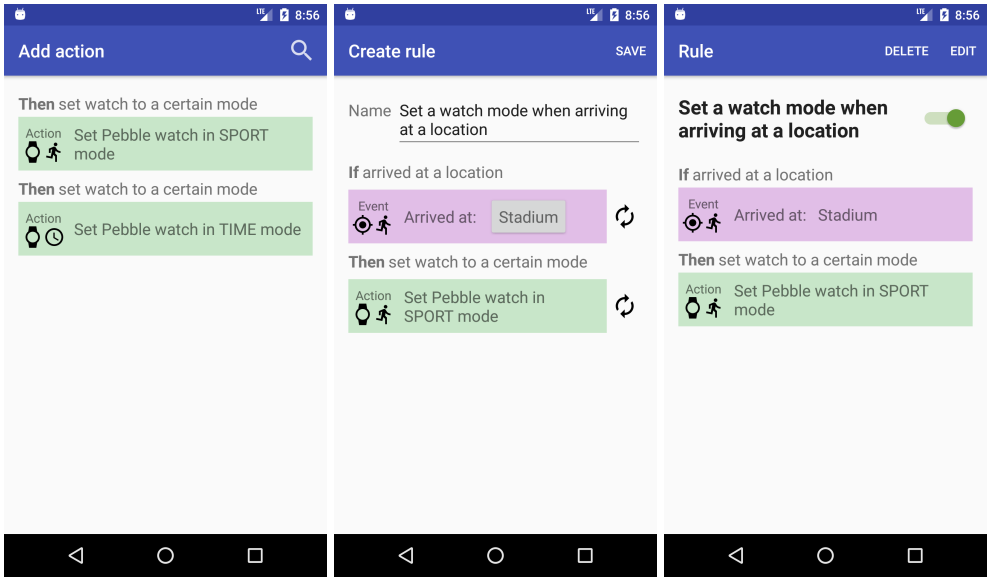
With this *half-open configuration*, the application presents a list of rule templates, such as the ones shown in Figure 3.5a, when the user initiates the rule-creation process from the rules overview interface. Again, every rule template in the list shows a summary of the rule with symbols. The user can search using text to refine the list of shown rule templates.

When the user has chosen a rule template, the interface to create rules with is shown. The template shows all the types that have to be filled in, which is illustrated in Figure 3.5b. In this case we can see that the template uses an arriving-at-location type of event. By pressing on the add button in this type, the user can select an event of this type. For now, in Figure 3.5c we can see that only one device offers this type of event. When the user selects this event, they are prompted with a list of possible locations from which they can select one. Adding states and actions works the same way. With the '*set watchmode*' type of action, we can see in Figure 3.5d that one device has two actions of this type. The selected events, states and actions are added to their type in the template as shown in Figure 3.5e. Eventually events, states and actions can

also be configured with specific values such as a specific location. Also, the instance of a type can still be changed to another instance of that type. When the template is completely filled in, the rule can be saved. Finally, the created rule is shown and could eventually be edited again.



(a) Rule templates (b) Rule template create empty (c) Rule template add event



(d) Rule template add action (e) Rule template create filled (f) Rule template details

Figure 3.5: Rule-creation process using rule templates

3.3.1.3 Open Configuration

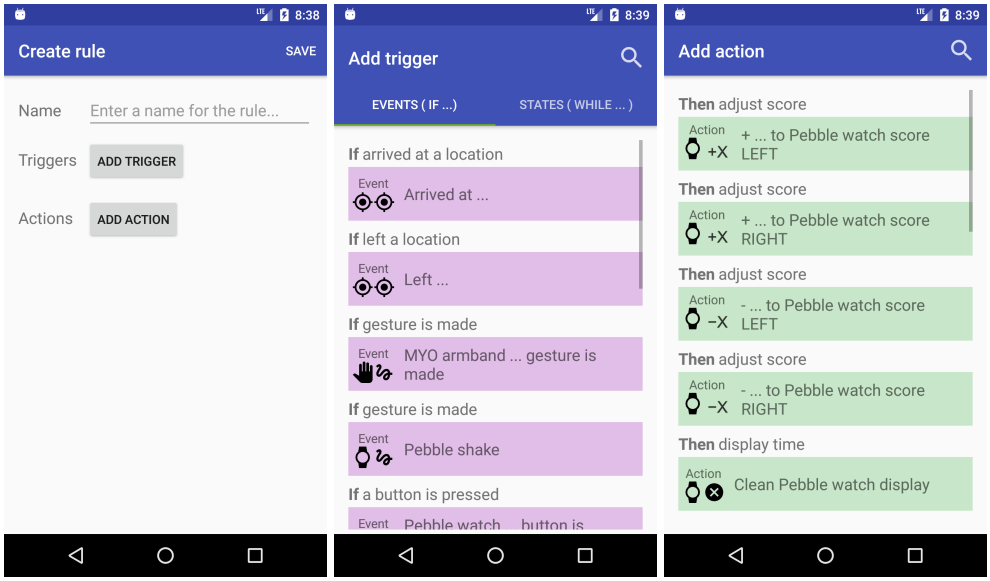
The last *open configuration method* allows the user to have full control on how events, states and actions from different devices are combined to create a rule. This task is cognitively more demanding for the user, because one should translate a solution to a real-life problem to a solution in our application in the form of a rule. Our research has proven that this task is still difficult for the average user. Remember that events and states are triggers. The ambiguity of a combination of events and states to achieve the desired trigger effect is still an issue. To assist the user the best we can, we applied a series of interface improvements proposed by Huang and Cakmak [27].

In the open configuration, the rule creation starts by showing an empty rule like in Figure 3.6a. First of all, the user can enter a name for the new rule. Then, two buttons are available to add triggers and actions. When the user adds a trigger to the rule, the interface in Figure 3.6b lists all available events and states. The user can add and delete as many triggers that are needed to fulfil what should trigger the rule. The same selection process applies to actions in Figure 3.6c. As with the above-mentioned configurations, the events, states and actions can eventually be configured through specific values. Finally, when the rule is finished and the user saves it, the details of the created rule are displayed.

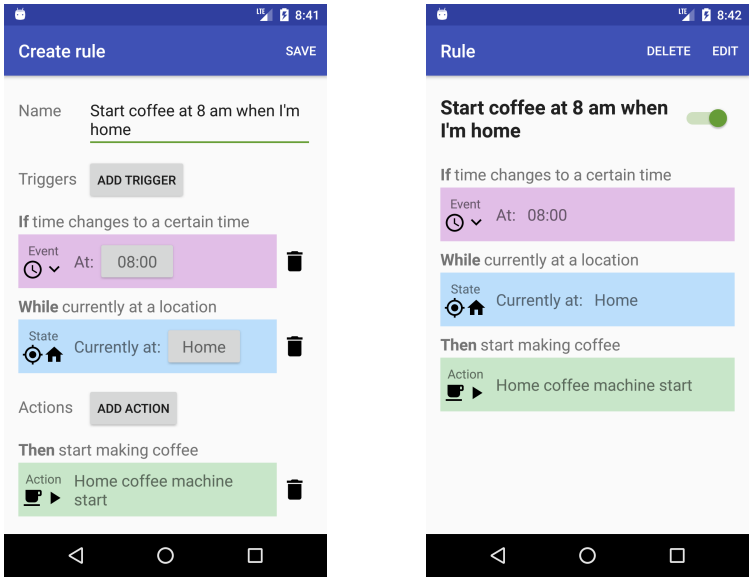
3.3.2 Devices

To be able to create and use rules, we need devices. Devices offer input and output functionality that is translated into events, states and actions. We added support for a few devices, which propose different kinds of functionality. But multiple devices can also propose equal functionality. For example, one can use the notification action to send a notification to a Phone or a Pebble watch. This means that both devices have an action of the same action type. All devices determine their available functionality, which are defined as event types, state types and action types.

In the device overview, shown in Figure 3.3b, the user can quickly start or stop a certain device by using the toggle next to the name of the device. A detailed overview of the events, states and actions from a device can be consulted by selecting it from the list. Finally, some devices can offer extra device-specific settings. For example in Figure 3.7 where the details of a Pebble smartwatch are shown. This device has multiple buttons that can be used to trigger button-press events and it can also detect when it is shaked. The Pebble smartwatch can display various kinds of information and does so with multiple watchmodes and notifications. The actual watchmode can be used as a state in rules. Within the sport watchmode, a score can be seen and modified. To modify the score,

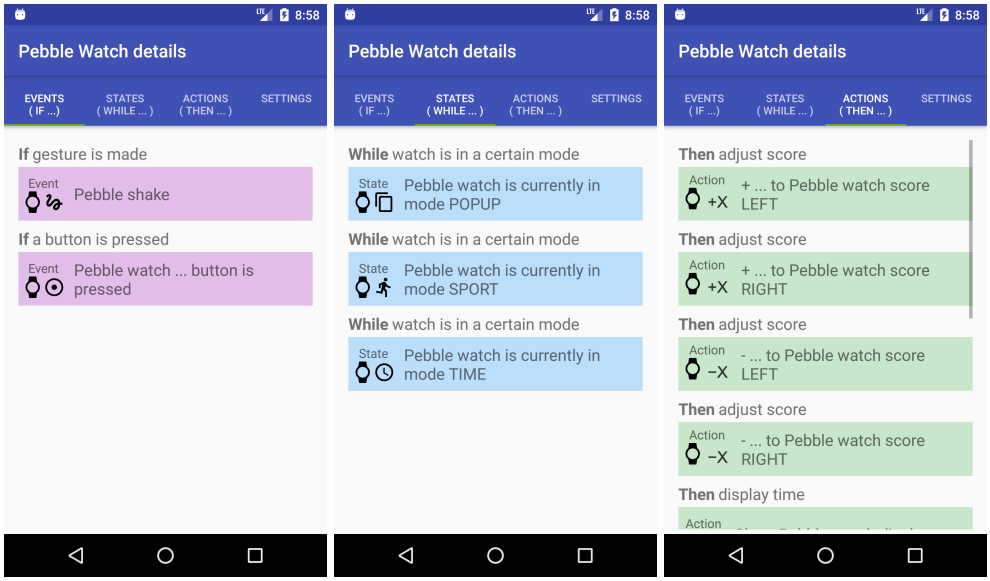


(a) Rule create empty (b) Rule add trigger (c) Rule add action



(d) Rule create filled (e) Rule details

Figure 3.6: Rule-creation process from an empty rule



(a) Events (b) States (c) Actions

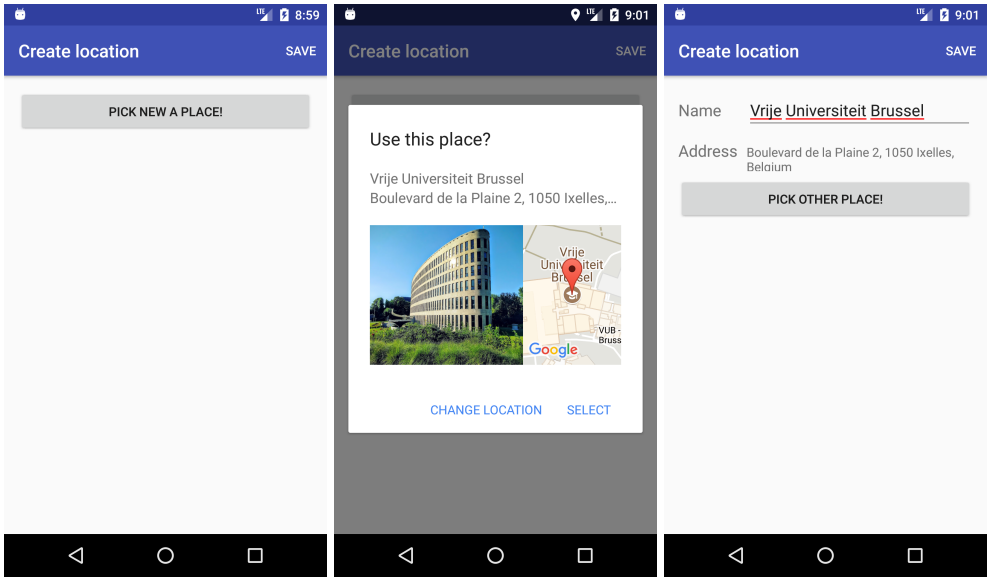
Figure 3.7: Pebble watch details

the device has specific scoring actions. Other actions are available to change the watchmode or display other information. Finally, a setting is available to see the score and reset it.

For now devices are fixed, which means they can not be edited nor be added or removed. A concrete application based on our design could allow users to add new devices to the system. This could for example be achieved by having an XML definition standard for devices, such that the system could detect and create the triggers and actions of a device for the system.

3.3.3 Locations

Another important feature of our application is the ability to use locations. The user can create and edit locations by pressing the create button on the location overview, shown in Figure 3.3c, or by selecting an existing location in the list. The application uses the Google Maps API service to find and select locations, as shown in Figure 3.8. The application saves the name, address and GPS coordinates of every created location. The locations device creates geofences, virtual perimeters in a radius around those locations, and offers events and a state that the user can use in rules. One event is triggered when the user arrives at a certain location and another event is triggered when the user leaves that location. In between those events, we know that the user is currently at that location, which can be used as a state in rules.



(a) Create empty (b) Select (c) Create filled

Figure 3.8: Location creation process

3.4 Application Implementation

In the previous section, we explained how the user experiences the interface of our application. This section will elaborate on how the rule system actually works, how the available devices are implemented and finally how the GUI is achieved.

3.4.1 Rule System and Rule Engine

For reasoning about context, and thus our rules, a good solution would be to use the Context Modelling Toolkit (CMT) [57]. This framework is implemented as a rule-based client-server architecture that offers a CMT Client Library and a CMT Server. Our application could use the CMT Client library to communicate through a REST API with the CMT Server that has a db4o[1] database and uses Drools[2] as a rule engine.

To avoid communicating over the internet, because of the relatively simple rules we need for our evaluation and finally to centralise everything in our Android application, we decided to create our own rule system and rule engine for Triggered. This rule system is implemented as an Android Service[3], a part of

[1] `https://sourceforge.net/projects/db4o/` [Accessed 22 July 2017]
[2] `https://www.drools.org/` [Accessed 22 July 2017]
[3] `https://developer.android.com/guide/components/services.html` [Accessed 22 July 2017]

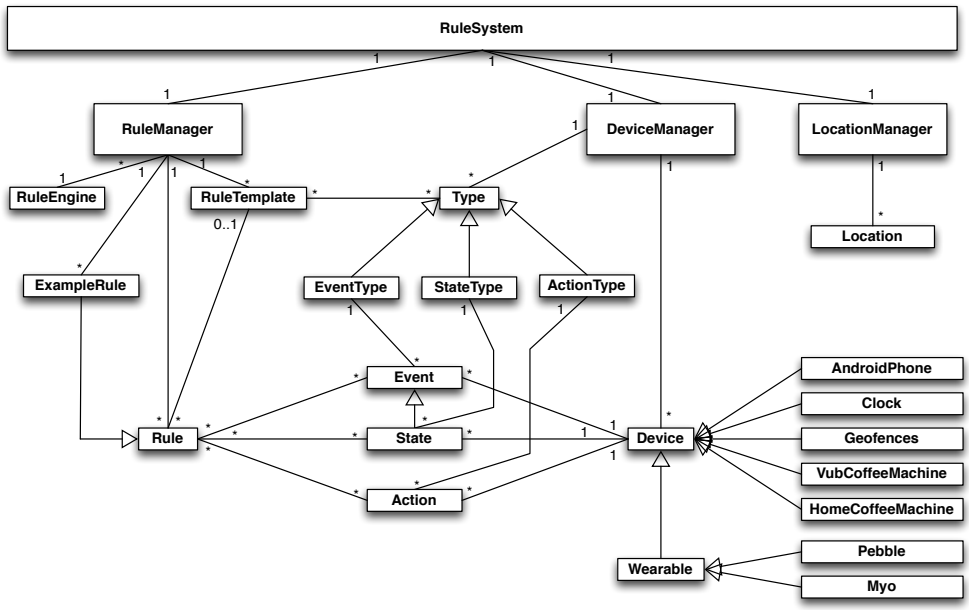


Figure 3.9: Entire rule system

the Triggered application which runs in the background. This way we avoid unnecessary complexity and data flows with an external rule engine. Due to a modular implementation, our rule engine could be enhanced or even replaced by, for example, incorporating the CMT Client library in our application to use CMT. Our rule system shown in Figure 3.9 combines a rule manager, a device manager and a location manager.

First of all, the devices that are used by the system are managed by the device manager. These are the building blocks of our system as they provide the events, states and actions. We will explain the actual devices in more detail later on in this chapter. The location manager offers functionality to create and manage locations, which can be used in rules for context awareness. Finally, the rule manager communicates with the rule engine and allows to create rules from scratch, based on rule templates or from example rules.

Figure 3.10 demonstrates how our rule engine works. Recall that a rule consists of triggers, which can be events or states, and actions. The rule engine receives information from devices when events or state changes take place. The first phase in the rule checking process is to identify which rules are affected by this newly acquired information. In our rule engine we decided to use the rules as observers using the observer pattern. Every trigger has a set of rules, using that trigger, which are notified when that trigger is triggered by a device. If this change means that a rule is completely triggered, the rule's actions will be executed. A rule is completely triggered and should be executed whenever all its states are true and every event has been triggered inside a certain time frame.

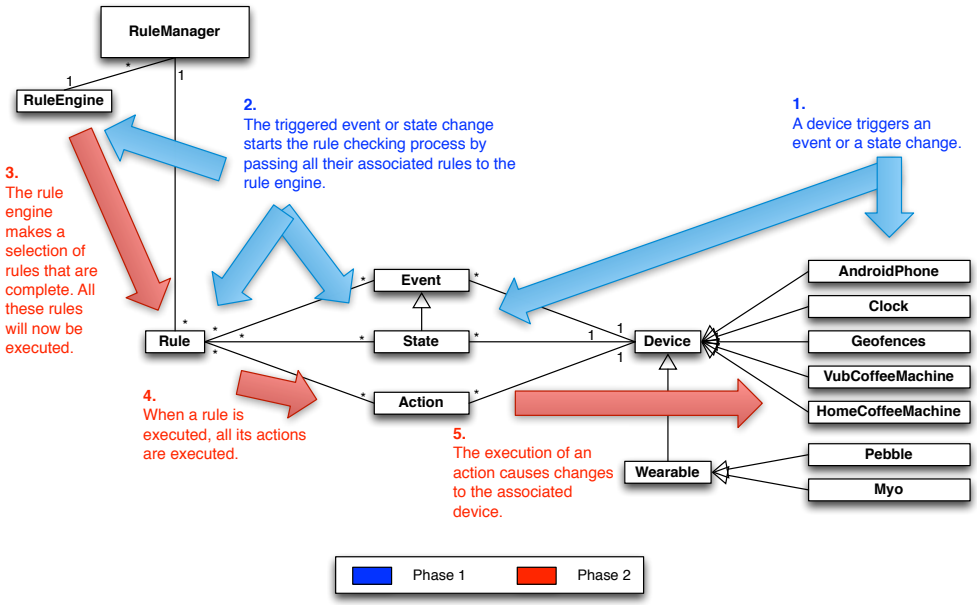


Figure 3.10: Rule engine phases

The rule engine analysed all affected rules and is ready for the second phase. The affected rules will now be executed by the rule engine, which means that all their actions are individually executed. After the executions, the rule engine has finished processing the trigger and is ready to handle new information to go through this process again.

This two-phased process is used to prevent that a state change could occur while the rule engine is evaluating every rule that is linked to the trigger. For example, a button press could be used to switch from mode A to mode B or vice versa. This button press event is used as a trigger for the two rules in Table 3.1.

In the case of a button press, these two rules will be checked sequentially for completeness. Without the two-phased process, rules would execute immediately when completeness is detected. In a scenario where currently mode A is active, we only want the button press to switch from mode A to B. But the sequential checking and execution would mean that a switch to mode B would happen before rule 2 is checked. Only after rule 1 is checked and executed, rule 2 is checked for completeness. Rule 2 is complete because mode B is now active, thus rule 2 will be executed as well and a switch to mode A happens. We can conclude that this is not what we wanted. With the two-phased process rule 1 and rule 2 will be checked for completeness, and afterwards only rule 1 would be executed.

Rule Name	IF	WHILE	THEN
Rule 1	button pressed	currently in mode A	switch to mode B
Rule 2	button pressed	currently in mode B	switch to mode A

Table 3.1: Two possible rules with the same event

3.4.2 GUI

Triggered is an Android application, which implies the use of the Android's Developer API[4], App Components and User Interface. An Android application uses the following App Components: Activities, Fragments, Services and Broadcast Receivers. Intents are used to send messages through the Android system to other App Components in the same or to different applications. Intents can be broadcasted to be received by Broadcast Receivers. To show lists of objects, RecyclerViews are used with the Adapter that corresponds to the type of the listed objects. Finally, we decided to base our style guide on the standards of material design[5] because it fits all our needs by default.

3.4.2.1 Main Activity

The MainActivity that is shown in Figure 3.11 is the central piece and home screen of Triggered. Before landing on the home screen, the user is prompted with the IntroActivity. This activity, which explains the core elements of Triggered, is started by the MainActivity using an Intent. When the user skips or went through the complete intro, the application returns to the MainActivity. Triggered has settings to change administrator settings, which can be reached from the MainActivity.

In order to communicate with the rule system, Android uses a technique called Binding. We created a RuleSystemBindingActivity that automatically binds with the rule system. MainActivity extends this activity, which means it is bound to the rule system and information can be retrieved.

This information involves the rules, devices and locations that are shown in their respective Fragment in the MainActivity. The RulesFragment, DevicesFragment and LocationsFragment extend the TrgrdFragment that allows for communication with the the parent Activity, MainActivity, using a Listener interface.

We decided to let every Activity and Fragment in our GUI extend RuleSystemBindingActivity and TrgrdFragment. This reduces code replication and encapsulates communication from actual GUI elements and code.

[4] `https://developer.android.com/index.html` [Accessed 22 July 2017]

[5] `https://www.google.com/design/spec/material-design/introduction.html` [Accessed 22 July 2017]

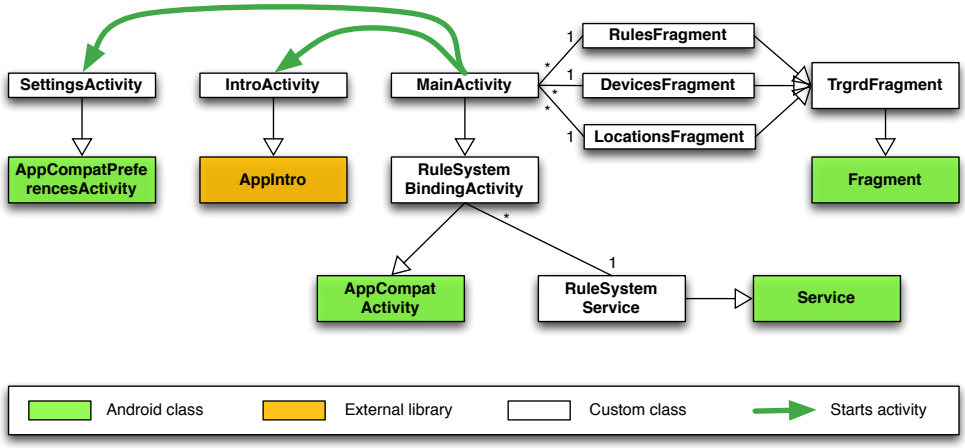


Figure 3.11: GUI elements connected to MainActivity

3.4.2.2 Rules, Devices and Locations

Each of the three above-mentioned Fragments are illustrated in more detail in Figure 3.12. Every Activity in this figure is extending an activity called "*RuleSystemBindingActivity*" to be able to communicate with the rule system.

First of all every Fragment shows a list of objects. The RulesFragment shows a list of rules, the DevicesFragment shows a list of devices and the LocationsFragment shows a list of locations. By selecting an object from these lists, one can see the details of that object in the RuleDetailsActivity, DeviceDetailsActivity or LocationDetailsActivity. The user can add rules and locations using the add button. For a new location, the user is prompted with the CreateLocationActivity. For a new rule, the shown activity depends on which configuration is selected in the settings.

3.4.3 Devices

Devices are essential for creating rules as they provide events, states and actions. Depending on the type of a device, it can offer certain functionality that we can use in our Triggered framework. For now we have a limited amount of devices, but in the future a user could, for example, add new devices to the system. We categorise devices in three categories, namely devices that are implemented on the Android smartphone, wearables and web services. Note that a '*Device*' in our system is an high-level abstraction of:

- An actual device, such as the Android smartphone
- A virtual device, such as the Clock (implemented on the Android smartphone)

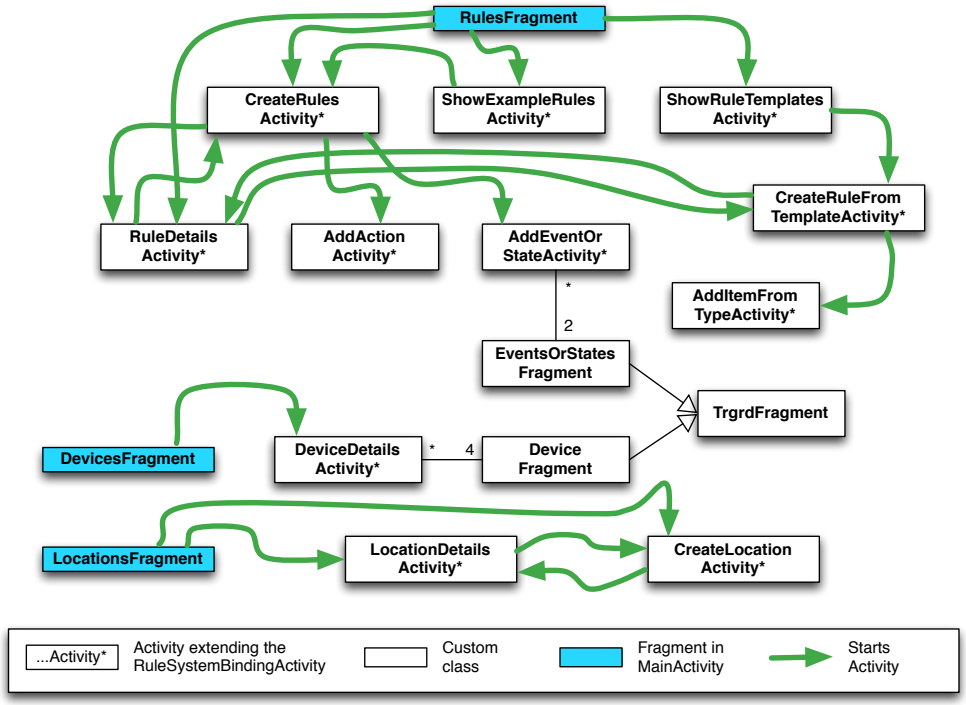


Figure 3.12: GUI elements connected to the MainActivity's fragments

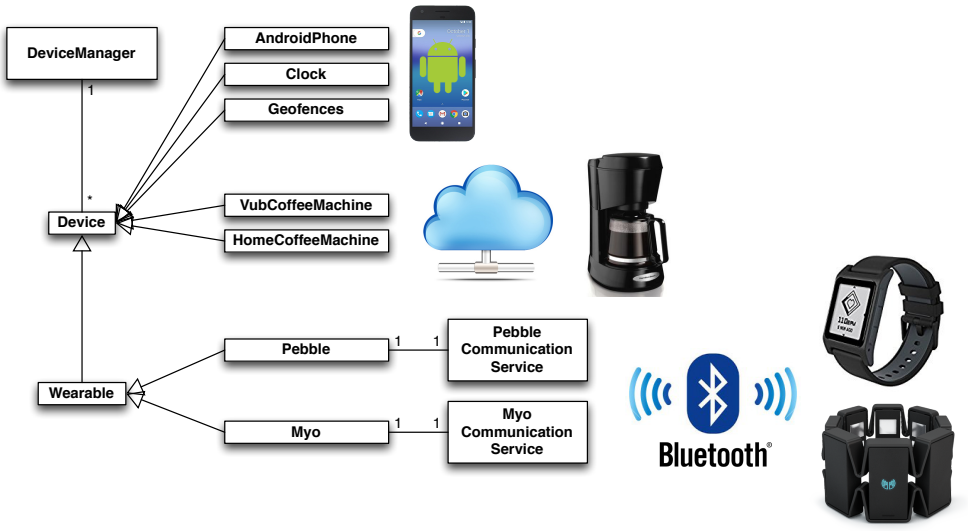


Figure 3.13: Devices

- A wearable, such as the MYO gesture-recognition armband

- A web service, such as the Coffee Machine (a device that is connected to the internet)

3.4.3.1 Phone, Clock and Locations

Our application runs on an Android smartphone. The Android operating system provides a broad range of functionality and services, which can be used for developing applications on the platform.

For this reason, our first device is an Android Phone as can be seen in Figure 3.13. Using Androids broadcasts, we can intercept and read events from the actual phone. For example, the OS natively sends broadcasts about alarms and phone calls. We translate this information to events and states for our rule system. On the other hand, it is possible to send official broadcasts to the Android system or personal broadcasts through our own Android application. The actions of the Android Phone device in our system send out official broadcasts, such as dismissing the alarm or muting the phone.

The Clock device is a virtual device that is implemented on the Android smartphone. This device is listening for an official time broadcast to know the time and when it changes to propose an "*at* time" event and a "*between* time *and* time" state.

The Locations device is another virtual device that is implemented on the Android smartphone. It is using the Android Geofencing service, which is running on the Android smartphone. This service creates geofences around the user's locations, which allows us to receive events when the user arrives at and leaves a certain location. This allows us to trigger "*arriving at* location" and "*leaving* location" events, as well as the "*currently at* location" state.

3.4.3.2 Wearables

We currently support two types of wearables. First of all, the Pebble watch which is a popular smartwatch with buttons, input sensors, a vibration motor and a black-and-white display. Second, the MYO gesture recognition armband, which analyses the user's forearm muscles to determine which gestures are made with that arm and fingers. Both wearables were chosen because of their interesting functionality and developer-friendliness. We used the corresponding Android SDKs to communicate with our Android application. To avoid blocking the rule system service, we decided to create a communication service for each wearable that is started and stopped by the according device in the rule system.

(a) Time mode (b) Sport mode (c) Popup mode (d) Notification

Figure 3.14: Pebble application

These two wearables communicate with the Android smartphone via Bluetooth, which is shown in Figure 3.13. On the other hand, other wearables and devices could communicate with our system by means of different communication technologies.

Pebble Watch To give the user some configuration options, we developed a simple application on the Pebble watch that is displayed in Figure 3.14. This application contains three interchangeable watchmodes: a time mode, a sport mode and a popup mode. All watchmodes show the time and mode-specific information. The time mode displays the actual weather and today's date. This mode only shows information and does not offer special functionality for now. The sport mode displays the user's current heart rate and a score that can be modified. The last watchmode is used to display popup messages. Finally, the user can send notifications to the Pebble watch which contain text and can be dismissed, while popups are used to show quick information for a short period of time.

MYO Gesture-recognition Armband Using the MYO gesture recognition armband we can receive gesture information and analyse it thanks to their Android SDK[6]. When the MYO is connected to the smartphone, it sends broadcast through the system every time a gesture is made. We simply read those events and translate them to device events in our system. The MYO recognises multiple gestures listed in Figure 3.15, but the accuracy is not the same for every single gesture. Therefore we decided to only support gestures which were accurate during our tests, namely the wave-in, wave-out and double-tap gestures.

[6] `https://developer.thalmic.com/docs/api_reference/android/index.html` [Accessed 22 July 2017]

Figure 3.15: MYO gestures.
Source: https://www.myo.com/techspecs [Accessed 18 July, 2017]

The other gestures, i.e. fist, fingers-spread, rotation and pan, can be supported in the future if their accuracy is good enough.

3.4.3.3 Web Services

All the above-mentioned devices are directly integrated in the smartphone, such as the Clock, or are communicating directly with the smartphone on which our application is running, such as the Pebble Watch. Web services can also act as devices in our system and offer triggers and actions. Asynchronous HTTP requests are currently used to communicate between the web service and the application.

Coffee Machine A supported device in our application that uses this technique is the coffee machine. This device is connected to the Web and offers functionality through a web service. For the action to start making coffee, we simply send an HTTP-call to this device as shown in Figure 3.13.

3.5 Summary

We created a good foundation with our framework, which allows for automation and configuration with a small number of devices and services. Triggered allows the user to use wearables, like the Pebble smartwatch and the MYO gesture recognition armband, as input devices with triggers and as output devices with actions. By integrating locations and the geofences around them, our application provides context-aware (more specifically, location-aware) functionality that can be used in rules.

Triggered is focussed mainly on integrating different ways of rule-creation, to be able to analyse their strengths and weaknesses. Our application could be more complete if other management functionality are added such as object removal, a real data storage and more advanced filtering for search purposes.

With a limited amount of devices, a basic GUI, a simplified rule engine and limited resources, we believe that Triggered shows the potential of the idea that drives this thesis.

Chapter 4

Methodology

Triggered allows for a personal usage of the user's devices by means of rules. To create such rules we propose three configuration methods. Each configuration method uses a different approach, which balances ease of use and configuration freedom. The *closed configuration* assists the user the most by proposing a list of predefined rules from which the user can select one and eventually adjust some specific values in the rule. The *half-open configuration* is a more general version of the closed configuration that proposes a list of rule templates, which are the skeletons of rules. These templates can be used by the user to create rules with more freedom than the closed configuration. Finally, the *open configuration* starts the rule-creation process with an empty rule that can build up by adding events, states and actions.

In this chapter we analyse how users experience these configuration methods in multiple scenarios. We expect to get a better insight on our Trigger-Action Programming (TAP) system. On the other hand, we would like to know if this idea of configurable wearables can enrich the use and popularity of wearables.

4.1 Research Questions

We will carry out an experiment, using our proposed Triggered framework and a questionnaire, in order to answer three research questions.

Research Question 1: *Is there a difference in efficiency and accuracy between the configuration cases?*

First of all, we want to observe if the three configuration methods are different in terms of efficiency and accuracy. The efficiency is the time that the user needs to create a rule, while the accuracy is a score that determines how accurate the created rule is. Logically the difficulty of creating a rule is low with the closed

configuration, medium with the half-open configuration and high with the open configuration. We expect to see this affecting the efficiency and accuracy of the rule-creation process. A low difficulty should translate to high efficiency and accuracy, while a higher difficulty should result in a lower efficiency and accuracy.

Research Question 2: *Is there a difference in usability between the configuration cases?*

Second, we will compare the usability of every configuration method to see whether they are different. To acquire such data we will use a USE questionnaire that measures usability with four factors, namely Usefulness, Satisfaction, Ease of use and Ease of learning. We believe that the level of configuration-freedom should have an effect on the usability of a certain configuration method for the rule-creation process. If a configuration method offers few to no personalisation options, one could experience this as less usable than a configuration method that offers more personalisation options.

Research Question 3: *Does the efficiency and accuracy of the configuration cases have an influence on their usability?*

Finally, we will investigate whether there is an influence between the two above-mentioned aspects of configuration methods. The three configuration methods have a certain level of difficulty which is tightly coupled to the degree of freedom in terms of personalisation. For example the closed configuration method offers a straightforward rule-creation process with limited personalisation possibilities. Thus, we assume that efficiency and accuracy will correlate with usability.

4.2 Research Methodology

For this thesis the Research trough Design (RtD) [65] methodology is used. This methodology starts by making a design that implements the proposed idea, the three configuration methods in this case. The design is then used to analyse and compare the three configuration methods in order to get actual insights and conclude with design implications.

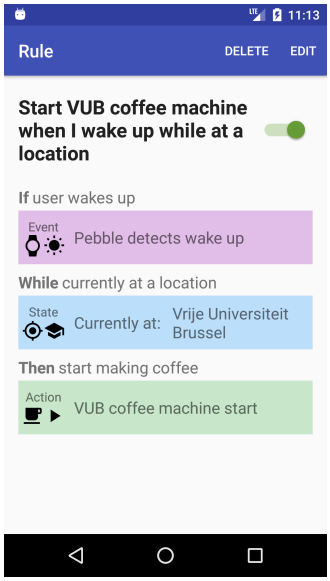
The units of analysis of our study are the three configuration methods, namely the *closed configuration*, the *half-open configuration* and the *open configuration*. Our design implements those configuration methods in an Android application. Based on the evaluation of our design, we will know whether one specific configuration method is better or a combination of multiple configuration methods is more interesting to achieve our goal of configurable wearables.

4.3 Procedure

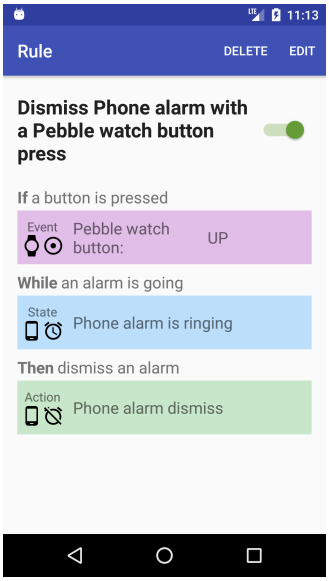
To evaluate our configuration methods we will use the within-subjects method, which means that every participant will test the three configuration cases in a random order. This random order combined with the randomly assigned scenarios reduces bias, more precisely the carry-over learning effect. We decided to target middle-aged participants that are used to mobile phones and applications, which reduces bias coming from inexperienced mobile users. Every participant is tested in a controlled setting composed of an Android smartphone with the Triggered application installed, a computer to analyse logged data from the smartphone, a Pebble watch, a MYO gesture-recognition armband and finally a camera to record the experiment if the participant agrees.

We created three possible scenarios in which a user could use Triggered to automatise specific tasks. This means that every configuration case can be associated with one scenario. Every scenario describes two tasks, namely an implicit and an explicit task. The difference between those two types of tasks is that the explicit task needs an explicit action from the user to be executed, while the implicit task does not and can detect by itself when it should be executed. It will be interesting to see if this small difference affects the user's thinking process and thus the rule-creation process. These scenarios will be explained in more detail later on in Section 4.4.

During the evaluation, the participant fills in a form that contains all the parts of the evaluation. For example, the evaluation form for participant 3 can be consulted in Appendix A. First of all, the participant's profile information is asked followed by a permission for recording the evaluation. Next, an introduction of Triggered is given using an example task. After reading this, the participant is prompted with the introduction in Triggered that briefly explains the core elements textually and visually. When the participant explored and understands the application, the configuration cases can be tested. For each configuration case the form states using which case, in which scenario and in which order the participant should create the rules from that scenario. Right after creating a rule, the participant fills in a Likert scale on how certain they are that the rule will work. Then, the participant is asked to perform the task in order to test the created rule. After this, another Likert scale must be filled in on how satisfied the participant is of the rule they made. At the end of every case a usability survey of eight questions, a selection from the USE questionnaire as explained in Section 4.1, must be filled in. After completing this process for the three configuration cases, a user experience survey about the complete application must be filled in. This user experience survey consists of 21 positive-to-negative, or inverted, Likert scales. On the last page, a top three of configuration cases is



(a) Rule 1a



(b) Rule 1b

Figure 4.1: Scenario 1 rules

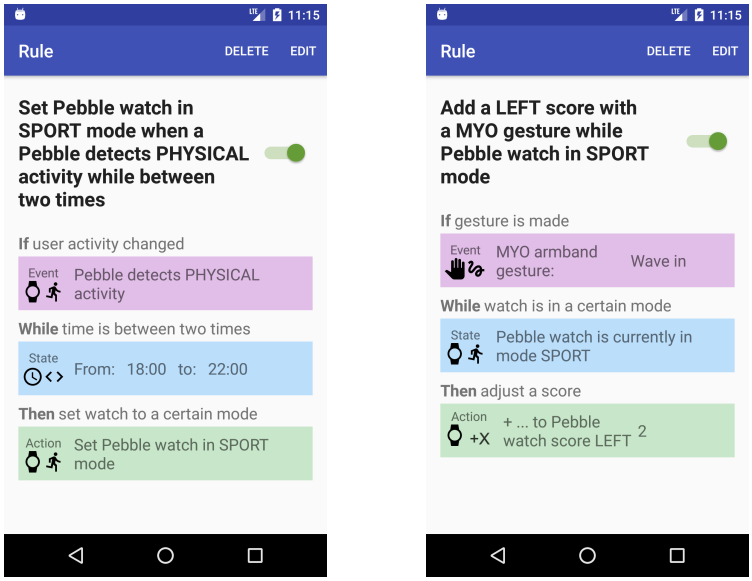
requested along with a motivation. Finally, the participant is asked whether they would use a finished version of this application, or a similar system, as well as a motivation and general remarks.

4.4 Scenarios

For this evaluation we imagined three possible scenarios. For each scenario the participant will create one implicit and one explicit rule. All the rules have the same level of complexity, which means that the difference between the rules is minimal and does not affect the rule-creation results.

4.4.1 Waking Up

The first scenario happens when the user wakes up. When one wakes up at a location, for example at home, then the coffee machine should automatically start making coffee. This rule consists of a wake-up event, a location state and a coffee starting action as can be seen in Figure 4.1a. The second proposed rule allows one to dismiss an alarm on a certain device by pressing a button on their smartwatch. Concretely, when an alarm on the user's phone is going off, the user can dismiss it by pressing the UP button on a Pebble watch. This rule is shown in Figure 4.1b.



(a) Rule 2a (b) Rule 2b

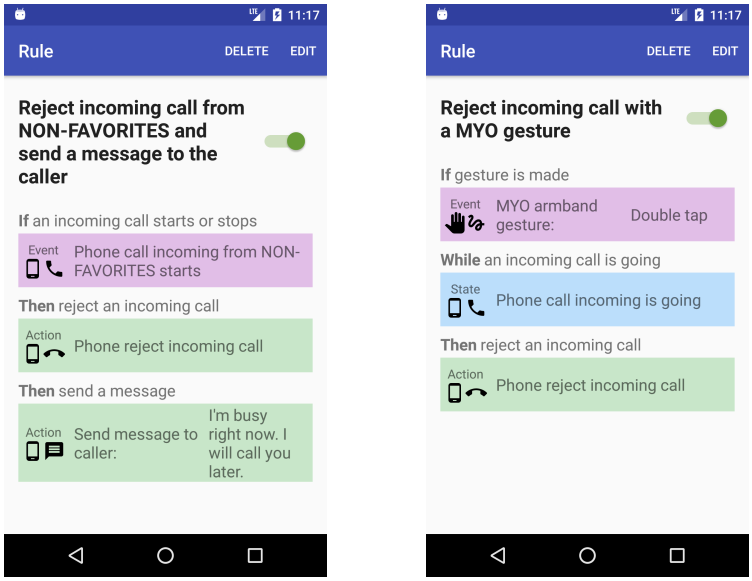
Figure 4.2: Scenario 2 rules

4.4.2 Sport

For the second scenario, we can imagine a sporty person as the user. One could make a rule to automatically set a watch in SPORT mode when a physical activity is detected between two times. More precisely, if the Pebble watch detects a physical activity between 18:00 and 22:00, the Pebble watch should be set in SPORT mode. The details of this rule in Triggered can be seen in Figure 4.2a. Another possible task that could be automated using Triggered is to keep a score of, for example, a football game. An athlete, the referee or even supporters could keep the current score of the game on their smartwatch. The Pebble watch offers this scoring functionality. One can make a rule to add points to a certain team on the Pebble watch's scoring board by making a gesture while the Pebble watch is in SPORT mode as shown in Figure 4.2b.

4.4.3 Watching TV

A third scenario in which automation rules could come in handy is while watching TV at home. If the user does not want to be disturbed by someone who is not in their favourite contacts, the incoming call could automatically be rejected and a message could be sent to that person. The details of this rule can be seen in Figure 4.3a. If one does not want to automatically reject calls but decide which ones to reject when they come in, one could make a rule to decide whether to



(a) Rule 3a (b) Rule 3b

Figure 4.3: Scenario 3 rules

reject a call consciously by making a gesture. Thus, a user can make a gesture with the MYO gesture-recognition armband while an incoming call is ringing in order to reject that call as shown in Figure 4.3b.

4.5 Participants

We collected data for our study from 15 middle-aged participants that are experienced with mobile phones and applications. From these 15 participants 12 were male and 3 were female. The mean age was 26,3 with a minimum of 22 and a maximum of 38. 2 Participants had a secondary education diploma, 6 participants had a bachelor's degree and 7 had a master's degree. Only 2 participants had experience with automation applications and none had experience with wearable technology.

4.6 Summary

In this chapter we explained how the three proposed configuration methods are evaluated using Triggered and a questionnaire. Every participant was tested and observed during the rule-creation process in all three cases. The results of these experiments will allow us to analyse how participants handle all the configuration methods. This analysis is given in the following chapter.

Chapter 5

Results

In this chapter we analyse the results of the experiments. First of all we summarise the data to get an overview. After a thorough investigation of this overview we will be able to determine which statistical tests can answer our research questions. Based on our analysis we propose three design implications that can be applied in the future.

Overall, we found that the results between the cases were close to each other. The overview of the data confirms our expectations in terms of increasing time needed to create a rule and decreasing accuracy of the rule made going from the closed configuration to the open configuration. Although this is the case, all rules were made in a reasonable time and with good accuracy. Note that the closed, half-open and open configuration are respectively referred to as closed, templates and open in our data. The following sections will discuss in more detail the results for every aspect of our evaluation.

5.1 Efficiency and Accuracy

The first type of data that we will investigate is the time needed to create a rule, called the efficiency, and the accuracy of the rule that the participant made. This information will give an answer to the first research question.

Research Question 1: *Is there a difference in efficiency and accuracy between the configuration cases?*

5.1.1 Efficiency

Our first interest is the efficiency of the rule-creation process. In general participants needed between 1 and 2 minutes to create a rule. This is a good result as this process is only needed once and needs little to no time later on for editing and turning the rule on and off.

The first rule that every participant made generally needed more time in order to understand the core elements and workflow of the application. Another anomaly occurred when users were stuck for a moment and did not want to give up, which resulted in a significantly higher time needed to complete the rule.

Table 5.1 shows a summary of the efficiency, expressed in seconds, for all the configuration cases. These numbers are represented visually with boxplots in Figure 5.1. These boxplots already highlight a difference in efficiency between the three configuration cases. The fact that the medians of every box do not fall into the range of the notches of other boxes means that the difference is most likely significant. The outliers above all the boxes reflect the anomaly when participants were stuck.

	Closed	**Templates**	**Open**
Min.	30.0	38.0	25.0
1st Qu.	42.5	60.0	82.0
Median	57.0	78.0	102.5
Mean	63.6	81.2	129.3
3rd Qu.	75.5	99.0	136.8
Max.	208.0	175.0	358.0

Table 5.1: Efficiency summary (in seconds)

Finally, to analyse if the configuration cases are equal or different we must perform an Analysis of Variance (ANOVA) test. This ANOVA test requires that all groups are normally distributed and have the same variance, which is not true with our configuration cases. The difference in variance between the cases is proven through the F-tests shown in Figure 5.2.

The Kruskal-Wallis rank sum test is a parameter-free alternative to ANOVA, which we can use with our configuration cases. The result of this test shown in Figure 5.3 rejects the null hypothesis that the groups are equal. We can conclude that there is a significant difference in efficiency between the configuration cases.

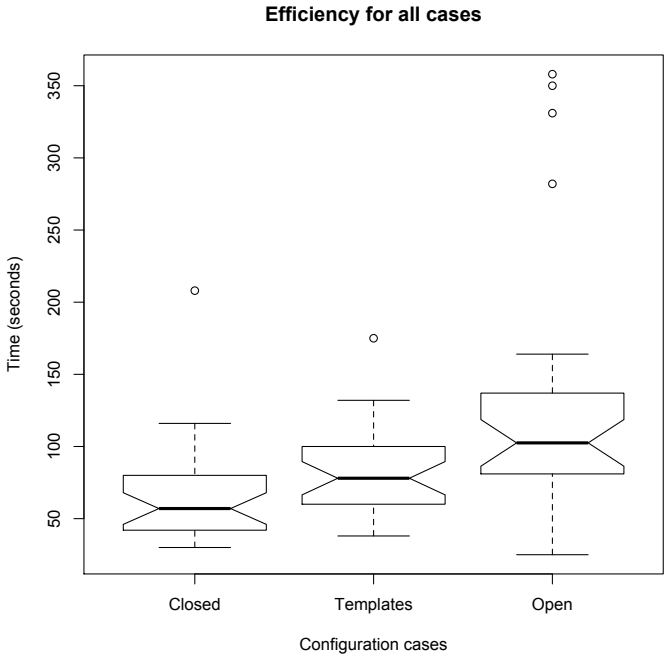


Figure 5.1: Efficiency boxplot

5.1.2 Accuracy

The accuracy of the created rules was generally very good. Every rule consisted of 3 elements from events, states and actions. A score on a scale of 5 was given, which was defined as follows:

0/5 = Participant gave up and did not make a rule.

1/5 = Nothing in the rule was correct.

2/5 = 1 out of 3 elements was correct.

3/5 = 2 out of 3 elements were correct.

4/5 = All 3 elements were correct but small errors with chosen device or values.

5/5 = Rule was completely correct.

With the closed and half-open configurations, almost all participants created correct rules. The open configuration showed that rules were not always completely correct when participants had to make them from scratch. The errors that participants made were mainly caused by the ambiguity between events and states, which was also highlighted by Huang and Cakmak in [27]. Although

```
F test to compare two variances

data:  efficiency$Closed and efficiency$Templates
F = 1.2822, num df = 29, denom df = 29, p-value = 0.5076
alternative hypothesis: true ratio of variances is not
equal to 1
95 percent confidence interval:
 0.61026 2.69380
sample estimates:
ratio of variances
          1.282154
```

(a) F-test between Closed and Templates

```
F test to compare two variances

data:  efficiency$Closed and efficiency$Open
F = 0.1639, num df = 29, denom df = 29, p-value = 5.352e-06
alternative hypothesis: true ratio of variances is not
equal to 1
95 percent confidence interval:
 0.07801875 0.34438909
sample estimates:
ratio of variances
         0.1639171
```

(b) F-test between Closed and Open

```
F test to compare two variances

data:  efficiency$Templates and efficiency$Open
F = 0.1278, num df = 29, denom df = 29, p-value = 3.398e-07
alternative hypothesis: true ratio of variances is not
equal to 1
95 percent confidence interval:
 0.06084976 0.26860201
sample estimates:
ratio of variances
         0.1278451
```

(c) F-test between Templates and Open

Figure 5.2: Efficiency equal variance tests

```
Kruskal-Wallis rank sum test

data:  Efficiency by Case
Kruskal-Wallis chi-squared = 24.1394, df = 2, p-value = 5.73e-06
```

Figure 5.3: Efficiency Kruskal-Wallis rank sum test

this caused some errors, participants understood the difference between events and states after an explanation about the error that they made. Table 5.2 shows a summary of the accuracy for all the configuration cases. All configuration cases have the same median value but different mean values.

	Closed	**Templates**	**Open**
Min.	3.0	5.0	3.0
1st Qu.	5.0	5.0	3.0
Median	5.0	5.0	5.0
Mean	4.9	5.0	4.2
3rd Qu.	5.0	5.0	5.0
Max.	5.0	5.0	5.0

Table 5.2: Accuracy summary (in a score on a scale of 5)

To know if the accuracy between configuration cases is equal or different we perform a Kruskal-Wallis rank sum test. The result of this test shown in Figure 5.4 rejects the null hypothesis that the groups are equal. We can conclude that there is a significant difference in accuracy between the configuration cases.

```
Kruskal-Wallis rank sum test

data:  Accuracy by Case
Kruskal-Wallis chi-squared = 26.0062, df = 2, p-value = 2.253e-06
```

Figure 5.4: Accuracy Kruskal-Wallis rank sum test

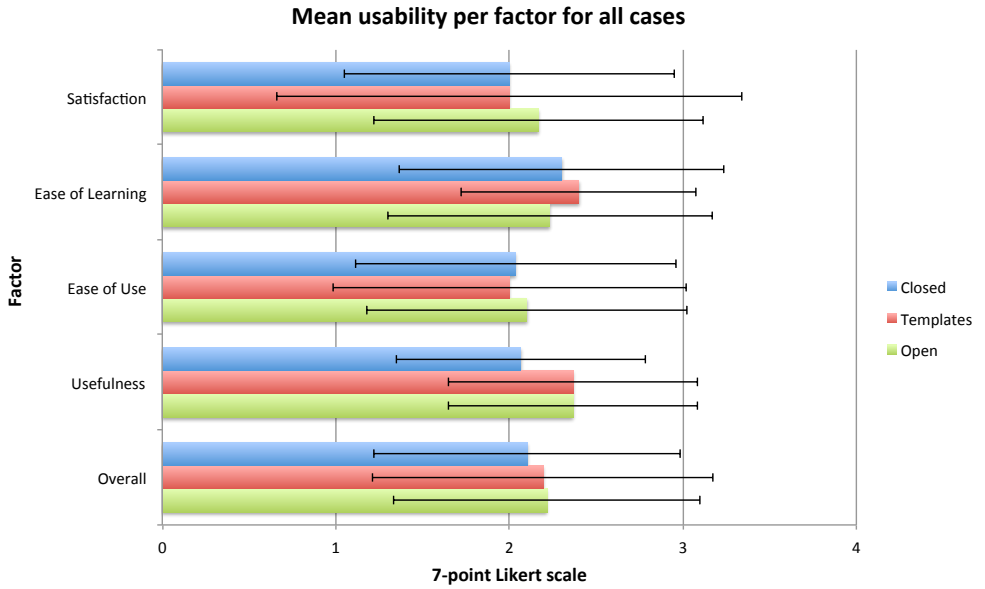


Figure 5.5: Usability means per factor barplot

5.2 Usability

The second important aspect of our evaluation is the usability of all the configuration cases. The usability can be split into four factors: satisfaction, ease of learning, ease of use and usefulness. For every factor the participant answered two questions based on a 7-point Likert scale ranging from *strongly disagree* to *strongly agree*. This will allow us to answer the second research question.

Research Question 2: *Is there a difference in usability between the configuration cases?*

The answers were really close between the three configuration cases. Figure 5.5 shows the mean score results of every factor and the overall usability mean score. Only positive values of the Likert scale are displayed as all the results were positive. Participants were more satisfied with the open configuration. The half-open configuration was the easiest to learn. Ease of use was almost equal between all the cases. Participants found the closed configuration less useful than the half-open and open configurations. Finally, all these factors together result in a mostly equal overall usability score. This equality is also reflected in the usability boxplots in Figure 5.6.

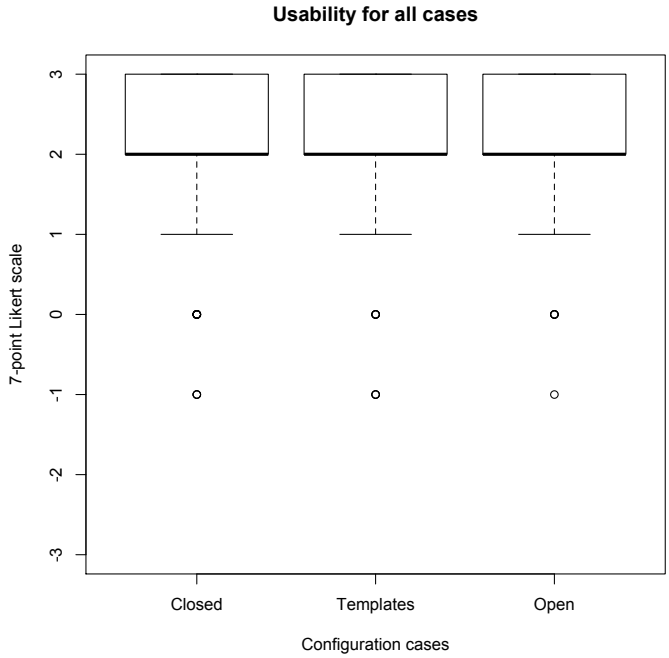


Figure 5.6: Overall usability boxplot

To confirm this equality a Kruskal-Wallis rank sum test is needed. The results of this test shown in Figure 5.7 does not reject the null hypothesis that the groups are equal. We can conclude that there is no significant difference in usability between the configuration cases.

```
Kruskal-Wallis rank sum test

data:  Score by Case
Kruskal-Wallis chi-squared = 0.4952, df = 2, p-value = 0.7807
```

Figure 5.7: Overall usability Kruskal-Wallis rank sum test

5.3 Influence

We found out that there is a significant difference in efficiency and accuracy, while on the other hand there is no significant difference in usability. Based on this findings we can answer the third research question.

Research Question 3: *Does the efficiency and accuracy of the configuration cases have an influence on their usability?*

The participants did not find the rule-creation process less usable depending on which configuration case was used. If there was an influence we would see a decrease in usability because of the decreasing efficiency and accuracy going from the closed configuration to the half-open and open configuration. We can conclude that there is no influence from the efficiency and accuracy on the usability of the configuration cases.

5.4 Other Findings

The evaluation also contained a user experience survey and a top three of the configuration methods. This information will tell us how the participants experienced the proposed design, through different factors, and what their preferred configuration method is.

5.4.1 User Experience

A user experience survey based on a 7-point Likert scale allows us to analyse multiple user experience factors, namely dependability, stimulation, novelty, perspicuity and attractiveness. The results were mostly positive and are shown in Figure 5.8 through the mean score of each factor and the overall user experience mean score along with error bars that represent one standard deviation. The dependability mean score was the highest, which means that the rule-creation process worked almost exactly like the participant expected. The lowest, but still positive, mean score is perspicuity. Participants were mostly neutral about the difficulty of the design.

5.4.2 Top 3

Participants were asked their top 3 of the configuration methods at the end of the evaluation. The results can be seen in Figure 5.9. A majority of participants, 9 out of 15, chose the open configuration above the half-open configuration and the closed configuration on the third place. 4 Participants decided to put the half-open configuration above the open configuration. In general participants

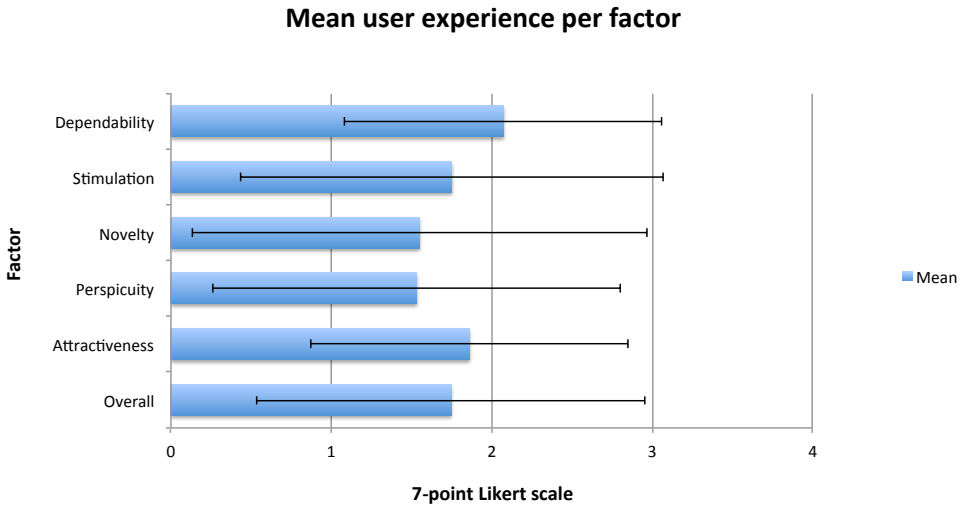


Figure 5.8: User experience

felt too restricted with the closed configuration. Finally, the idea of having both the open and half-open configuration options in the same application was often mentioned.

5.5 Design Implications

Based on the information that resulted from our evaluation, we describe three different design implications or improvements that can address the shortcomings discovered in our system.

Design Implication 1: *Offer templates by default along with a possibility to switch to an open configuration method.*

First, we can focus on the proposed configuration methods in the system. Participants were the most at ease and showed perfect accuracy with the half-open configuration method, which uses templates to create and edit rules. Still, a majority of the participants expressed their interest in the open configuration method, because of the limitations of templates. We propose two possible solutions to offer the best of both worlds to the user. A simple solution is to allow the user to switch between the half-open and open configuration methods through, for example, a "Switch" setting in the settings interface. A more advanced solution would be to extend templates in such a way that the user can start with a template and eventually continue to create the rule using the open configuration method. This could be achieved by adding an '*Extend*' button in

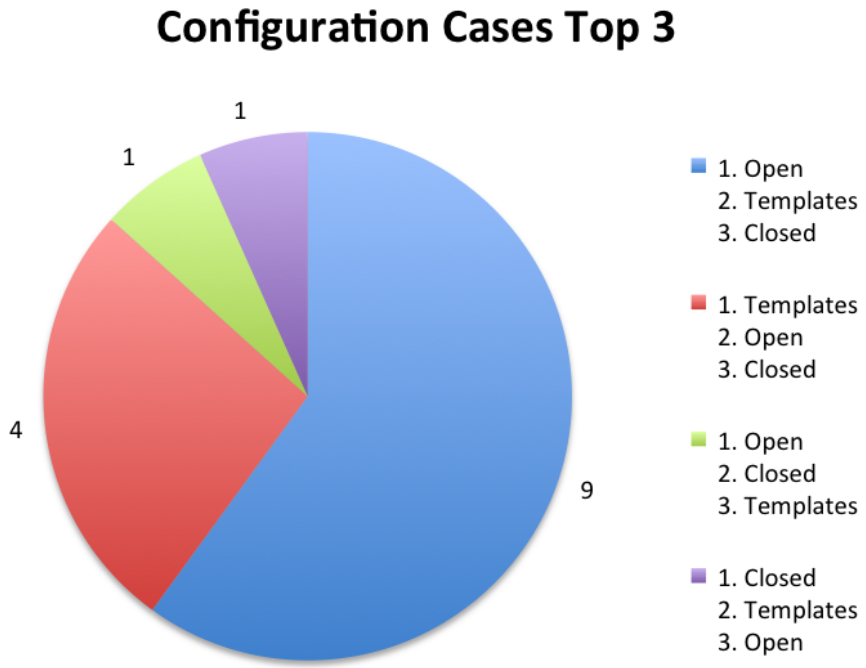
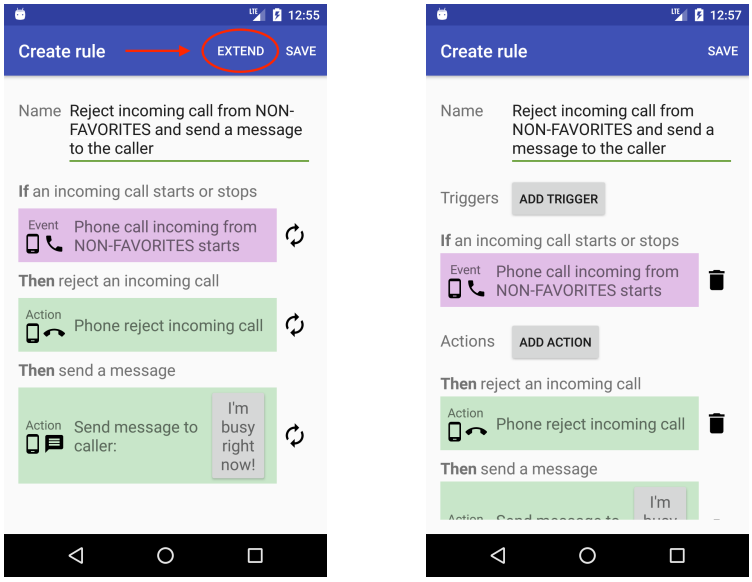


Figure 5.9: Participants' top 3 of cases



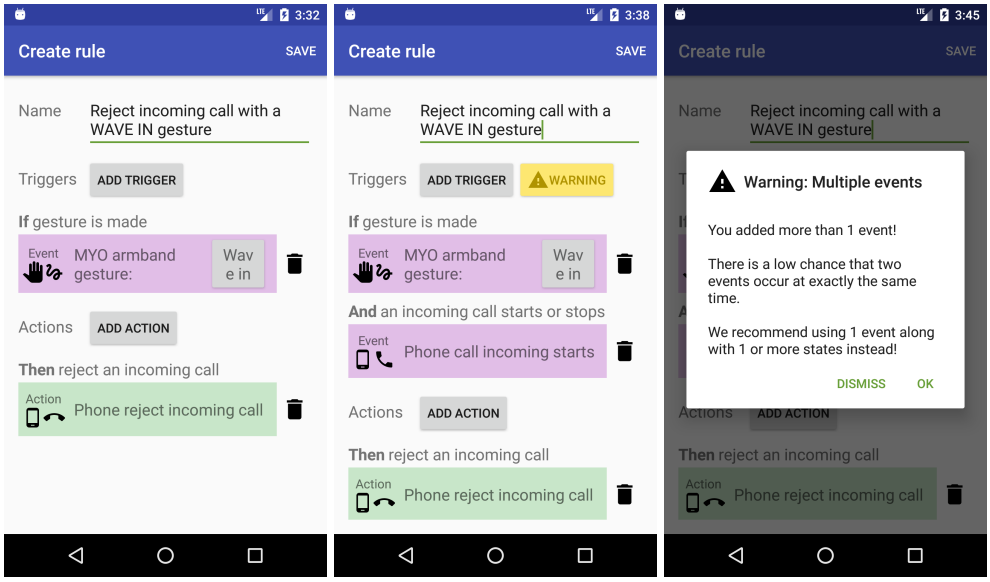
(a) Template rule-creation (b) Open rule-creation

Figure 5.10: Extending a rule template

the rule-creation interface, which would allow the user to continue to create the rule in an open way with the elements of the template as shown in Figure 5.10. If this advanced solution is applied, one should also introduce or highlight this functionality to the user.

Design Implication 2: *Assist the user with rule-logic during the rule-creation process.*

An important aspect of the rule-creation process is the logical thinking that is needed to select and combine the correct events, states and actions in order to achieve the wanted effect. With the closed and half-open configuration methods participants required few to no logical thinking as the rules were completely predefined in the form of example rules or structurally predefined with rule templates. On the other hand, the open configuration method showed some troubles from the participants related to the logical thinking needed to combine events, states and actions correctly. For example, multiple participants made the error of combining the "*incoming call starts ringing*" event with the "*gesture made*" event to trigger a rule, which is practically impossible to occur at the same time. Instead, the participant has to use the "*incoming call is ringing*" state along with the "*gesture made*" event. Another interesting observation was made when a participant thought that an event was not required because it could be logically derived from the "*reject call*" action that a call is currently incoming.



(a) Without notification (b) With notification (c) Warning popup

Figure 5.11: Warning notification

By implementing an intelligent assistant or logic checker the user could be better informed and assisted during the rule-creation process, which will result in better created rules. The interface could be improved by prompting the user with an informational popup window when a logical error is detected or a logical suggestion is available. If this is too obtrusive for the user, the user could be notified with a kind of notification on which the user can press to open a popup. Figure 5.11 shows a warning that appears when the user adds more than one event to a rule. This warning is an instruction that lets the user decide whether or not he applies it. Other types of notifications could be used, such as a blue '*Help*' or red '*Error*' notification.

Design Implication 3: *Provide an intuitive search functionality in order to find templates, events, states and actions more easily.*

This last design implication targets the ease of use of the design. During the evaluation participants barely or did not use the search functionality. Even if our design only offers text-based search, we believe it does enhance the rule-creation process. The fact that it was not used often is because our design showed a list of all the possibilities and the amount of elements the participant could choose from was very limited. When the amount of data in the system will be greater, the user will more likely have to use a search functionality.

The lists from which a user can choose should stay small. One should aim showing lists with ten or less elements. An improvement to our design should reduce these too long lists and help users to find rule templates, events, states and actions more easily. All these elements should be categorised and grouped. For example, rule templates could be grouped by scenarios or devices such as "*Driving a car*", "*Sport*" or "*Pebble Watch*".

5.6 Summary

The evaluation has demonstrated that participants are more interested in the half-open and open configurations. This evaluation highlighted results in the learning phase. We believe that the efficiency and accuracy overhead of the open configuration method decreases and equals that of the other configuration methods after the learning phase. From the 15 participants 14 would use a final version of this system. Home automation was the main reason to use this system, but participants were also interested to use and configure wearables with it.

Chapter 6

Discussion and Conclusion

We would like to conclude with this last chapter by discussing the contributions of this thesis. Additionally, we provide our thoughts on how the proposed framework can be enhanced and what is possible in the future to provide an optimal trigger-action programming framework for configurable wearables.

6.1 Discussion and Contributions

With our literature study we explored the current state of end-user development and wearables. End-User Development (EUD) has been around and researched for decades, which means that it is now at an advanced state with a lot of well-developed methodologies. We discussed four of them, namely *Natural Language Programming (NLP)*, *Programming-by-Demonstration (PbD)*, *Visual Programming (VP)* and *Trigger-Action Programming (TAP)*. On the other hand, research about wearables is much more recent. We gave an overview of what they are, their design, how they affect the user's attention, their role in the health and fitness sector, how they contribute to a smart environment and finally how they can be used for novel interactions. For this thesis we decided to bring both topics together and presented a framework based on TAP to allow end users to configure and personalise their wearables.

We designed a framework called Triggered that gives the end user more control over their devices and wearables. The devices' and wearables' functionality can be used as triggers and actions to create rules for automating tasks. Such automation systems already exist in the consumer world, for example the online platform IFTTT. But these existing systems are often focussed on using only one event to trigger one action and simply connecting devices and web services to automate simple tasks. For example, with IFTTT[1] one can create a rule to change the background image of their smartphone every time they upload

[1] `https://www.ifttt.com` [Accessed 22 July 2017]

a picture on Instagram[2]. Triggered differs in two ways. First, we allow users to use multiple triggers (events and/or states) and multiple actions. Second, while existing systems mainly use devices as input for rules, our framework allows the end user to manage their devices and wearables and use those as output as well through actions in rules. This way the end user can decide how a device or wearable should work in specific scenarios. For example, the end user could create multiple rules to manage when and how their Pebble Watch changes watchmodes or even enable remote alarm control using gestures.

We proposed three configuration methods to create rules for configuring devices and wearables. To create rules with the *closed configuration* the user can choose from a list of predefined rules and eventually adjust some values. The *half-open configuration interface* lets the user choose from a list of templates that predefine the skeleton of a rule using event-types, state-types and action-types. Finally, the *open configuration* gives the user complete freedom to build rules from scratch by adding and deleting events, states and actions as much as they want to create very specific rules.

We did an evaluation to know how the user experiences these configuration methods and thus which one, or a combination of which ones, we should use in our Triggered framework. In this evaluation fifteen participants made two rules with each configuration method. These rules required triggers and actions from multiple devices to automate tasks in specific scenarios. For each rule, we recorded the rule-creation process, timed it and asked the participant about the usability. In the end, the participants shared their experience, their top three of the configuration methods and general remarks.

All this data allowed us to analyse statistically if there were differences between the configuration methods. We discovered that efficiency and accuracy is significantly different between the configuration methods. The closed configuration scored better than the half-open and open configurations, but they all had good efficiency and accuracy. The usability was not significantly different. This confirmed our expectations that users are fine adapting to a more difficult environment for creating rules if that gives them more freedom. A majority of the participants preferred the open configuration, even though it had the steepest learning curve. Participants felt confident using templates, but felt too restricted with the predefined rules from the closed configuration.

Finally we introduced three design implications that can be applied on our framework. By combining the half-open configuration with the open configuration the users can create rules the best way that fits them. One can start creating a rule with a template and decide to extend it using the open configuration method. To assist the user during the rule-creation process we recommend

[2] `https://www.instagram.com` [Accessed 22 July 2017]

using notifications. Finally, to avoid getting lost in too long lists and guiding the user during their search process we recommend to categorise items and offer an advanced search and filter interface.

6.2 Future Work

We mentioned earlier that our framework was a foundation of our idea to offer configurable wearables to the end user. We build an application with a limited amount of devices and functionality that shows how our framework can be used. We quickly experienced that within this thesis, that is limited in time and resources, a completely finished framework would not be achievable. We decided to focus on what would be the best way to create rules for the end user by proposing and evaluating three possible configuration methods. We believe this was the first important step towards configurable wearables as it is important that the end user feels comfortable during the rule-creation process.

During the design and evaluation phases we learned a lot on how we could improve our framework in the future. We divide these findings and ideas for future work into three categories. First, our framework and rule engine could benefit from Artificial Intelligence (AI). Second, we wonder how devices can be added to our framework by manufacturers and end users. Finally, we discuss how the Triggered application could be enhanced.

6.2.1 Artificial Intelligence

We live in a connected world that is getting smarter in a lot of ways. As we briefly discussed in Subsection 2.1.4, the Internet of Things (IoT) and smart devices are more and more present in our daily lives. The evolution of computer science has changed how people live and more specifically how they absorb and consume information. This growth in available information is beneficial for using AI techniques in order to analyse the user. Based on how they consume, react to and produce new information one can build user profiles. Using these user profiles our framework could adapt itself to the user's needs and interests. This could, for example, be useful to guide the end user by proposing interesting templates for that specific user during the rule-creation process or to keep the user updated on possible rules based on their devices and interests.

AI could also be applied in the form of an assistant or logic-checker during the rule-creation process. We already introduced a part of this idea with Design Implication 2 in Section 5.5. For now our framework does not check if rules are logically correct. One could, for example, create a rule with an action to dismiss an alarm. This rule would not be logically correct if there is not a state about a ringing alarm, because one should only dismiss an alarm if it is actually ringing. During the evaluation we observed that users had troubles making the correct

rule, not because of how a rule must be made but rather which combination of events, states and actions a rule must contain. A logic-checker or assistant could reduce insecurities of the end user or even help the user to make even better rules.

6.2.2 Devices and Wearables

For now we added support for a limited amount of devices, which was enough to show and test our framework. In the future the end user should be able to add devices to the Triggered application to use those in new rules. Ideally a standard would exist for defining what devices are and what functionality they offer in order to analyse and automatically support newly added devices. Once that our framework knows what the device is and does it can generate events, states and actions that can be used in rules. Unfortunately, we have to find another way of adding devices because of the fact that such a standard is very difficult to achieve by manufacturers in the competitive consumer world. One could try to achieve this standard within the open-source world, but open-source devices are often less interesting and offer less functionality than closed-source devices.

On the other hand, manufactures that do allow developers to use their devices could make their future products more configurable. For example, we used the developer-friendly MYO gesture-recognition armband. The manufacturer of the MYO offers an SDK for developers, which determines its functionality. A next iteration of the MYO could, for example, be more configurable in terms of controlling the lights of the MYO or offering new vibration patterns.

6.2.3 Triggered Application

The Triggered application on the Android smartphone could also be improved on a lot of aspects. First of all, the user interface could be enhanced. With Design Implication 3 in Section 5.5 we already mentioned that adding an advanced search and filter is needed. Next, for now one can only create stand-alone rules. By stand-alone we mean that a rule is not connected or related to other rules. We believe that in the future rules could, for example, be grouped in the application. The end user could group them manually or the application could do that automatically. Groups of rules could define how their elements are related. This concept of groups could bring new functionality to the application. For example, one could "*chain*" the rules of a group in a certain order. For example, when one rule is executed the rest of the chain is executed as well. Another possible improvement would be to incorporate a social community in the application. One could create a new rule and share it with friends or export it as a new template that is shared with everyone.

6.3 Conclusion

The research in this thesis has shown how trigger-action programming can be used to empower end users to configure how their devices and wearables work. The Triggered framework demonstrated that end users are able to create rules to automate tasks in their life and personalise their devices. We did an evaluation and analysis with three different configuration methods to create rules, which resulted in three design implications that can improve our framework. The reactions from our participants confirmed that there is an interest from the public to use such a framework. We believe that there is a lot of room for improvements and new ideas for future work on the Triggered framework.

Bibliography

[1] Robin Abraham, Margaret Burnett, and Martin Erwig. Spreadsheet Programming. *Wiley Encyclopedia of Computer Science and Engineering*, 2008.

[2] Reem Albaghli and Kenneth M. Anderson. A Vision for Heart Rate Health Through Wearables. In *Proceedings of UbiComp 2016, ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct*, pages 1101–1105, Heidelberg, Germany, September 2016.

[3] Shaikh Shawon Arefin Shimon, Courtney Lutton, Zichun Xu, Sarah Morrison-Smith, Christina Boucher, and Jaime Ruiz. Exploring Non-touchscreen Gestures for Smartwatches. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 3822–3833, San Jose, USA, May 2016.

[4] Gilles Bailly, Jörg Müller, Michael Rohs, Daniel Wigdor, and Sven Kratz. ShoeSense: A New Perspective on Hand Gestures and Wearable Applications. In *Proceedings of CHI 2012, ACM Conference on Human Factors in Computing Systems*, pages 1239–1248, Austin, USA, May 2012.

[5] Yannick Bernaerts, Matthias Druwé, Sebastiaan Steensels, Jo Vermeulen, and Johannes Schöning. The Office Smartwatch: Development and Design of a Smartwatch App to Digitally Augment Interactions in an Office Environment. In *Proceedings of DIS 2014, Companion Publication on Designing Interactive Systems*, pages 41–44, Vancouver, Canada, June 2014.

[6] Margaret M Burnett. Visual Programming. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 1999.

[7] Keith A Butler, Robert J K Jacob, and Bonnie E John. Human-Computer Interaction: Introduction and Overview. In *Proceedings of CHI 1998, ACM Conference on Human Factors in Computing Systems*, pages 105–106, Los Angeles, USA, April 1998.

[8] Xiang'Anthony' Chen, Tovi Grossman, Daniel J Wigdor, and George Fitzmaurice. Duet: Exploring Joint Interactions on a Smart Phone and a Smart Watch. In *Proceedings of CHI 2014, ACM Conference on Human Factors in Computing Systems*, pages 159–168, Toronto, Canada, April 2014.

[9] Pei-Yu (Peggy) Chi and Yang Li. Weave: Scripting Cross-Device Wearable Interaction. In *Proceedings of CHI 2015, ACM Conference on Human Factors in Computing Systems*, pages 3923–3932, Seoul, South Korea, April 2015.

[10] Allen Cypher and Daniel Conrad Halbert. *Watch What I Do: Programming by Demonstration*. The MIT Press, 1993.

[11] Jose Danado and Fabio Paternò. Puzzle: A Mobile Application Development Environment Using a Jigsaw metaphor. *Journal of Visual Languages & Computing*, 25(4):297–315, August 2014.

[12] Bertrand David, Yun Zhou, Tao Xu, and René Chalon. Mobile User Interfaces and their Utilization in a Smart City. In *Proceedings of ICOMP 2011, International Conference on Internet Computing*, pages 383–388, Las Vegas, USA, July 2011.

[13] Luigi De Russis and Fulvio Corno. Homerules: A Tangible End-User Programming Interface for Smart Homes. In *Proceedings of CHI 2015, ACM Conference on Human Factors in Computing Systems*, pages 2109–2114, Seoul, South Korea, April 2015.

[14] Saul Delabrida, Thiago D'Angelo, Ricardo A R Oliveira, and Antonio A F Loureiro. Building Wearables for Geology: An Operating System Approach. *ACM SIGOPS Operating Systems Review*, 50(1):31–45, March 2016.

[15] Laura Devendorf, Kimiko Ryokai, Joanne Lo, Noura Howell, Jung Lin Lee, Nan-wei Gong, M Emre Karagozler, Shiho Fukuhara, Ivan Poupyrev, and Eric Paulos. "I don't want to wear a screen": Probing Perceptions of and Possibilities for Dynamic Displays on Clothing. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 6028–6039, San Jose, USA, May 2016.

[16] Anind K. Dey, R. Hamid, C. Beckmann, I. Li, and D. Hsu. a CAPpella: Programming by Demonstration of Context-Aware Applications. In *Proceedings of CHI 2004, ACM Conference on Human Factors in Computing Systems*, pages 33–40, Vienna, Austria, April 2004.

[17] Anind K. Dey, Timothy Sohn, Sara Streng, and Justin Kodama. iCAP: Interactive Prototyping of Context-Aware Applications. In *Proceedings of PERVASIVE 2006, International Conference on Pervasive Computing*, pages 254–271, Dublin, Ireland, May 2006.

[18] Gerhard Fischer, Elisa Giaccardi, Yunwen Ye, Alistair G Sutcliffe, and Nikolay Mehandjiev. Meta-Design: a Manifesto for End-User Development. *Communications of the ACM*, 47(9):33–37, September 2004.

[19] Brian J Fogg. Persuasive Technology: Using Computers to Change What We Think and Do. *Ubiquity*, pages 89–120, December 2002.

[20] Thomas Fritz, Elaine M Huang, Gail C Murphy, and Thomas Zimmermann. Persuasive Technology in the Real World: A Study of Long-Term Use of Activity Sensing Devices for Fitness. In *Proceedings of CHI 2014, ACM Conference on Human Factors in Computing Systems*, pages 487–496, Toronto, Canada, April 2014.

[21] Mitchell Gordon, Tom Ouyang, and Shumin Zhai. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 3817–3821, San Jose, USA, May 2016.

[22] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. *Future Generation Computer Systems*, 29(7):1645–1660, September 2013.

[23] Bob Hardian, Jadwiga Indulska, and Karen Henricksen. Balancing Autonomy and User Control in Context-Aware Systems - a Survey. In *Proceedings of PerCom 2006, IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 51–56, Pisa, Italy, March 2006.

[24] Chris Harrison, Desney Tan, and Dan Morris. Skinput: Appropriating the Body as an Input Surface. In *Proceedings of CHI 2010, ACM Conference on Human Factors in Computing Systems*, pages 453–462, Atlanta, USA, April 2010.

[25] Seth Holloway and Christine Julien. The Case for End-User Programming of Ubiquitous Computing Environments. In *Proceedings of FoSER 2010, FSE/SDP Workshop on Future of Software Engineering Research*, pages 167–172, Santa Fe, USA, November 2010.

[26] Steven Houben and Nicolai Marquardt. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of CHI 2015, ACM Conference on Human Factors in Computing Systems*, pages 1247–1256, Seoul, South Korea, April 2015.

[27] Justin Huang and Maya Cakmak. Supporting Mental Model Accuracy in Trigger-Action Programming. In *Proceedings of UbiComp 2015, ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 215–225, Osaka, Japan, September 2015.

[28] Jan Humble, Andy Crabtree, Terry Hemmings, Karl-Petter Åkesson, Boriana Koleva, Tom Rodden, and Pär Hansson. "Playing with the Bits" User-configuration of Ubiquitous Domestic Environments. In *Proceedings of UbiComp 2003, International Conference on Ubiquitous Computing*, pages 256–263, Seattle, USA, October 2003.

[29] Alia Ibrahim. *Framework for High Level Programming of Wireless Sensor and Actuator Networks*. PhD thesis, University of Surrey, August 2015.

[30] Ken Kawamoto, Takeshi Tanaka, and Hiroyuki Kuriyama. Your Activity Tracker Knows When You Quit Smoking. In *Proceedings of ISWC 2014, International Symposium on Wearable Computers*, pages 107–110, Seattle, USA, September 2014.

[31] Majeed Kazemitabaar, Liang He, Katie Wang, Chloe Aloimonos, Tony Cheng, and Jon E. Froehlich. ReWear: Early Explorations of a Modular Wearable Construction Kit for Young Children. In *Proceedings of CHI EA 2016, ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2072–2080, San Jose, USA, May 2016.

[32] N D Lane, S Bhattacharya, P Georgiev, C Forlivesi, and F Kawsar. An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices. In *Proceedings of IoT-App 2015, International Workshop on Internet of Things towards Applications*, pages 7–12, Seoul, South Korea, November 2015.

[33] Kyungmin Lee, Jason Flinn, and Brian Noble. The Case for Operating System Management of User Attention. In *Proceedings of HotMobile 2015, International Workshop on Mobile Computing Systems and Applications*, pages 111–116, Santa Fe, USA, February 2015.

[34] Gilly Leshed, Eben M Haber, Tara Matthews, and Tessa Lau. CoScripter: Automating & Sharing How-To Knowledge in the Enterprise. In *Proceedings of CHI 2008, ACM Conference on Human Factors in Computing Systems*, pages 1719–1728, Florence, Italy, April 2008.

[35] Henry Lieberman, Fabio Paternò, Markus Klann, and Volker Wulf. End-User Development: An Emerging Paradigm. In *End User Development*, pages 1–8. Springer Netherlands, 2006.

[36] James Lin, Jeffrey Wong, Jeffrey Nichols, Allen Cypher, and Tessa A Lau. End-User Programming of Mashups with Vegemite. In *Proceedings of IUI 2009, International Conference on Intelligent User Interfaces*, pages 97–106, Sanibel Island, USA, February 2009.

[37] Hugo Liu and Henry Lieberman. Programmatic Semantics for Natural Language Interfaces. In *Proceedings of CHI EA 2005, ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pages 1597–1600, Portland, USA, April 2005.

[38] Byron Lowens, Vivian Motti, and Kelly Caine. Design Recommendations to Improve the User Interaction with Wrist Worn Devices. In *Proceedings of PerCom 2015, IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 562–567, St. Louis, USA, March 2015.

[39] Simon Mayer, Nadine Inhelder, Ruben Verborgh, Rik Van De Walle, and Friedemann Mattern. Configuration of Smart Environments Made Simple: Combining Visual Modeling with Semantic Metadata and Reasoning. In *Proceedings of IOT 2014, International Conference on the Internet of Things*, pages 61–66, Cambridge, USA, October 2014.

[40] Abhinav Mehrotra, Mirco Musolesi, Robert Hendley, and Veljko Pejovic. Designing Content-driven Intelligent Notification Mechanisms for Mobile Applications. In *Proceedings of UbiComp 2015, ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 813–824, Osaka, Japan, September 2015.

[41] Chulhong Min, Seungwoo Kang, Chungkuk Yoo, Jeehoon Cha, Sangwon Choi, Younghan Oh, and Junehwa Song. Exploring Current Practices for Battery Use and Management of Smartwatches. In *Proceedings of ISWC 2015, ACM International Symposium on Wearable Computers*, pages 11–18, Osaka, Japan, September 2015.

[42] Pranav Mistry, Pattie Maes, and Liyan Chang. WUW - Wear Ur World - A Wearable Gestural Interface. In *Proceedings of CHI EA 2009, ACM Extended Abstracts on Human Factors in Computing Systems*, pages 4111–4116, Boston, USA, April 2009.

[43] Vivian Genaro Motti and Kelly Caine. Micro Interactions and Multi Dimensional Graphical User Interfaces in the Design of Wrist Worn Wear-

ables. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 59(1):1712–1716, 2015.

[44] Vivian Genaro Motti and Kelly Caine. Smart Wearables or Dumb Wearables? Understanding How Context Impacts the UX in Wrist Worn Interaction. In *Proceedings of SIGDOC 2016, ACM International Conference on the Design of Communication*, Silver Spring, USA, September 2016.

[45] Brad A Myers, Andrew J Ko, and Margaret M Burnett. Invited Research Overview: End-User Programming. In *Proceedings of CHI EA 2006, ACM Extended Abstracts on Human Factors in Computing Systems*, pages 75–80, Montréal, Canada, April 2006.

[46] Brad A Myers, John F Pane, and Andy Ko. Natural Programming Languages and Environments. *Communications of the ACM*, 47(9):47–52, September 2004.

[47] Michael Nebeling, Stefania Leone, and Moira C Norrie. Crowdsourced Web Engineering and Design. In *Proceedings of ICWE 2012, International Conference on Web Engineering*, pages 31–45, Berlin, Germany, July 2012.

[48] Michael Nebeling, Alexandra To, Anhong Guo, Adrian A de Freitas, Jaime Teevan, Steven P Dow, and Jeffrey P Bigham. WearWrite: Crowd-Assisted Writing from Smartwatches. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 3834–3846, San Jose, USA, May 2016.

[49] Mark W. Newman, Jana Z. Sedivy, W. Keith Edwards, Trevor F. Smith, Karen Marcelo, Christine M. Neuwirth, Jason I. Hong, and Shahram Izadi. Designing for Serendipity: Supporting End-User Configuration of Ubiquitous Computing Environments. In *Proceedings of DIS 2002, Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, pages 147–156, London, England, June 2002.

[50] Ian Oakley and Doyoung Lee. Interaction on the Edge: Offset Sensing for Small Devices. In *Proceedings of CHI 2014, ACM Conference on Human Factors in Computing Systems*, pages 169–178, Toronto, Canada, April 2014.

[51] Fabio Paternò. End-User Development: Survey of an Emerging Field for Empowering People. *ISRN Software Engineering*, 2013(532659):1–11, 2013.

[52] Lukasz Piwek, David A. Ellis, Sally Andrews, and Adam Joinson. The Rise of Consumer Health Wearables: Promises and Barriers. *PLoS Medicine*, 13(2):1–9, 2016.

[53] Stefania Pizza, Barry Brown, Donald McMillan, and Airi Lampinen. On Time: The Smartwatch in Vivo. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 5456–5469, San Jose, USA, May 2016.

[54] Reza Rawassizadeh, Blaine A Price, and Marian Petre. Wearables: Has the Age of Smartwatches Finally Arrived? *Communications of the ACM*, 58(1):45–47, January 2015.

[55] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Others. Scratch: Programming for All. *Communications of the ACM*, 52(11):60–67, November 2009.

[56] Konrad Tollmar, Frank Bentley, Cristobal Viedma, and I L Libertyville. Mobile Health Mashups. In *Proceedings of PervasiveHealth 2012, International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, pages 65–72, San Diego, USA, May 2012.

[57] Sandra Trullemans and Beat Signer. A Multi-layered Context Modelling Approach for End Users, Expert Users and Programmers. *CEUR Workshop Proceedings*, 1602:36–41, 2016.

[58] Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, and Michael L. Littman. Practical Trigger-Action Programming in the Smart Home. In *Proceedings of CHI 2014, ACM Conference on Human Factors in Computing Systems*, pages 803–812, Toronto, Canada, April 2014.

[59] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L Littman. Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 3227–3231, San Jose, USA, May 2016.

[60] Martin Weigel, Tong Lu, Gilles Bailly, Antti Oulasvirta, Carmel Majidi, and Jürgen Steimle. iSkin: Flexible, Stretchable and Visually Customizable On-Body Touch Sensors for Mobile Computing. In *Proceedings of CHI 2015, ACM Conference on Human Factors in Computing Systems*, pages 2991–3000, Seoul, South Korea, April 2015.

[61] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.

[62] Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. Serendipity: Finger Gesture Recognition using an Off-the-Shelf Smartwatch. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 3847–3851, San Jose, USA, May 2016.

[63] Jeffrey Wong and Jason I Hong. Making Mashups with Marmite: Towards End-User Programming for the Web. In *Proceedings of CHI 2007, ACM Conference on Human Factors in Computing Systems*, pages 1435–1444, San Jose, USA, April 2007.

[64] Yang Zhang, Junhan Zhou, Gierad Laput, and Chris Harrison. SkinTrack: Using the Body as an Electrical Waveguide for Continuous Finger Tracking on the Skin. In *Proceedings of CHI 2016, ACM Conference on Human Factors in Computing Systems*, pages 1491–1503, San Jose, USA, May 2016.

[65] John Zimmerman, Erik Stolterman, and Jodi Forlizzi. An Analysis and Critique of Research Through Design: Towards a Formalization of a Research Approach. In *Proceedings of DIS 2010, ACM Conference on Designing Interactive Systems*, pages 310–319, Aarhus, Denmark, August 2010.

Appendix A

Evaluation Form

Carraggi Nicolas
Master Computerwetenschappen
Vrije Universiteit Brussel

Making wearables fit the user: Evaluation of Triggered

1. General information

Participant ID: 3 Start time: End time:

Place of evaluation: Date of evaluation:

2. Profile

Name: Gender: Male Female Other
Age:
Education: Profession:

Do you have any experience with rule-based (automation) applications? Yes No

Do you have any experience with wearables? Yes No

3. Questions before

Do you agree that this evaluation is recorded?
* This will only be used to analyse the users' actions and reactions.
* This will not be used for anything other than this evaluation.
* This recording will not be shared.

Yes for video & audio | No for video Yes for audio | No

4. Results

	Time (seconds)	Accuracy (1-5)
Rule 1: T 1 a		
Rule 2: T 1 b		
Rule 3: C 3 b		
Rule 4: C 3 a		
Rule 5: O 2 a		
Rule 6: O 2 b		

5. Introduction

Triggered is an application, which can be used to automate tasks that combine devices (smartphones, wearables, etc.) and services.

For example, turn on the TV at home with a double tap gesture.
This task requires an event, a state and an action.
Making a double tap gesture is the event.
Being at home is the state.
Turning on the home TV is the action.

The triggers of a rule, events and states, determine when a rule is triggered.
When the rule is triggered, the actions will be executed.

6. Case 1

Case: Templates
Scenario: 1 a → b

6.1 Rules

Rule 1

– Before using rule –

	Strongly Uncertain	Uncertain	Somewhat Uncertain	Neutral	Somewhat Certain	Certain	Stronlgy Certain
How certain are you this rule will work?							

– After using rule –

	Strongly Unsatisfied	Unsatisfied	Somewhat Unsatisfied	Neutral	Somewhat Satisfied	Satisfied	Stronlgy Satisfied
How satisfied are you of the rule you made?							

Rule 2

– Before using rule –

	Strongly Uncertain	Uncertain	Somewhat Uncertain	Neutral	Somewhat Certain	Certain	Stronlgy Certain
How certain are you this rule will work?							

– After using rule –

	Strongly Unsatisfied	Unsatisfied	Somewhat Unsatisfied	Neutral	Somewhat Satisfied	Satisfied	Stronlgy Satisfied
How satisfied are you of the rule you made?							

6.2 Usability survey

Complete this survey, with the questions applied on HOW you created the rules.

	Strongly Disagree	Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Agree	Stronlgy Agree
It is useful to create rules this way.							
It does everything I would expect it to do.							
It is easy to use.							
I can recover from mistakes quickly and easily.							
I learned to use it quickly.							
I quickly became skillful with it.							
It works the way I want it to work.							
It is pleasant to use.							

7. Case 2

Case: Closed
Scenario: 3 b → a

7.1 Rules

Rule 3

– Before using rule –

	Strongly Uncertain	Uncertain	Somewhat Uncertain	Neutral	Somewhat Certain	Certain	Stronlgy Certain
How certain are you this rule will work?							

– After using rule –

	Strongly Unsatisfied	Unsatisfied	Somewhat Unsatisfied	Neutral	Somewhat Satisfied	Satisfied	Stronlgy Satisfied
How satisfied are you of the rule you made?							

Rule 4

– Before using rule –

	Strongly Uncertain	Uncertain	Somewhat Uncertain	Neutral	Somewhat Certain	Certain	Stronlgy Certain
How certain are you this rule will work?							

– After using rule –

	Strongly Unsatisfied	Unsatisfied	Somewhat Unsatisfied	Neutral	Somewhat Satisfied	Satisfied	Stronlgy Satisfied
How satisfied are you of the rule you made?							

7.2 Usability survey

Complete this survey, with the questions applied on HOW you created the rules.

	Strongly Disagree	Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Agree	Stronlgy Agree
It is useful to create rules this way.							
It does everything I would expect it to do.							
It is easy to use.							
I can recover from mistakes quickly and easily.							
I learned to use it quickly.							
I quickly became skillful with it.							
It works the way I want it to work.							
It is pleasant to use.							

8. Case 3

Case: Open
Scenario: 2 a → b

8.1 Rules

Rule 5

– Before using rule –

	Strongly Uncertain	Uncertain	Somewhat Uncertain	Neutral	Somewhat Certain	Certain	Stronlgy Certain
How certain are you this rule will work?							

– After using rule –

	Strongly Unsatisfied	Unsatisfied	Somewhat Unsatisfied	Neutral	Somewhat Satisfied	Satisfied	Stronlgy Satisfied
How satisfied are you of the rule you made?							

Rule 6

– Before using rule –

	Strongly Uncertain	Uncertain	Somewhat Uncertain	Neutral	Somewhat Certain	Certain	Stronlgy Certain
How certain are you this rule will work?							

– After using rule –

	Strongly Unsatisfied	Unsatisfied	Somewhat Unsatisfied	Neutral	Somewhat Satisfied	Satisfied	Stronlgy Satisfied
How satisfied are you of the rule you made?							

8.2 Usability survey

Complete this survey, with the questions applied on HOW you created the rules.

	Strongly Disagree	Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Agree	Stronlgy Agree
It is useful to create rules this way.							
It does everything I would expect it to do.							
It is easy to use.							
I can recover from mistakes quickly and easily.							
I learned to use it quickly.							
I quickly became skillful with it.							
It works the way I want it to work.							
It is pleasant to use.							

9. User experience

	1	2	3	4	5	6	7	
Annoying								Enjoyable
Not understandable								Understandable
Creative								Dull
Easy to learn								Difficult to learn
Valuable								Inferior
Boring								Exciting
Not interesting								Interesting
Unpredictable								Predictable
Inventive								Conventional
Obstructive								Supportive
Good								Bad
Complicated								Easy
Unlikable								Pleasing
Usual								Leading Edge
Unpleasant								Pleasant
Motivating								Demotivating
Meets expectations								Does not meet expectations
Clear								Confusing
Attractive								Unattractive
Friendly								Unfriendly
Conservative								Innovative

10. Comparison between cases

Top 3 of cases:

1)

2)

3)

Advantages and disadvantages of each case?

11. General remarks

Would you use a finished version of this application? Yes No

If you answered no, why wouldn't you use it?

If you answered yes, what would you like to use it for?

Do you have any other remarks or suggestions?