Simplifying interaction on mobile devices using context-aware computing

Jakob Berglund & Filip Lindqvist

2015

Master's Thesis

Department of Design Sciences
Lund University

Abstract

This thesis investigates how context-awareness and learning from user patterns could improve the user interaction for a smartphone application that can control the computer and home and how it could create a smarter more personal experience. It looks at both how a existing customers would react in three different experiments (a survey, interviews and a prototype of the application), and how non-users would like an application that could control their home with respect to context-awareness. In the experiment carried out on the non-users a web-based simulator was used to illustrate the different levels of awareness during an interview where the different scenarios were discussed and compared. It finds that that the existing customers were interested in creating a more personal experience by adding context-awareness and learning user patterns. In addition, it finds that a majority of the non-customers would like as much automation as possible.

Keywords: User-Experience, Adaptive User Interface, Context-Awareness, Handheld Computing

Acknowledgements

First and foremost, we would like to thank Accelerace DK and Sydenergi (SE) for the insights they help us gain during Next Step Challenge, Europe's most ambitious business accelerators for late startups. During the programme in Esbjerg (Denmark) they helped us gain insight of our business as a whole, but also supported this project with their experience within the technology market. The programme also helped us by providing interviews from the other startup companies within Next Step Challenge, which helped to differentiate across cultures.

In addition we would like to thank our fellow thesis project, "Product and strategy optimization with lean mobile analytics" by Jonathan Kjellsson, that helped us implement and use some of the analytics tools that were used in this project.

Furthermore, we would like to give a extra special thanks to our excellent inspiriting company supervisor at Unified Intents Philip Bergqvist, who helped us preform all the necessary tests and understand what the users were looking for. Lastly, we would like to thank our supervisor at LTH Johanna Persson, and Joakim Eriksson that also is our examiner, for all their guidance and wisdom.

Work distribution

The work was distributed equally between Jakob and Filip.

Contents

1 Introduction

1.1 Project background

Computer software is today becoming more complex than ever before. The performance and mobility of computers and mobile devices increases day by day. This generates a huge potential for more features, but creates the problems of presenting features to the user in an efficient manner.

1.2 Unified Remote

Unified Remote is a smartphone application that allows you to control certain applications on a computer. It is a cross-platform solution that allows you to control everything from media players to slide presentations. It also allows the phone to function as a remote touch pad and keyboard for the computer.

1.2.1 How it works

Unified Remote connects to a computer running a specific server software that can be downloaded from the company website[1]. The app and the server connect using Wifi or Bluetooth. The process for installing the product for a user looks as follows:

- The user downloads the app from an appstore (Play Store[2], App store[3], Windows Market [4]).
- The user starts the app and are told that they need to download the server software from the company website.
- The user installs the server software.

[1] https://www.unifiedremote.com

[2] https://play.google.com/store/apps/details?id=com.Relmtech.Remote&hl=en

[3] https://itunes.apple.com/us/app/unified-remote/id825534179?mt=8

[4] https://www.windowsphone.com/en-us/store/app/unified-remote/bf53969d-8078-4de5-9322-adda5cba4f87

- The app detects the server over Wifi or Bluetooth.
- The User is now able to control their computer from the app.

The user is presented with a list of different remote controls for different software. All these remote controllers are open-source and can be modified according to the users need.

1.2.2 Terminology

Remote
A remote is a view in the app that simulates a remote control that controls one or more programs on a computer see figure 1.1.

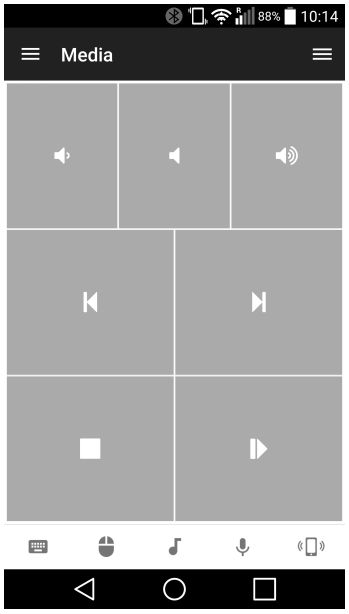


Figure 1.1: Media Remote in Unified Remote.

Action
An action performs something on the computer. A remote consists of a number of actions. Normally they are represented by buttons in the remote but they can be hidden from the user interface or be a part of a more complex control like a slider.

Power User
A Power User is a user that uses more features than the average user. They also have a greater understanding in how the application works and can solve

problems that arises themselves. Furthermore, they are interested in future development and in some cases try to help by reporting and investigating errors they find in the application.

1.2.3 Server/Client architecture

Unified Remote uses a server/client architecture just like a normal web browser. When the client is connected to the server, the client downloads all available remote controls from the server just like a web browser downloads web-pages from a web-server. When the user interacts with the application information about the interaction (so called actions) is sent to the server. The server then executes the actions based on a Lua [5] script file.

1.2.4 Current users

Currently the application has a very technical user-base consisting of mostly men in the ages between 18 and 34. There are a number of different personas that describe these users or at least what the company thought of the users before this project. The company believes, based on analytics data from application usage, that the normal user has a computer connected to a TV and uses the computer to play movies and control home entertainment. There is also a group of users that uses it to control presentations and different work related tasks.

The company is however working to expand into new areas like gaming and smart-homes. The issues that this project will investigate are mostly within smart-homes, but all test in the application will be on the current application that controls a computer.

1.3 Problem definition

This project will attempt to find away to improve the user experience of the classical on-demand remote control interaction metaphore. Striving to automate and simplify the user interface. In this process a number of question's rise to the surface.

RQ1 Is the user willing to give away control to an application in order to simplify their life?

RQ2 Do users want the application to create the patterns and understand the context or does the users want to create this themselves?

[5] http://www.lua.org/about.html

1.4 Project restrictions

This is a master thesis project executed by two students within the Faculty of Engineering (LTH) at Lund University, Sweden. The master thesis project spans one semester and is equal to 30 credits, which at 100% work-rate corresponds to a total of 20 weeks.

1.5 Structure

The structure of the report is based on the standardized IMRAD (Introduction – Method – Results – and – Discussion), with some minor additions. The current chapter introduces the circumstances, context and limits of this project. Following, the background chapter lays out the theory behind the problem addressed in this report. Further on, the method chapter explains the principles and tools used in the project. The result chapter introduces the intermediate results. The following chapter discusses the validation and correctness of the data, and investigates possible conclusions from these results. This is then summed up in the conclusion chapter that concludes the result of this project. In the last chapter, the possible future work is sketched.

2 Background and theory

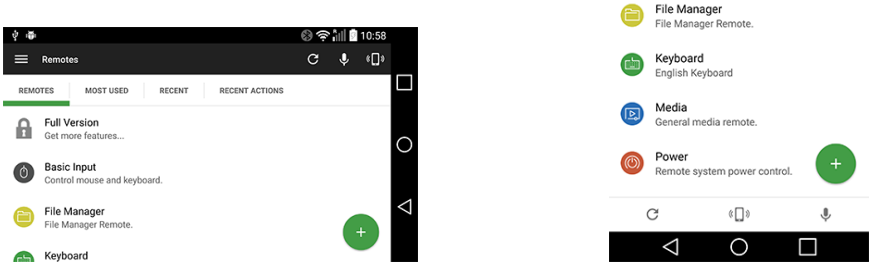
This section covers some of the background behind context and context-aware computing.

2.1 Context-aware computing

In the early days of the computer era, devices were stationary and designed for a specific purpose. Today the situation is quite different. Modern smartphones are the ultimate example of the complete opposite. Computers are being used in different environments and for different purposes, resulting in different contexts.

> *Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that considered relevant to the interaction between a user and an application, including the user and applications themselves.* - The definition of context et al. Anind K. Dey and Gregory D. Abowd [2]

Context-aware computing is a concept where information about the user and its context is taken into account when presenting information. It could be an orientation sensor in a smartphone that knows what direction the device is held and changes the interface based on this. If the smartphone were to be changed from a vertical position (figure 2.1b) to a horizontal position (figure 2.1a), the interface changes layout to better utilize the wider screen area. However, it could also be a more complex context that this. Modern smartphones have multiple sensors, such as location or temperature that could be used. As shown in figures 2.1b and 2.1a the current version of Unified Remote is context-aware in terms of device orientation. This behaviour is norm and is defined by the underlying operating system that provides this functionality by default.



(a) Landscape layout. **(b)** Portrait layout.

Figure 2.1: Basic context-aware layout changes

2.1.1 The taxonomy of context-aware features

Schilit, Adams and Want were three of the first to define context-aware computing in 1994 [9]. They define four categories of context; *Computing context* (connectivity, connection cost or reachable resources), *user context* (user profile, location, other users nearby or social situation), *physical context* (the physical environment, noise, light, traffic or temperature) and *time context* (time of day, weekday, season, year). Furthermore, storing the previous contexts generate an additional category *context history*. Context-aware software is defined as systems that adapt to the location of use, the collection of nearby people, hosts and other accessible devices. In addition to detecting changes over time to allow adaption to these inputs. In the paper by Schilit, Adams and Want, context-awareness features are split into four categories (see figure 2.1); proximate selection, automatic contextual reconfiguration, contextual information and commands, and context-triggered actions.

	Manual	Automatic
Information	Proximate selection	Automatic Contextual Reconfiguration
Commands	Contextual Commands	Context-triggered Actions

Table 2.1: The four categories of context-aware computing [9]

Proximate selection

If a user interface is aware of the location or distance to objects that are relevant to the user, it is possible to prioritise actions related to these objects. A good example of this is a yellow pages catalogue where businesses can be presented in order by distance from the user. The distance is probably very relevant to the user, and thus the interface should prioritize the nearest businesses. However, it is important that the distance really is of matter to the user if this ordering is done. The big struggle is to allow interfaces to show this prioritization in combination with for example alphabetical order.

Automatic contextual reconfiguration

Reconfiguration allows the software to add and remove components based on the current context. If a terminal is controlling two machines and one gets disconnected, then the component interface of that machine can be hidden on the terminal to signal to the user that the machine is offline. The struggle of this method is to not confuse the user if the context is changing often, and allow the user to recover from incorrectly reported context information.

Contextual information & commands

Users are often predictable in certain environments, which are often bound to specific actions. Tasks that are regularly done in the kitchen differ from those regularly done in the office. By remembering commands executed in different contexts, these can be showed as suggestions to the user. As an example, a launcher for applications can show office related programs and commands, like printing, when it is used during business hours and connected to the company network. While used in the kitchen, it can show applications for showing recipes and to-do lists, when it is connected to the home network.

Context-triggered actions

As mentioned, users are often predictable in a certain environment. Sometimes so predictable that it is possible to execute actions on the users behalf. This is the basis of context-triggered actions. Basic rules can be set up on an `IF-THEN` basis to automate certain tasks. This can be achieved either by users setting up their own rules, or by utilizing statistical learning.

2.1.2 Pascoe's taxonomy of features

In Pascoe (1998) another taxonomy of context-aware features is suggested[8]. The taxonomy is based on generic core capabilities without the dependency on application, function, or interface.

Contextual sensing

The lightest level of contextual awareness, were environmental states are detected and presented to the user. The perfect example would be a navigation application that locates the user using the GPS-system and places a marker on a map accordingly.

Contextual adaptation

This class involves tailoring the service to the user, based on the circumstances. Reconfiguring itself and adapting to fit the current user context. Examples of this would be a device adapting to the light level and adjusting the screen brightness accordingly. Within remote controlling this level could mean that a different set of controls are shown on daytime, as opposed to nighttime.

Contextual resource discovery

Contextual resource discovery uses information about the current context to discover and utilize resources within the same context. This could involve detecting a resource on a wireless network which could provide a context. A wireless home network is an example of where this could be utilized, to connect to different devices when the user is within the home context. Applying this within remote controlling could mean discovery and control of devices within a specific network. For example, a home entertainment system.

Contextual augmentation

The last level is based on argumentation of the reality, by adding contextual information. By linking data with context information it is possible to give the user information about the reality. This could be a tour-guide application that arguments the reality with information about the surroundings, giving information about nearby buildings and interests. In addition, it is possible to start a process based on the information about the reality to create an event-driven system that reacts to external factors. These links would enable execution of a process when the user is summoned in a specific situation.

2.1.3 The combined model

In Dey and Abowd (1999) the authors combined the theories of Schilit, Adams and Want and Pascoe into a simpler taxonomy. By comparing the taxonomies of Schilit and Pascoe, it is seen that *contextual sensing* is very similar to Schilit's *proximate selection* without the need of users selecting a specific item. Furthermore, *contextual adaption* maps directly to *context-triggered actions*, in the sense that both modifies or executes a service based on the user context. Moreover, Pasoce's *contextual resource discovery* is mapped *automatic contextual reconfiguration*. However, the last level, *contextual augmentation*, have no common ground with the levels defined by Schilit. Similarly, the *contextual commands* have no equivalent in Pascoe's taxonomy.

Schilit, 1994	Pascoe, 1998
Proximate Selection	Contextual Sensing
Automatic Contextual Reconfiguration	Contextual Resource Discovery
Context-triggered Actions	Contextual Adaption
Contextual Information & Commands	-
-	Contextual Augmentation

Table 2.2: The mapping between the to taxonomies

Dey and Abowd (1999) presents three categories to cover the major differences in Schilit's and Pascoe's taxonomies [2]. The categorization does not distinguish between information and services since it in many cases are hard to draw a straight line between the two. For example, if the user is presented with local printers available to use, is that information or services/resources? It depends entirely on the user. Secondly, *Automatic Contextual Reconfiguration* and *Contextual Resource Discovery* is torn down and merged into the two parts, presentation and execution. Thus, it is not treated as a separate feature category. The presentation of the availability of local resources and the usage of their services is two separate parts.

Presentation
Presentation of information and services to a user.

Execution
Automatic *execution* of a service.

Tagging
Tagging of context to information for later retrieval.

Table 2.3: The three categories of context-aware features
that a context-aware application may support

The authors choose to classify the context information itself as two types; primary and secondary [2]. The primary contexts are *location*, *identity*, *activity* and *time*. The residuals are left to the secondary type. Contexts of the secondary tier is argued as derivations of the primary one. Secondary context information such as phone numbers, can be derived from the identity of user using external data sources. Furthermore, some of the context of the secondary type may need multiple primary context to be used. For example, a forecast of weather, need both time and location to give context to an entity.

2.1.4 Deriving context using sensors

In a later paper Schmidt et al. 1999 emphasize the use of sensors to derive contexts [11]. A prototype in this paper show the basic implementation of automatic screen orientation that today is a fundamental feature of almost every portable device. This was shown using an early personal digital assistant (PDA) with an orientation sensor strapped on the backside, together with custom software rotating the screen based on the data from the sensor. Furthermore, many other sensory data is suggested.

Optical/Vision
Sensors: photo-diode, color sensor, IR and UV-sensor
Contexts: detection of objects, landmarks, people and gestures

Audio
Sensors: microphone and ultrasonic
Contexts: loudness, type of background noise, base frequency and speaker identification

Motion
Sensors: mercury tilt switch, angular sensors and accelerometers
Contexts: type of movement (walking, running, biking and driving)

Location
Sensors: GPS and cellular triangulation
Contexts: position (latitude, longitude) and location (work, home, abroad)

Bio-Sensors
Sensors: pulse, skin resistance and blood pressure
Contexts: activity, health and emotional state

Other
Sensors: touch, temperature and air pressure

2.2 Context-aware user interfaces

A special case of context-aware computing is context-aware user interfaces. User interfaces that react to the context and reconfigure the user interface based on this.

2.2.1 Implicit and explicit interactions

There is a distinction to make between implicit and explicit human interaction with context-aware systems. Traditionally, the user interacts with an interface through a graphical interface, gestures or speech. This is an explicit interaction were the user explicitly tells the system to execute an action, through a direct action. However, in Schmidt (2000) it is declared that the interactions that contexts infer on systems is an implicit human interaction. This is generalised into the concept called Implicit Human computer Interaction, iHCI [10]. Implicit interaction is interactions that the user implicitly makes by changing its context, causing side-effects triggered by systems. Consequently, this may occur with or without the users knowledge.

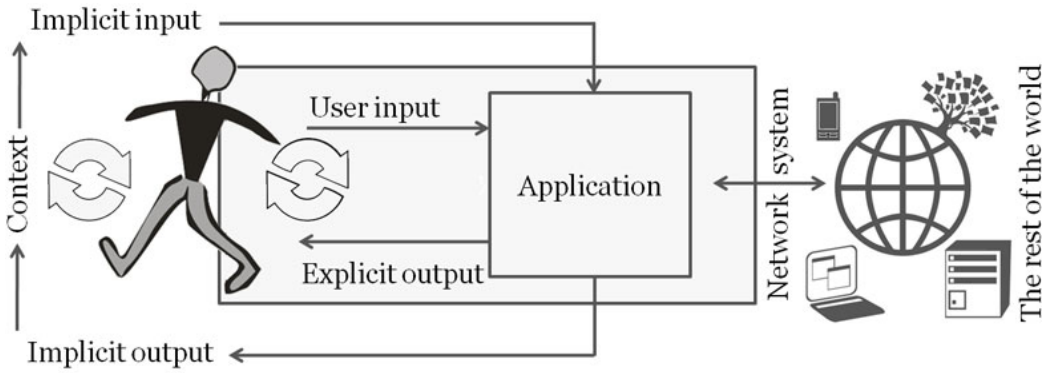


Figure 2.2: The concept of implicit and explicit human computer interaction.
©Albrecht Schmidt, The Interaction Design Foundation

2.3 User surveys

Surveys are one of the most common way to collect data from users. Surveys are so common because they are easy to conduct both in term of design and interpreting the results at least at a first glace. Blair Et al. (2013) there are a few issues with surveys that need to be taken into account when conducting one [1].

Firstly, one needs to understand what group of users that actually will answer the survey. Since it will never be possible to get every user to answer a survey one need to think about what group of users are in the subset that is being asked. In many cases only users that are very interested in the product or service will take the time to answer a survey. If this is the case, then that needs to be taken into account when reading the results. In other words, it is important to know who is answering and think about if these are the users you would like to have the answers from.

The second problem is tied to the first one. It is important to know if the group that is being surveyed have their own agenda. Some users might have an interest in changing the results in a particular way.

2.3.1 Survey types

There are several different ways to conduct a survey. Each of these has different properties. In this report several of these will be used. Below follows a description of the different survey types:

A web based survey is the easiest version to perform in the context of this report since the product has users all over the world. With a web-survey one can easily

get results fast for a very low cost both in term of cost and time. There are a few drawbacks from using web-surveys. First of all it is not possible to ask any additional questions if that would be relevant. It is also not possible to inspect the user and get a personal understanding about who the person is. The only information that will be provided about the person answering, would be a rough understanding of where the user answered the questions. This could be both at work or at home. A survey may also be presented to a user at an inappropriate time. However, it is a lot easier to quantify the results and see patterns using this technique, as opposed to interviews.

Phone or Skype interviews are also a possible way to get answers to a survey. The problem is then that people might get a call at a very inappropriate time and the user either chooses not to answer the survey or that they will not think through the answers before answering. It is also hard to have multiple choice questions with more than 3-4 alternatives since people will have a hard time remembering the alternatives. Therefore, all phone interviews in this report will not have any direct quantitative questions just qualitative questions (more on this in the User interview section) 2.4.

The last survey type used in this report is the face-to-face interview. This is the type that gives the best context and qualitative information. However, it takes a lot of time and it can be hard to get a representative group when a product has a global market without spending enormous amount of time and money on traveling. However, if it is possible to find a few major groups of users with less expensive survey methods it would then be possible to only visit a representative from each group of users. For an interview to work as a survey it needs to be a structured interview since a structured interview is targeted towards statistics.

2.4 User interviews

There are three types of interviews the structured where the each person that is being interviewed are asked the same questions in exactly the same order. These interviews are mostly targeted towards statistic and quantitative research just like a normal survey. There is also a semi-structured interview where the questions are not prepared in detail and are a lot more opened ended. This allows the interviewer to ask additional questions if the interviewer wants to learn more about a specific area. These interviews are a lot more qualitative since the person being interviewed are allowed to give any answer not just the ones that the interviewer had prepared beforehand. It is however, a lot harder to draw conclusions from these interviews since the results are unstructured. Instead of getting a numeric figure and a graph the results are not structured.

Lastly there is the unstructured interview where only a topic has been decided. And, the interviewer has a plan for what should be discussed. There is no questions that should be answered it is up to the interviewer to find the questions he or she likes to get answered during the interview. This form of interview is very qualitative and is mostly appropriate when there is little knowledge about what should be answered. Therefore, it could be a good first step to understand a new area. After this interview more quantitative questions could be created.

2.4.1 Interview techniques

As discussed by Raymond Opdenakker et. al. [7] depending on how an interview is conducted different types of feedback can be used by the interviewer. The form that gives the most quality is the face-to-face interview, where the interviewer cannot only listen to what the user says but also get a feeling for the body language and the mood that the interviewee is in during the interview. For example, if the interviewee is angry over something that has just happened it will be easy to see during a face-to-face interview. If not a face-to-face interview is possible the next best kind of interview when it comes to picking up social cues from the interviewee is Video phone calls. They can add a bit more information like hand gestures. A different problem with a phone interview is that it is not possible to create an ideal interview location where it is possible to conduct several interviews. This can create a difference in the answers just from the context around the interviewee. It is also harder to get the full attention from the interviewee when the interviewer does not have control over all distractions around the interview.

When a face-to-face interview is not possible an interview could also be conducted over a messaging service[1] [2] or email. A messaging service does have some of the advantages as phone or face-to-face interviews does Raymond says since it is still a synchronous form of interview where the interviewer would get answers just like in a normal conversation that is conducted with phone or face-to-face. It is also possible for the person to convey feelings with the help of emojis this is however, not the same quality emotions as in a real conversation.

Email is different from all the other techniques in that it is an asynchronous form of communications. This changes the interview since it gives the interviewee time to think about the answers before he/she is answers. When an interviewee have more time for the answers the answers will become cleaned up and sometimes even checked and corrected by a third party. However, it works well for purely fact based interviews for this very reason, but it would not work for emotional interviews.

[1] Facebook messanger - `https://www.messenger.com/`

[2] Hangouts - `http://www.google.com/+/learnmore/hangouts/`

2.5 Prototypes

Prototypes are an essential part of the human computer interaction design process. The most basic prototypes are called low fidelity, and are often performed on a mixture of paper, cardboard and post-it notes. The purpose is to get a low-cost prototype that is easy to change and adapt to insights at an early stage. In contrast, high fidelity prototypes results in partially complete functionality at a higher cost.

2.5.1 Low fidelity

Prototypes with low fidelity are easy to implement and are the first step of the human computer interaction prototype process. Often the prototypes are implemented on paper and tries to conceptualise and communicate the minimal viable product to potential users and team members. Low fidelity (LoFi) prototypes helps focus on the conceptual design and takes away much of the cost of changing prototypes from gained insights. The low costs make it possible to cheaply iterate upon different solutions, and promotes throw-away prototyping. Egger writes that fidelity is a remedy to the tunnel vision that is generated from a technical implementation[3].

2.5.2 High fidelity

Prototypes with higher fidelity comes with higher cost in both development and cost of change. Creating high fidelity prototypes requires technical knowledge in addition to the cost in terms of time implementing the prototype. As mentioned by Matthew Klee in the article "Five paper prototyping tips" [6] there are several psychological reasons for why one should create a paper prototype. For the team developing the idea a paper prototype makes it easier for the entire team to be a part of the development which is great, because it might not be the one most skilled at developing a high fidelity prototype that had the best ideas. It is also easy to collaborate around a design when all one have to do is cut paper and draw. It doesn't matter how fast someone is at developing a high fidelity prototype because it will not be as fast as it is to make it in paper.

When testing a prototype Klee also mentions an advantage in that users feel that they can be a lot more critical and speak their mind when something is made in paper since they understand how easy it is to change the design. The reason for this is that users try to be nice and not hurt the developers feelings. With a paper prototype they understand that it is created to be easily changed. Whereas a higher fidelity software prototype always look a bit to good and proclaims that it is hard to change.

Egger (2000) shows that students, especially within software engineering and information technology, tends to get a tunnel vision towards familiar designs, when the fidelity is higher [3]. They tend to use ideas from previous designs and tend to narrow possible ideas based technical possibilities and limitations.

When the paper prototype has been developed and tested and the most serious interaction issues has been figured out, it is time to develop a higher fidelity prototype. This could be a start of the real product where the developer starts with a rough design and then test that design to find more issues. The higher the fidelity the implementation it is the closer to a real product the prototype is and the more expensive and harder to change it is.

When the product has been developed and is used by users it is important not to stop prototyping and testing. When the product is out and used this is the perfect high fidelity platform to do prototyping on especially if it is a product used by large amount of users. The reason for this is that it is possible to compare how users use a product by dividing the users up. That way some users get the old version and some get the prototype version. This method is used in this report for doing high fidelity prototyping. When the users do not know if they are testing a prototype or using an old normal version, it may not always be possible to talk to the users about their experience, but it is normally possible to set up a statistical target for these users to reach. It could be to get more users to click the buy button. This is also often the most effective way to improve small things since it is impossible to get a user to say, "Change that text to ... And, I will be more likely that I will buy this product". The concept of creating small sparks of a product, trying to optimize and test what is best, is often recalled to as A/B-testing. It allows to easily test different variations and measure the outcome of different changes. There are companies that specialize in this kind of prototyping[3].

2.5.3 Criticism of fidelity classification

Stephanie Houde and Charles Hill propose in "What do Prototypes Prototype?"[5] that prototyping should not be divided into "High fidelity" and "Low fidelity", but instead the designer should think about what they like to test. Since it is not the materials a prototype is built of that makes it good or bad, it is what is possible to find out by building this prototype that is important. Houde and Hill suggest that the design process should be divided into three different dimensions; What role will it play in a users life? How should it look and feel? How should it be implemented? They write that it better to answer these questions instead and it does not matter what material or what tools they are built in. It is normally hard to build a prototype that would feel, look and have the correct role in a person's

[3]Optimizely `https://www.optimizely.com/`

life at the same time. It is often possible to create a prototype that tests one of these aspects. The article gives an example of a computer for architects where they tested the role and feel with the help of a pizza box with some weights in. They let architects run around with this box when they worked. They then pretend that they used it as a computer. That way they were able to see that the shape and weight would not be good enough for an architect because of everything else they were walking around with. In other words, by focusing on what they needed to find out and build a prototype for this and not focusing on the tools used they were able to build a cheap prototype that gave a real conclusion.

2.6 Related products

Today there are several products on the market that takes advantage of knowing the user context.

2.6.1 Google Maps

Google Maps[4] is a navigation application that provides interactive maps, navigation directions in combination with satellite/aerial imagery and a rich catalog of local business. It is a great product where many features depends on context-awareness, to yield a rich and useful interface. One of the feature that is typical within this field is automatic map rotation based on sensory data. In all modern phone have an embedded three-axis magnetometer that allows the phone to read the magnetic structures of the earth to determine the orientation of the device. This allows the phone itself to rotate interface into a vertical or horizontal layout. However, in addition it allows the device to obtain the compass direction that is used to automatically rotate the maps according to the current cardinal point. Furthermore, the application combines multiple traffic data streams to present the current traffic condition to users using it to navigate. This allows the application to warn the user, in addition to suggesting reroutes to avoid heavy traffic.

2.6.2 Google Now

Google Now is an intelligent personal assistant[5], that uses context-awareness and natural language interfaces. The program has a vast amount of information about the user. Mainly due to the connection to their Google account and the location sensor of smartphones. Location-awareness allows the interface to suggest nearby

[4] Google Maps - `http://www.google.com/maps`
[5] Google Now - `http://www.google.com/landing/now/`

attractions, businesses and public transportation. Looking at location over time allows it to learn where the user work and where the user's home is located. This allows suggesting commuting suggestions and travel times. Furthermore, the access to the users email and calendar allows it to present neat information about bought tickets, flight arrivals, upcoming events and restaurant reservations.

2.6.3 IFTTT (if-this-then-that)

IFTTT (if-this-then-that)[6] is a web service that allows users to automate a variety of different task. The service connects to large amount of web services to a simple user interface that allows the user to create rules based certain events that can trigger other events. This can for example be that the user is tagged in an image on a social network, the service can then trigger a download of the image and save it to a cloud storage (such as Dropbox[7]), or simply send an email to a certain address with the image.

2.6.4 Related products within remote controlling

There are different types of competitors for the current application. The way they impact this project also differs.

Directly competing applications

There are a few direct competitors to the application[8][9][10][11]. These applications are similar, in terms of features, to the current version of Unified Remote, for example, controlling multiple programs on a computer from a smartphone. These applications have the same ability and reason to create context-aware solutions since they have the same problems as Unified Remote with too many features that can be

[6] IFTTT - `http://www.ifttt.com/`

[7] Dropbox - `http://www.dropbox.com/`

[8] Max Remote
`http://play.google.com/store/apps/details?id=com.bitunits.maxremote`

[9] Win - Remote Control
`http://play.google.com/store/apps/details?id=banamalon.remote.win.lite`

[10] All in One
`http://play.google.com/store/apps/details?id=com.allinoneremote`

[11] Bluetooth Remote PC
`http://play.google.com/store/apps/details?id=cz.rozkovec.android`

hard to find. So far, there is no found direct competitors that have implemented a context-aware interface.

Specific remote control applications

App that control a specific program [12][13][14] is also a competitor to the current application. These applications focus on controlling a single application really good instead of controlling multiple applications. Since these applications do not control multiple devices or programs the number of features is limited, thus the problem of presenting relevant features is not as important.

Physical universal remotes

For controlling multiple TVs and other devices universal remotes[15] has been used for a long time. The challenges for universal remote controls are very similar to the challenges facing Unified Remote since they are also trying to build a user interface that allows users to easily control multiple devices in an easy way. A big difference is that universal remote manufacturers need to create hardware that is cheap enough for users to afford to purchase. In addition, only advanced users would pay a significant amount of money for a remote control. Most universal remotes only communicates over IR with other devices, which only allows one-way communication with devices. Thus, there is no state and ability to adapt to the current context.

Physical remotes from manufacture

Normal physical remote controls [16] that are distributed with new TVs and other devices have a bit different possibilities than universal remote controls (2.6.4) since they are distributed by the manufacturer the manufacturers could add proprietary technology to be able to include any kind of communication and sensors to the

[12] VLC Mobile Remote
`http://play.google.com/store/apps/details?id=adarshurs.android.vlcmobileremote`

[13] Kore
`http://play.google.com/store/apps/details?id=org.xbmc.kore`

[14] Timote for Spotify
`http://play.google.com/store/apps/details?id=com.jbl.android.spotimote`

[15] Logitech Harmony
`http://www.logitech.com/en-us/harmony-remotes`

[16] Physical Remote Controls
`http://en.wikipedia.org/wiki/Remote_control`

remote[17]. With this technology the manufacturers are able to create smarter interfaces for users. However, they are still limited to their own products and brands. There is also less need for smart macro features that can be found in universal remotes when a remote is only controlling one device.

[17] LG Smart Remote
http://www.lg.com/us/tv-accessories/lg-AN-MR500-magic-remote

3 Method

3.1 Initial survey

The company has access to a list of dedicated users that have signed up for newsletters and for helping with beta testing. This list consists of 4831 email addresses. Since users had to voluntary submit their email addresses the group will be biased to being advanced and/or interested users. See Appendix A for a list of questions and the results with 504 answers. As seen in question nine and ten the users that answered this survey is advanced users that is often early-adopters.

There are two questions that give information about what users would like to see in the app in the future. Question 13 that asks what users would like to have help from if there was an intelligent design that could help them and 14 that asks what they think should control the behavior of the intelligent design.

The survey was used to get a quick estimate of what kind of automation the most active users would like. The survey also get a starting base for later part of the project.

A survey was selected to get quantitative results for later parts of the project. Because this was the goal of the survey the focus in the questions was designed to be easy to answer without a free text answers. The survey was also designed so that it should be easy and fast to answer to get high quality answers throwout the survey and they user should not suffer from survey fatigue.

Since the results are structured and quantifiable the results is easy to illustrate using diagrams. No statistical analysis was performed to conclude variance and certainty.

3.2 User interviews

In order to better understand the underlying problem and getting an insight into user life, interviews was used. A quantitative survey only gives a very broad view of what users think and like. To really understand why users think like they do, it is important to talk to them since that form of interview give the most context 2.3.1,

in this case a phone interview is the closes that could be conducted with customers since they are located all over the world.

3.2.1 Audience

To find users that would be interested for an interview a random selections of the users that had answered the survey was selected and an email was sent to them informing about the interview opportunity and also a link to an online scheduling tool called youcanbook.me[1] where users could select a time that worked for them.

3.2.2 Environment

All interviews were conducted over Skype[2] and all participants were asked if they agreed to have the interview recorded, recordings was conducted with the skype plugin QuaSkype Call Recorder[3]. Notes were also taken during the interview for all questions.

3.2.3 Questions

The questions that were asked can be found in Appendix B. The first questions are targeted towards the user themselves. The users were asked to describe their primary occupation and interests, in addition to how long they have been using the product. This is important to know, in order to understand the users technical background, but also get an understanding of how much they have used the product. This is followed by some questions about the users feedback on the current product. This provides useful insights on the current product, but also ensures validation of the problem background. Finally, a section about context-awareness and what people feel about that. At the very last, an open discussion was held to allow the interviewee to ask anything about the company, the product and the future.

3.3 Context-aware prototypes

Based on results from the interviews, surveys and underlying background, prototypes were created. To be able to test the different implementation.

[1] youcanbook.me - `https://youcanbook.me/`
[2] Skype - `http://www.skype.com`
[3] QuaSkype Call Recorder - `http://www.ecamm.com/mac/callrecorder/`

3.3.1 Scenarios

To be able to test how different types of awareness could be implemented in an easy way several different scenarios were discussed however, because of time constrains only one scenario could be tested. A scenario that every test subject could relate to needed to be selected. In the end the scenario of what happens when a user arrive home after work was the selected scenario. It was also important that this scenario felt as personal as possible for the user to be able to relate to the situation and be able to imagine if the awareness level could work in real life. To be able to personalise the experience for the user a form where the user selected three different things that the user always do when they get home.

3.3.2 Scope & appliances

One of the future goals of the product is to branch into the smart-home and Internet of Things (IoT) market. These have a apparent problem, the number of features and possibilities to control. This makes it a perfect target for the prototypes, which address the problem of bringing the relevant features to the user at the right time. Since this is the target, the prototypes was designed to control common home appliance and media:

- Lights
- TV
- Music
- Movie
- Coffee maker
- Heating

3.3.3 Prototype fidelity & motivation

Since illustrating and discussing different levels of automation is hard. A tool (See figure 3.1) that could illustrate the difference for the interviewees was needed. It was also important that the tool did not feel like the finish product since that would make it harder to get the interviewee to express what they think 2.5.1. It would also be hard to illustrate this using a paper prototype. Therefor a low-fi simulator was created that show an apartment and a person that is able to move around in the apartment. The simulation will be used as a way to illustrate the different levels before a unstructured interview was conducted 2.4. The interviewees was also shown

a mobile interface in which all interaction with the apartment was conducted. The interface of this mobile interface was also designed to not look to good since we did not want the interviewees to think that they where using an finished interface.

Figure 3.1: On-demand view of the automation level simulator.

3.3.4 Technical implementation

The prototypes were build using a web-based solution. The back-end uses the server-side platform NodeJS[4] which enables rapid development of a large variety of applications using JavaScript[5]. The front-end was built using a web framework called Ractive.js[6], a template-driven user interface library optimised for web applications. This combination allows rapid and easy development of web applications. In addition, to being easy to deploy into a cloud provider such as Heroku[7].

[4] `http://nodejs.org/` NodeJS - A platform for easily building fast, scalable network applications

[5] `http://en.wikipedia.org/wiki/JavaScript` JavaScript (also known as ECMAScript)

[6] `http://www.ractivejs.org/` Ractive.js - The diamond age of web development

[7] `http://www.heroku.com/` Heroku - Cloud platform as a service

These tools allow rapid development of a functional interface that is easy to adapt and change. The simulator was built to allow a complete interaction with the phone interface while providing a simulation of the user context. One of the main priorities was to keep the feeling of a paper-prototype, by using white wireframe look. The goal of this was mainly to preserve the benefits of a low fidelity paper prototype mention in 2.5.1.

The prototype yields an interactive phone interface with buttons and texts. The behaviour of this interface was easy to define as scripts utilizing the dynamic nature of JavaScript. The house was drawn using a Canvas[8] to easily draw walls and other elements. In addition to the phone and the house, the interviewee is presented with a player that is possible move around in the house environment using the arrow keys. In addition, the earlier listed appliances are present within the home environment. This allows the interviewees to simulate their behaviour, both in terms of movement and their daily routine.

3.3.5 Interviewees

It was concluded that it would only be possible to conduct these interviews in a face-to-face manner, since it is allows observation of how the interviewees reacted to the different scenarios in detail. Furthermore, since the prototype was not used to test the interface of the context-aware application itself, but rather the level of responsiveness in it, there was no conflict in helping the interviewees through some of the difficult steps.

Since only advanced users of the app had been interviewed in previous test, it was also important to get a broader picture of the user base. Therefore, only people that were not active users of the app, were selected. All the interviewees did however know about the app and how it could be used.

Therefore people from other nearby companies were selected. These users were in most cases interested in technology, but none were developers. All of them are running their own IT based company.

3.3.6 Conducting the interviews

The interviewees were first asked to fill in what they normally do when they arrive home after work. They had to select three different tasks of the six possible options (See list in 3.3.2 for options). After this the interviewees was directed to test the different levels of automation, they were also asked to talk and describe what was happening and what they did. When they had tested all the different levels, an unstructured interview was conducted where the interviewees were asked what they

[8] `http://en.wikipedia.org/wiki/Canvas_element` - HTML5 canvas-element

thought about the different levels, they were also asked to compare the different levels.

3.4 Product prototype

The last test conducted, tests how learned patterns and context awareness can be implemented in the app. The same group that answered the first survey was targeted.

3.4.1 Implementation

The prototype was implemented as forks of the current application, with minor additions. Each of the forks produced was assigned a unique identifier in the analytics software Amplitude, to be able to track the usage of the different forks.

3.4.2 Prototypes

Three different prototypes where created to be able to test how learning user behaviour and how context could factor in how users used the application. To know if there was any changes a baseline version was created. The baseline was the exact same application that was already in use. This applications was used as a reference that the results from the other applications where compared to.

3.4.3 Test group

The three implementations was sent to the test group using e-mail. The group was split into three parts, each one getting a version of the application. The purpose of the implementations was not mentioned. The instructions did not mention the actual purpose of the implementation, but rather skewed the purpose to be a test of new performance tweaks in the application. The only difference between the emails was a support email address that should be used if there was any questions. The only difference in the address was that in the end there was an identifier that identified what group the user was in in case they needed help or had any questions.

3.4.4 Data collection

The data collection was implemented using an analytics tool called Amplitude[9]. In addition, a survey was sent out the try to get qualitative data.

Statistic data

In Amplitude there is a lot of data that could be compared. However, since the number of users that tried the apps was very limited only about 500 users in each group and only a fraction of those did use the application most of the data have too low significance to be used. The metrics that will be used from Amplitude are number of users and the length of each session.

Survey

To be able to compare the results from the three different apps three different surveys was sent out one for each test group. The first four questions where the same in all three surveys and the first six questions where the same for the survey sent to the "Time" test group and the "Pattern" test group. The time group got one additional question. See the questions in appendix C.

[9] Amplitude - Analytics for growing apps - https://amplitude.com/

4 Results

4.1 Initial survey

From the initial survey it is clear that most of the current users use the application at home, figure 4.1. Figure 4.2 also show that users are mostly interested in controlling home entertainment and smart home devices with the application. When users were asked to imagine that Unified Remote could do if it could learn from the user most of the users said that they liked Unified Remote to "Hide remotes or buttons" as seen in figure 4.3. After that, users were asked what event during the day they felt was most important 4.4 they felt that understanding different situations would be the best, then the location of the user. See appendix A for all questions.

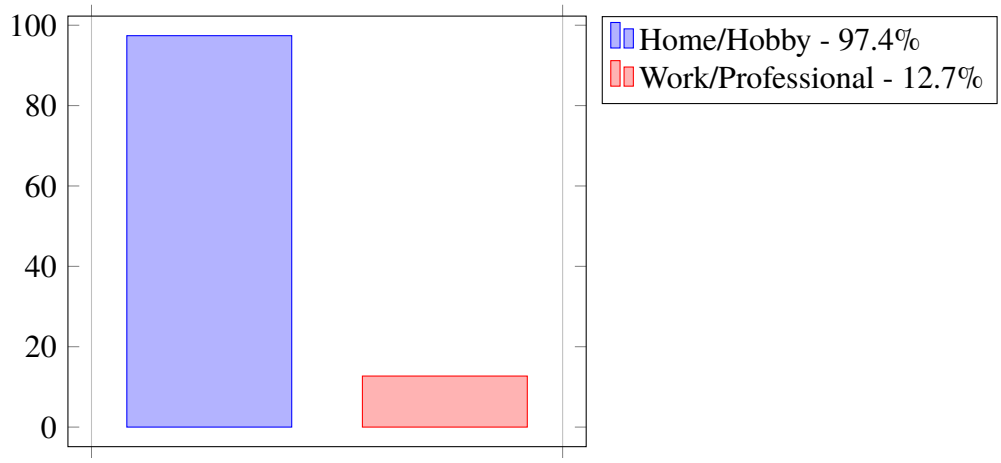


Figure 4.1: Where do you use Unified Remote?

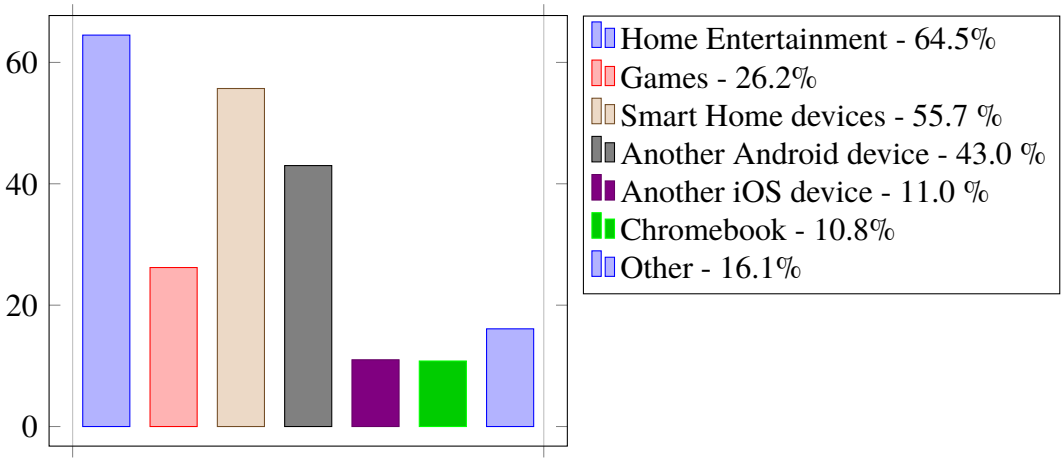


Figure 4.2: What else, if any, would you like to control using Unified Remote?

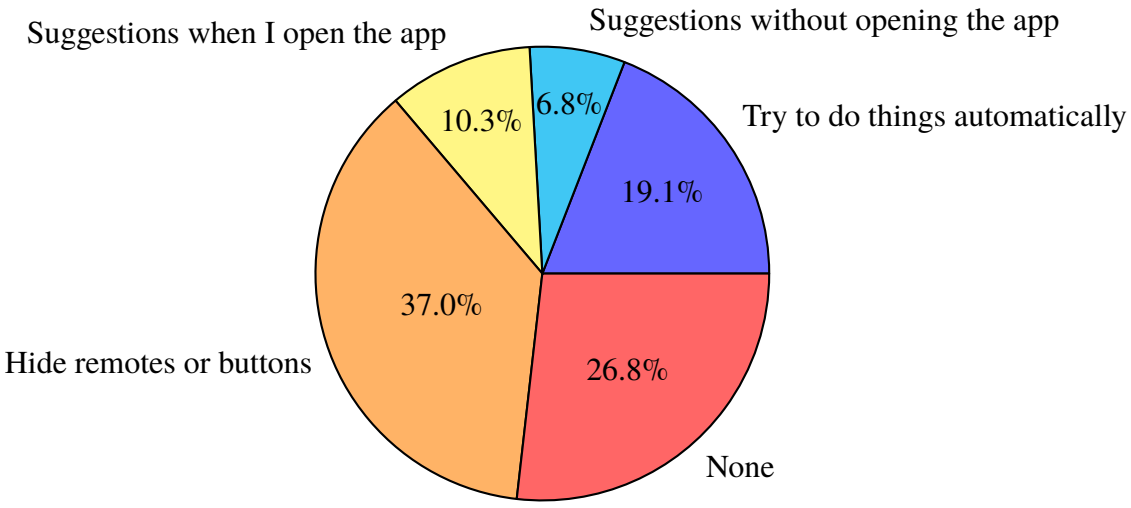


Figure 4.3: Imagine that Unified Remote could learn from your behavior to make the app easier to use. What would you prefer?

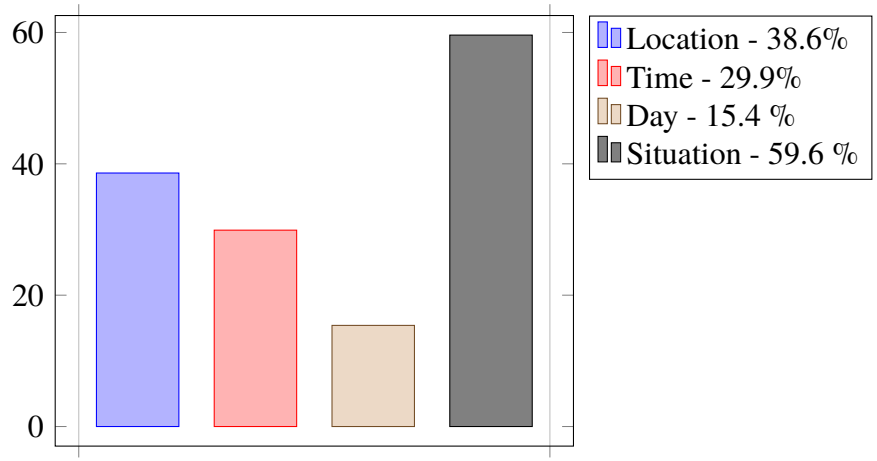


Figure 4.4: Again, imagine that Unified Remote could learn from your behavior to make the app easier to use. What is most important for you?

4.2 User interviews

The general response to making the product smarter was very positive. Most users thought that a model that learns the behavior of the user would be a suitable level one of the users would even like to see full automation where the system would run actions without user interaction. What is important however, as at least one user said, is privacy.

4.3 Context-aware prototypes

The goal was to present a number of different possible implementations that could be compared and reasoned around. To measure this, different metrics was required to be able to compare the different implementations.

4.3.1 Levels of application awareness

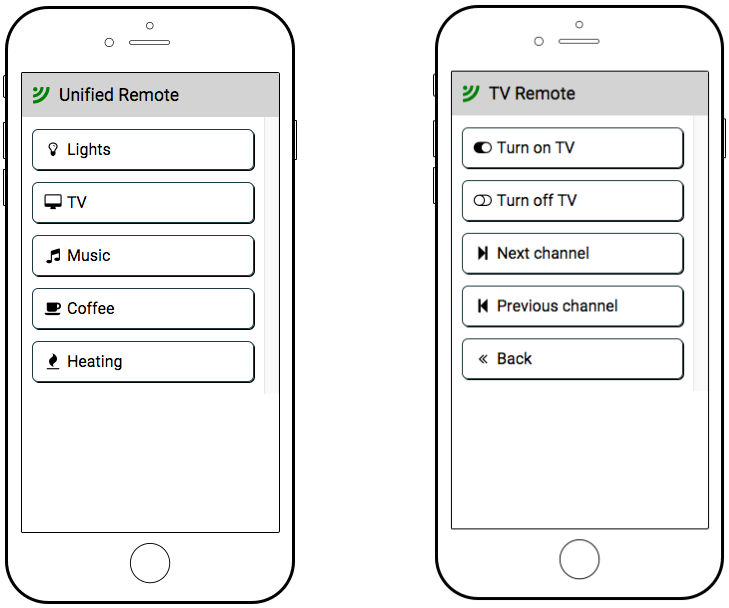
There is a fine balance between control and automation, either you give the user a full control and all possibilities or you removed the control on favor of automatic triggers. Giving full control and presenting all possibilities, makes the application harder to use, in terms of finding the features and finding the right one. In contrast,

if you remove all control from the user, and make everything automated, the user loses control. In addition, there is a large risk that a complete automation of tasks will behave unexpectedly. As a result, the user might develop a fear of what the application might do, resulting in a degraded user experience. Therefore, it is needed to find a balance between these to extremes.

Each version adds another level of awareness to the user context, starting with the current implementation that is purely on-the-demand.

On demand

This is how the application currently works. Users select from all actions what they would like to do. One action can be conducted at a time it is not possible to stack actions.



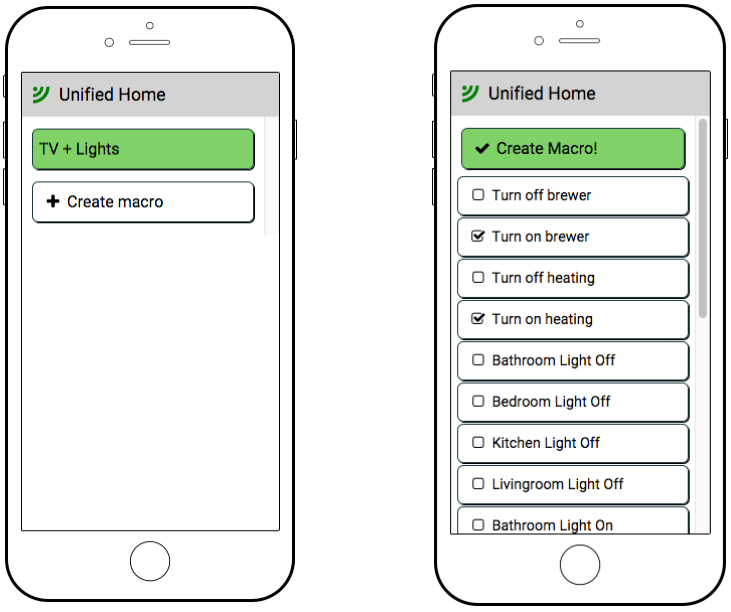
(a) All remotes in a list **(b)** The TV remote

Figure 4.5: The most basic level, without any notion of context

Macros

The same as On-the-demand except that the user can create shortcut sequences that can be programmed to a specific button that always is available. This would reduce the number of clicks and the time needed to perform a normal sequence

of pre-configured actions. However, the number of possible macros could be very long and the device does not try to help the user find suitable actions.



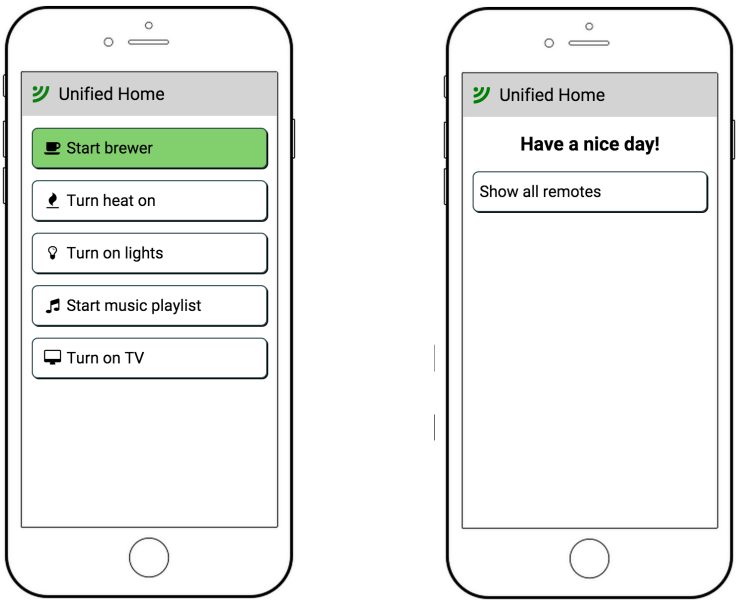
(a) Macros in a list **(b)** Interface for creating a macro

Figure 4.6: The second most basic level, list of basic actions that can be tied to a button

On this level the macros are saved per server instance, which yields a user context. Normally the users have a Unified Remote server installed at home that only is reachable through the home network. This binds the usage to the user and the home environment. The context in this case is based on resource discovery and location since servers are only reachable when the user is within the local home network.

Default macros

The same as the macro version above, with one modification. The macros are not user-defined but rather based on common patterns and predefined situations. These could be defined by the developers of the application, and is built into application. This can be easy, simplified versions of the current. In this sense the implementation actually loses the user's context since the macros is not created by the user.

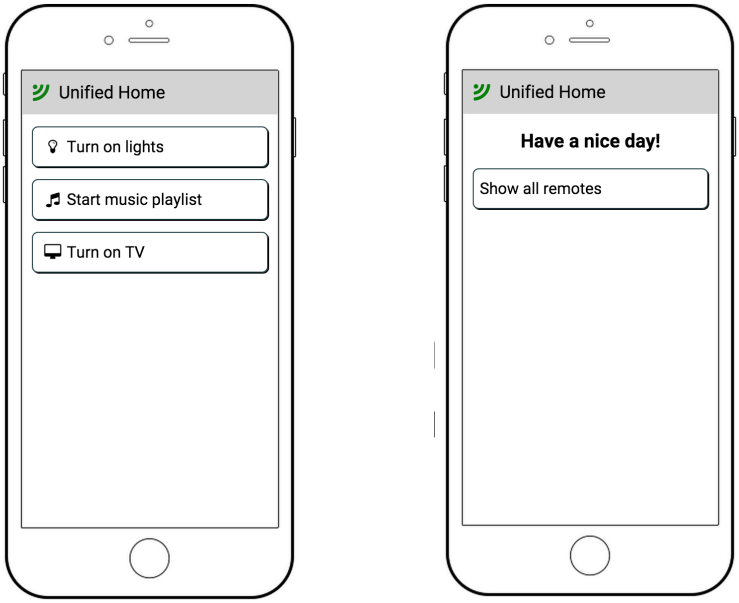


(a) Default macros in a list **(b)** All macros has been executed

Figure 4.7: The version were default macros is used

Context-awareness with autonomous learning

The application uses the context to create a model for the users habits. The application looks at position, time, available devices, state of available devices to figure out what the user like to do next. Then, shows the user shortcuts to functions in the application as well as create macros with series of functions. However, the user still has to press a button for anything to happen. Thus, this level does not yield any implicit interaction, it is still explicit.

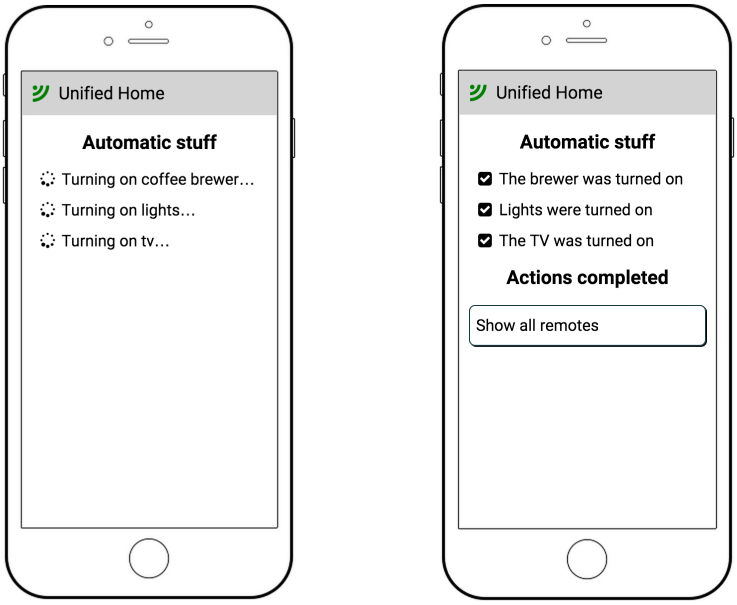


(a) Learned actions in a list **(b)** All macros has been executed

Figure 4.8: The version were learned macros is used

Unsupervised automation (automagic)

Same as "Context-awareness with autonomous learning" above, except the applications itself decides when it is appropriate to perform the actions. There is no user interface where the user can select what the application should do and when. This generates an implicit interaction between the user and the system.



(a) Triggers upon entering the house **(b)** All action completed

Figure 4.9: The version that automatically triggers actions

4.3.2 Context-aware prototypes

In total four people were interviewed about their thoughts and experience within the area. Some of the interviewees had a family and some not.

Some of the subjects had strong opinions about automation versus interface triggered actions. "If I would buy a product like this, it would be because of the automatic part of it" one subject said. In contrast, the application should never control the user. There is a fine line between suggesting things to the user, and suddenly the program making choices for you. The difference is between one click, and zero clicks. In general, the users would like the application to recognize patterns, but choose if they would like to automate. Some might want to self try to replicate their daily routine, by configuring macros and tie them to a context triggering, such as time and location. Some users seem to be eager in a quest to automate their life, and others seem to be concerned with not having a daily routine. "Human behaviour changes all the time".

Generally privacy was of no concern for the subjects. "This is no problem since it is something I choose to do." as it was expressed by one subject. Users seem to be willing to trade their data to gain the benefits of a smarter experience.

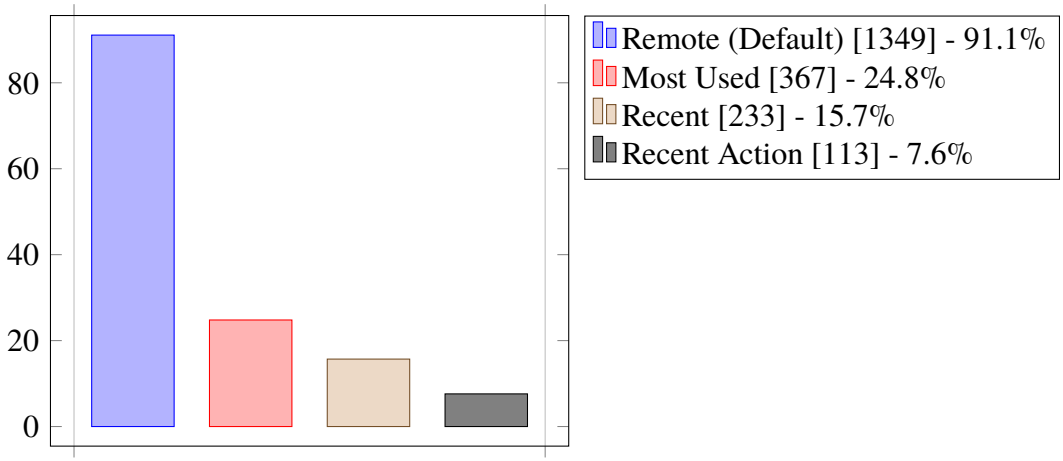


Figure 4.10: What tabs in Unified Remote do you use? (1481 responses)

4.4 Product prototype

The high fidelity prototype was implemented in two versions, one with time-context and one with only the pattern aspect. In addition, we sent out a copy of the current application to get a baseline in the data collection.

4.4.1 Baseline

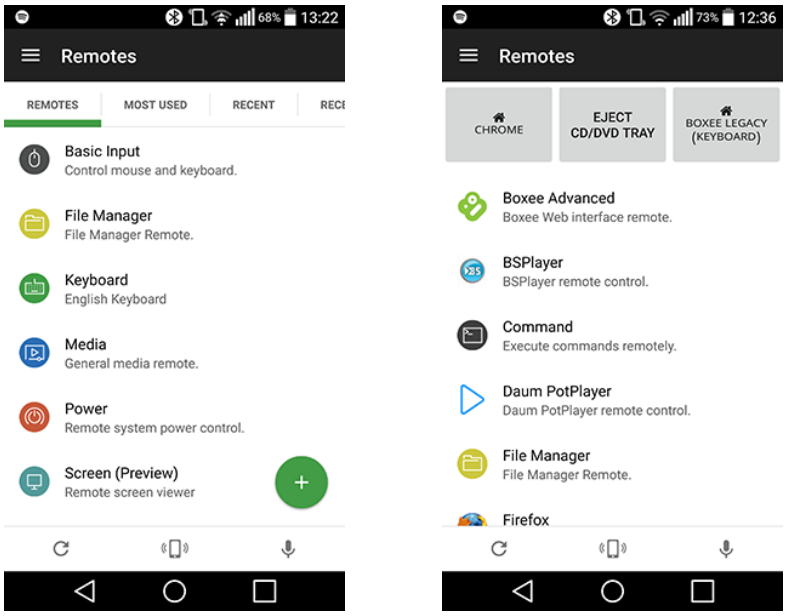
This application is an exact replica of the current application available to consumers see figure 4.11a. The only difference is the ability to track the application using the Amplitude software. The baseline application has the ability show remotes based on different sorting settings. The application has four different tabs that display different lists of possible remotes and actions. Three of the tabs displays remotes based on different sorting; alphabetical, 'recently used' and 'most used'. The last tab displays all the recent action executed by individual remotes.

4.4.2 Pattern

To the simplify the usage of the application, the different tabs were collapsed into a single one. Shown by the survey results in figure 4.10 most of the users does not use the other tabs, but rather stick with the first one presented (the alphabetical sorting).
The data showed that all the tabs were not used, thus they may not provide any

additional value. To improve the experience, it was decided to remove the "recently used" tab. Furthermore, the "most used" and alphabetical list were merged into a single list. This list combines ideas from both of the two tabs. The first ten remotes are the ten most used remotes historically by the user. Right after these remotes is the remaining remotes listed in an alphabetical order. The number ten is based on the data from the survey question "4. How many remote controls do you use on a regular basis?" in appendix A. Allowing up to ten remotes covers all of the users, including most of the power users.

The second modification is a view at the top of the list, displaying the three latest actions see figure 4.11b. Actions are the most atomic part of the application, and they are the endpoint of every interaction chain. The root-cause that triggers a user to use the application, is that they want to trigger an action to control something. Thus, to simplify the interface the obvious further development would be to extract the actions from the nested remotes view, and put them on the start page. This allows user to trigger actions without finding the correct remote, where it is located normally.



(a) Normal application interface **(b)** Time and pattern interface

Figure 4.11: Interface of the applications sent to users in the product prototype test

4.4.3 Time

To further improve the previous version, the use of a time context could potentially improve the product further. Thus, an easy improvement was to allow recent actions to be grouped based on the time of day. The time of day is split into six classes, see table 4.1. As action were triggered they were saved to a list based on the time of day, and then the recent actions view shows only the recent actions within the current class.

Time Of Day	Class
06:00 - 10:00	Morning
10:00 - 14:00	Noon
14:00 - 18:00	Afternoon
18:00 - 22:00	Evening
22:00 - 06:00	Night

Table 4.1: The classes used when grouping time of day

4.4.4 Amplitude Results

The figure 4.12 show the distribution between session times for each application. The graph shows that the normal application have a bit fewer "0 second" sessions than both time and pattern. However, for the longer sessions the curve looks similar for the "1-3 minutes" to the "1+ hour".

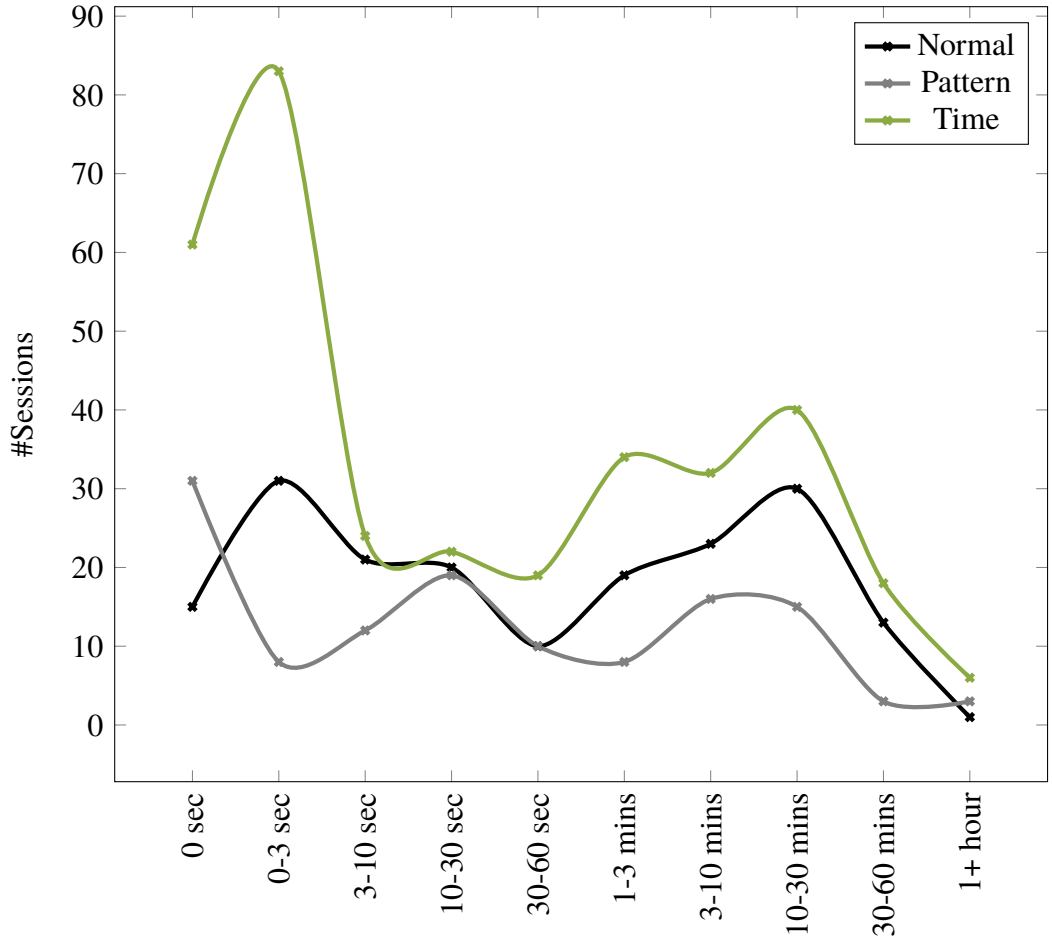
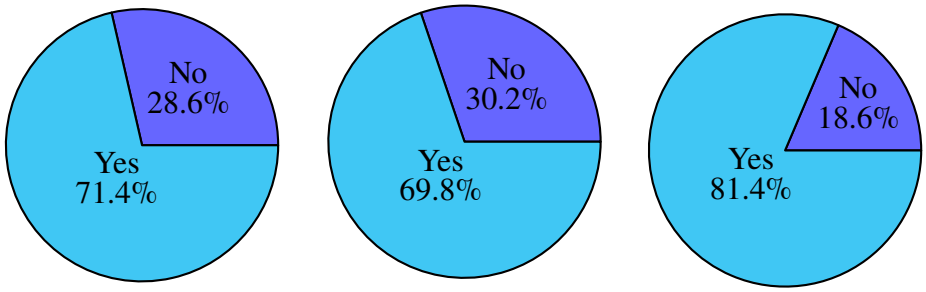


Figure 4.12: Distribution of session times between the different applications

4.4.5 Survey Results

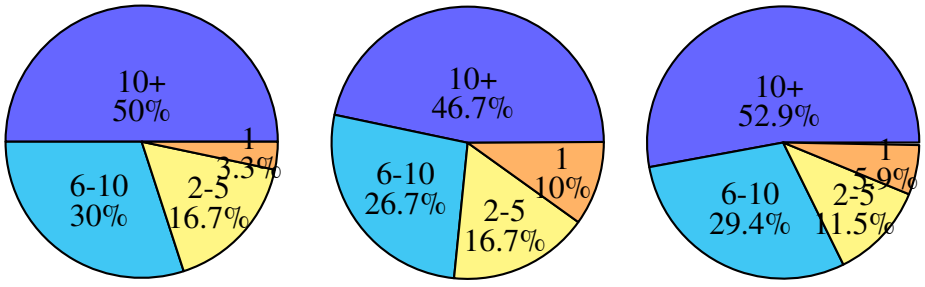
The survey shows that the usage amount is similar figure 4.14 the ratio between the number of people answering the survey and the number that tried the application is also similar between the applications figure 4.13. All applications changed

how users used the application figure 4.15 however, both the "pattern" and "time" applications changed more how the users used the application than the normal application. Most users liked the changes figure 4.16, but some did not notice that anything was changed at all. For the "pattern" application there was a significantly higher number of users that did not like the changes. Most users in the "time" and "pattern" applications did find the new top action bar useful figure 4.17 however, more of the pattern users did do that than the "time" users. The results for if the users found the remotes easier to find was equal between the "time" and "pattern" applications figure 4.18. About half of the users understood how the list was sorted in the "time" application figure 4.19.



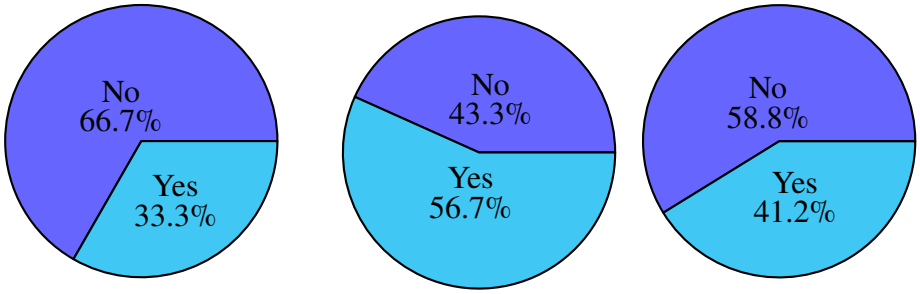
(a) Normal **(b)** Pattern **(c)** Time

Figure 4.13: Did you use the app sent to you?



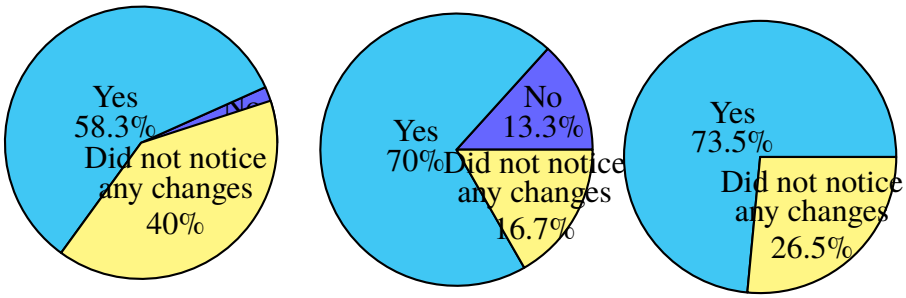
(a) Normal **(b)** Pattern **(c)** Time

Figure 4.14: How many times did you use the app?



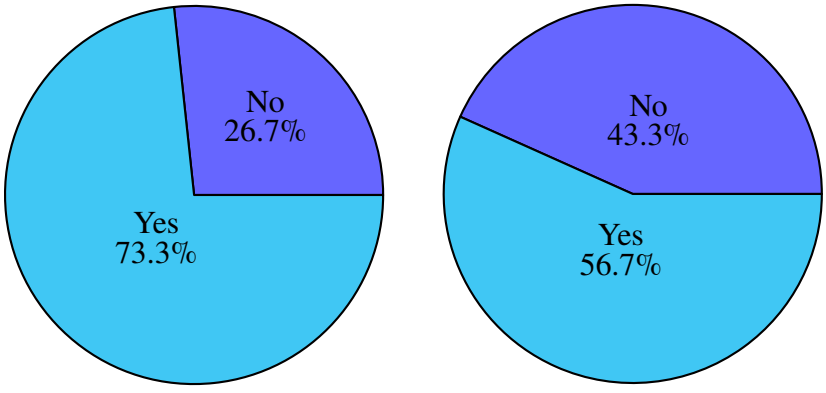
(a) Normal **(b)** Pattern **(c)** Time

Figure 4.15: Did the app change how you use Unified Remote?



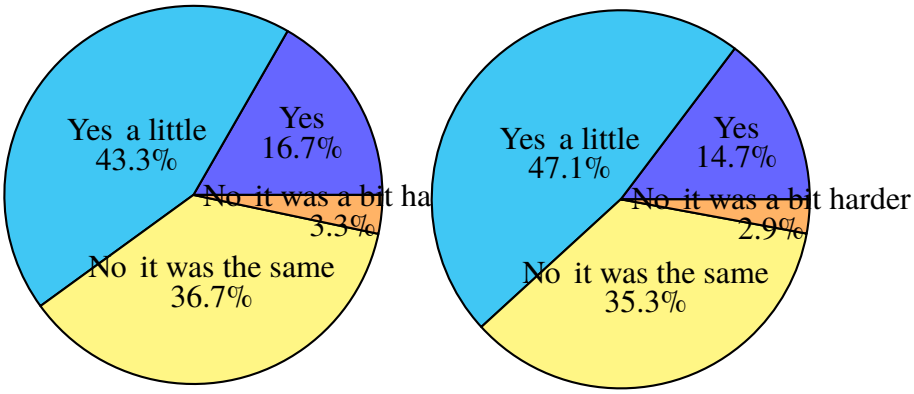
(a) Normal **(b)** Pattern **(c)** Time

Figure 4.16: Did you like the changes?



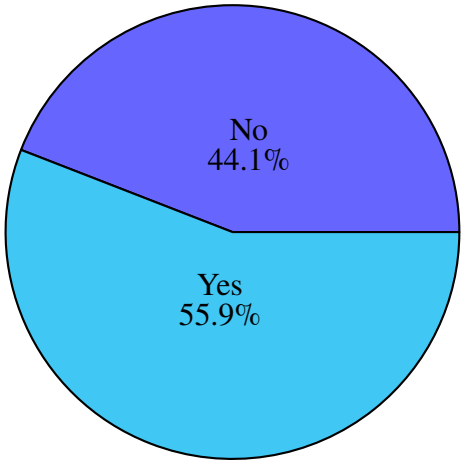
(a) Pattern **(b)** Time

Figure 4.17: Did you find the top action bar use full?



(a) Pattern **(b)** Time

Figure 4.18: Did you find it easier to fund the remote you were looking for)



(a) Time

Figure 4.19: Did you noticed that the remotes list sorting was time and usage based?

5 Discussion

5.1 Initial Surveys

The main purpose of the survey was to make sure the base assumption of what the user base look like was as expected. The survey was sent out to interested and tech savvy users and the users responding to the survey did not represent the full user base. However, since they did represent the most interested users that are the biggest fans of the application they did represent a good starting point. However, if this group was convinced that it is a good idea, it does not mean that a broader group would think likewise.

A survey is a very quantitative form of research that returns a lot of results that can easily be compared. However, since the results are very dependent on what and how questions are asked it is dangerous to only rely on survey data. There is also a tendency that people get tired of answering surveys and thereby starting out serious, but at the end participators are fatigued [4] and the quality of answers will suffer. Therefore, the survey sent to users was short and only contained 13 questions.

Since users clearly indicated that they would be interested in context-aware implementations this was seen as a positive sign for further quality research of what level the context-awareness should be placed. The surveys result indicated two possible levels of context-awareness. The first level is purely based on what users think they need, and what they are able to do to create their desired experience. The second level would be full automation that will learn what the users normally do and do this automatically.

5.2 Interviews

A lot of insights about how customers use the product and what they expect from the future of the product can be found with interviews. The results are purely qualitative, in contrast to the quantitative first survey. Together with previous research it is clear that making a solution that learns and adapts to user behavior is needed

and would be well-received by users.

When talking to users it is always hard to make them understand how something would influence them. It is also not possible to get a user to invent a product for you. A user can express problems that they have and they can suggest solutions, but they cannot create the product. This is especially true when a product is meant to change someone's behavior since some people is convinced that they already have optimized their life to the fullest. Asking them to change will always create resistance. It is like it was for Apple when they released the original iPhone, people thought it was cool, but only one day of battery on a cell phone made no sense. In other words, if Apple would have asked its users or random people what they wanted they would not have described an iPhone. Most likely they would have described a more powerful version of what they already had. A different example that is very famous is a quote from Henry Ford.

> *If I had asked my customers what they wanted they would have said a faster horse.* - Henry Ford

The reason for this is that people normally think around what they already have, not what would be possible. This is one of the jobs that an inventor has. It is hard to think outside the box beyond the norm.

It is important that users feel safe when using a product that learns from behavior and understand why they are recommended something. It becomes more important the more advanced the prediction becomes. If the prediction can be done on the device itself, without sending the data over network to a server, it is safer and more trusted by users. Most services today do the opposite and place all logic in the cloud, where the user has no control. The advantages of this approach is the ability to collect larger amounts of data, and at the same time allow changes to the algorithm, and easy testing with a small cost. However, placing everything in the cloud works when for example recommending what article you might like to read [1]. When the application is able to control your home some users prefer if they can keep some of the control to themselves.

5.3 Context-aware prototypes

Context-aware solutions are hard to discuss because you need to have the same image of how they work, in order to compare how different solutions would influence one's life. It is important for both parts in a discussion or interview to know the exact boundaries of the context-aware solution. Paper prototypes would not work since the interesting part of this experiment was not how the interface

[1] Google Now - `http://www.google.com/landing/now/`

should be designed, but rather to know the feelings a user had about a given level of context-awareness. Therefore a simulator was built to simulate the different scenarios, where the user was able to move around in an apartment and get a feeling for how the different levels would have worked for them in real life.

A think out loud solution for the interviews was chosen because it was important to understand if the interviewees understood what was happening. In cases where the interviewees did not understand the questions, were quickly answered and the scenario was explained. It was important to explain this in detail since the simulator was not built around a platform that the interviewees already understood. It is hard to summarize the entire interviews since they were all allowed to think freely in an unstructured way. Since the interviews were unstructured, the results might not be very accurate and hard or even impossible to summarize to a conclusion of what the interviewees really wanted. It would be good to have two parts of the interview. One part that let the users think freely and express what they are thinking, and a second part that was quantitative were the interviewees were asked a series of predefined questions. This would have given the benefits from both interview types.

Face-to-face interviews were selected because it is easier to conduct an unstructured interview compared to over a phone since it is possible to see the full reaction of the interviewee. Because of this, people that worked in the same office, but in different companies were selected. This caused some bias in the results since everyone was founders of small tech companies and all have an open mind to new ideas. The age difference was also quite small. They were not users of Unified Remote. The interviewees were also used to think freely and understand new concepts. This is not a perfect test group for this experiment since they do not give a good representation of the intended audience. However, it is a pretty good representation of the current user group, according to the initial survey performed. In general, it was assumed, that if this group does not think it is a good idea, a broader group will certainly not think that either.

The general impressions from the interviews were that they all thought it was an interesting improvement to the product that would be needed when a smart-home solution was going to be built. All of the users thought that some kind of context-awareness was needed. However, some of the users thought it would be a good idea if the application learned their behavior and acted by itself, others thought that it would be enough if the application gave suggestions for what it thought the user would like to do. The response was quite different depending on what path the discussion took. There also seamed to be a difference between the family situation, were interviewees that had family were less interested in having the application automate things. In contrast to the interviewees without family that had a more structured everyday routine. However, too few interviews were held to give any conclusive results for this, but one can imagine that if one have a family it

would be annoying if someone in the family has an application that automatically changes things in the house without taken into account the other family members.

5.4 Product Prototype

The final prototypes where built as an A/B-test where three different applications were tested against three different user groups. The different groups did not know about the other groups, all they knew was that they were testing some performance improvements. One of the applications had no changes at all and where only used as a control group. The other users had some improvements to the user interface where different aspects of the application were more visible than in the normal application. None of the applications had any new features, only interface changes. The goal of the prototypes was to find out if adding more context awareness and pattern recognition to the normal interface was desired by the users. The easiest way to do this was using a naive pattern recognition technique like the "most used remotes" or "recent actions" and promote them even more. Furthermore, adding a time aspect to the most used remotes part it was possible to add context-awareness to the interface.

Because these changes were simple to make in the application, it was easier to make them and then let users test and evaluate, rather than making paper prototypes and asking users to test them. Paper prototyping would also not work because the feature needed to be tested against real users from around the world. Making face-to-face meetings with these users would have been very expensive. The long term usage of the application was also more interesting than the immediate usage during an interview or observation session. Especially since the application needed time to learn the behavior of the user during at least 24 hours.

Since all users received the same email and the same information about the application they started the test with the same expectations. It should also be possible to see the difference in usage and responses in the survey.

Looking at the results from the survey it is clear that the usage of the applications was similar between the applications. The number of times the applications have been used by different users are, more or less, the same which makes the results more relevant. It is however interesting that 33,3% (figure 4.15) of users for the normal application say that their usage of the application did change even if there were no changes for these users. The reason for this could be that they had created some modifications to the application they had installed. For example they could have added Quick actions or widgets. When they installed the application that was sent to them widgets or Quick Actions were not transferred to the new application. Then, their usage of the application did not change because of

the new application, but because they did not use the application for long enough. The change for users with the pattern and time applications were however higher, which tells us that users did use the new design elements to some degree in these two versions.

Most users seemed to like the changes made, even in the normal application. It is interesting that users liked the changes in the normal application, this can probably be explained by users thinking that there were performance improvements. They had to say it was better than before since they did not have any performance issues. The results for the other applications could however be more accurate since most of these users had an opportunity to see what had changed. The number of users that did not find any changes was still significantly higher for the normal users as seen in figure 4.15.

The users that noticed the changes in the time and pattern applications, seems to think that the added actions bar list was useful. The reason for users not to think the action bar was useful could be because some users only use widgets or the mouse remote that did not show in the top bar. Some users also seemed to have some problem with the functionality. It might be that some of the use cases were missed in testing, but the majority of users thought it was useful so an improved implementation could be implemented in the future.

Most users thought that reordering of the remotes in the remotes list was an improvement. However, some users would probably prefer not to show remotes that they did not use. In figure 4.18 it is clear that users like the idea however, for some users it was a small or no improvement. This could be because they have already removed all remotes that they did not use and thereby finding the remotes was not an issue. There is a small increase in how many users thought it was easier to find a remote in the time application compared to the control, this is however a small increase that is hard to draw any conclusions from. For the time improvements a longer testing time would probably be necessary before the reordering would make sense. A lot of users did however say that they understood that the remotes list was reordered based on both time and usage. This is seen in figure 4.19 it might however be that these users understood that it was based on usage, but not time and then answered yes.

Looking at the statistical results from the analytics tool Amplitude it looks like the number of sessions is a bit higher for the time and pattern applications. This could be because interacting with the application is faster when the recent actions are easier to reach. It could also be because the users found the remote they were looking for faster. How the "0 second" sessions should be interpreted is hard to know it could be that Amplitude is not able to measure the length since the session is very short. It could also be that there was some problem with the application or that a user used a widget or quick action.

Even if the applications were sent out to over 1500 people only a fraction actu-

ally did try and installed the applications. This could very well impact the results in a significant way. Only a few users could change the results in both the Amplitude session times and in the survey. Only about 40 people did answer the survey for each application because of this each user answers did represent over 2% of the answers. Thus even single answers could have a significant impact on the results.

The fact that the users that did test the application only represent the core users and not the average users is both a good and a bad thing. It probably gave a bit more users that was willing to test the application and they probably gave it a bit more time when testing than the average user would have done. These users do however, tend to have special usage of the application. They often do use the more advance features of the application and thereby there will be more usage of custom remotes, widgets and Quick Actions than it would have been if it was the average user that tested the application.

It could also be questioned if one should listen to the power users when developing new features since they often have other interests than the average users. Listening to these users might push the features that are not interesting to reach a bigger audience. For this test it was however the only group of users that was possible to reach since they were the only ones that had indicated that they were interested in testing new versions of the application. These users are also easy to reach since their email addresses were known.

Reaching groups that do not have a higher than average interest in a product is hard since they do not want to spend time in testing a product they just want something that works. The only way to get average users is by pushing the changes to everyone, but this would require a lot more time being spent on designing and implementing the feature, to make sure it was good enough to be added to the normal product. This would have taken too long time for a short one week test. It would also not be possible to get people to answer a survey around what they thought in this case without a lot more work. Interrupting a user while using the application would not have been a good idea since they could become annoyed by the interruption. If the user is annoyed when answering the survey they might not give the answers they would have if they were not annoyed.

6 Conclusion

6.1 Initial survey

The survey proved that the assumptions about the group of users that receives the newsletter and participate in the beta test group were true. This information was important in the next step since then it was known which users would be easy to reach and interact with. It was also clear that they were not a good representation of the population of the world. It is however, unclear if it is a good representation of all the users of the application.

6.2 User interviews

Users thought that it would be interesting to have an application that would learn their behavior and adapted the user interface and gave recommendations to what it learned. They also thought it would be interesting with full automation especially when the application controls the home. It is however, hard to know if users would like this in reality or if it just something that sounds good when it is explained to them. It would also be very important that the algorithms that were used did not do too many errors since that would cause more frustration. Security and privacy are also important subjects to a lot of users. To make a more secure and private solution, a solution that only runs on the mobile phone should be investigated.

6.3 Context-aware prototypes

Building a prototype to illustrate different scenarios proved very efficient since there were never any misunderstanding during the interviews about how a scenario worked. The interviews showed that there are many different use cases for a context-aware solution. There are also many different needs that might not always work together. Because of this it is important to select one way going forward in the development. That is why it was decided that the best ways forward should

be the easy way and try to improve the experience for the current users with their current problems.

6.4 Product prototype

The product prototype worked well for testing improvements to the learnt patterns and context awareness. The survey and the Amplitude results show that most of the users that tested the application did like the changes and that the changes did change how users used the application. It is however, hard to know if the same changes would translate to average users if the applications were released to everyone. To be more certain of the results, a test would be needed in the released application. However, the results from the initial test do show that there is a high likelihood that the changes would be appreciated if released to everyone.

6.5 Overall

All tests have showed that users are interested in an application that is smarter and adapts to the users needs. Some users would think that it would be a bit scary when an application does know your habits, but in our tests we have shown that most users would accept the trade-off and thereby we have answered RQ1. They would be willing to exchange their data for the convenience of an application that will do whatever you would like whenever you would like it. It seems like most users would like a combination of self learning and control, were the application would learn patterns, but the user would turn the automation on, this answer RQ2. So that the application would ask the user when it has learned something new if the user would be interested in this every time a specific context occurred. That way would most likely be the easiest way for the user so that they not need to create macros or do any type of configuration. We have showed that it is possible with very small measures make it easy for the user to get access to the features the users use the most.

7 Future work

The project could be expanded to get more data and gain a higher statistical significance. The interviews could have been done with more subjects, the survey sent to a broader audience, and the context-prototypes could have included a way to build macros using context-triggers.

Secondly, it could be interesting to start recording the usage of users behaviour, and then try to analyse this in greater detail. An obvious continuation of this project would try to build a machine learning core to learn user behaviour over time. Trying to recognize patterns within the user behaviour and tie them to a specific context. Not only looking at individual actions, but capture sequences of actions as a macro in a specific context.

Furthermore, allow the application to share an even deeper context by looking at states of the device it is controlling. A shutdown device is applicable to certain action since it needs to be powered on first. This information is could be of great potential, in terms of simplifying the user interface by removing some visible elements.

Another interesting idea would be to try to dynamically change the user interface of individual remotes to fit the users needs. If a remote has ten buttons, but only four is used, why not hide or shrink the buttons that are unused.

Bibliography

[1] Johnny Blair, Ronald F. Czaja, and Edward Blair. *Designing Surveys: A Guide to Decisions and Procedures*. SAGE Publications, 2013.

[2] Anind Dey and Gregory Abowd. Towards a better understanding of context and context-awareness. Springer, 1999.

[3] Egger Florian N. Lo-fi vs. hi-fi prototyping: how real does the real thing have to be? *"Teaching HCI" workshop*, 2000.

[4] Mirta Galesic and Michael Bosnjak. Effects of questionnaire length on participation and indicators of response quality in a web survey. *Public Opinion Quarterly*, 73(2):349–360, 2009.

[5] Stephanie Houde and Charles Hill. What do prototypes prototype? *Handbook of Human-Computer Interaction (2nd Ed.)*, 1997.

[6] Matthew Klee. Five paper prototyping tips, Mars 2000.

[7] Raymond Opdenakker. Advantages and disadvantages of four interview techniques in qualitative research. *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, 7(4), 2006.

[8] Jason Pascoe. Adding generic contextual capabilities to wearable computers. In *Wearable Computers, 1998. Digest of Papers. Second International Symposium on*, pages 92–99, Oct 1998.

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

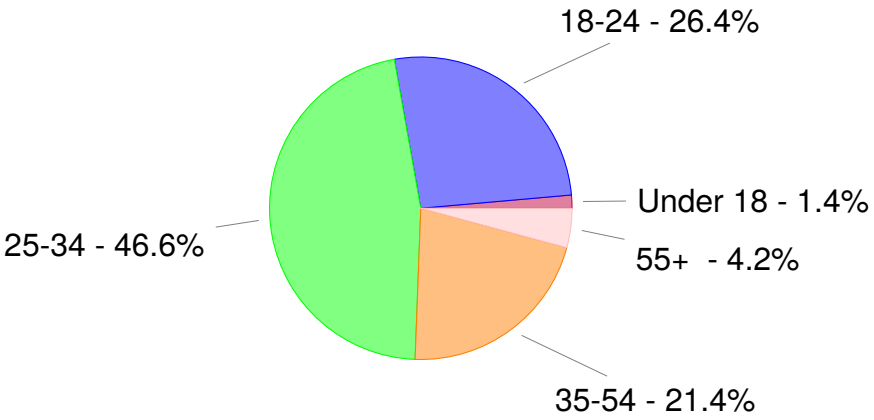
[10] Albrecht Schmidt. Implicit human computer interaction through context. *Personal technologies*, 4(2-3):191–199, 2000.

[11] Albrecht Schmidt, Michael Beigl, and Hans-W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

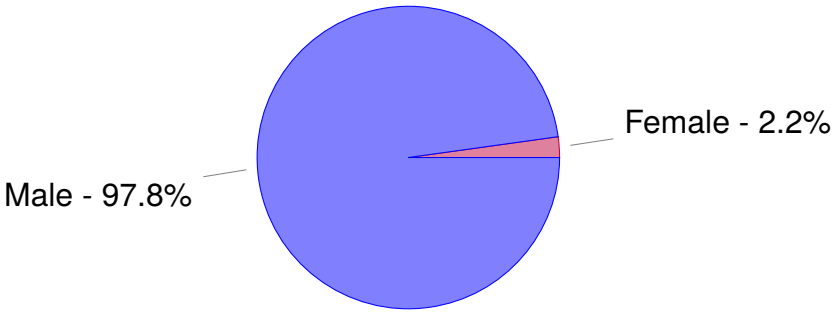
Appendices

A Intial survey

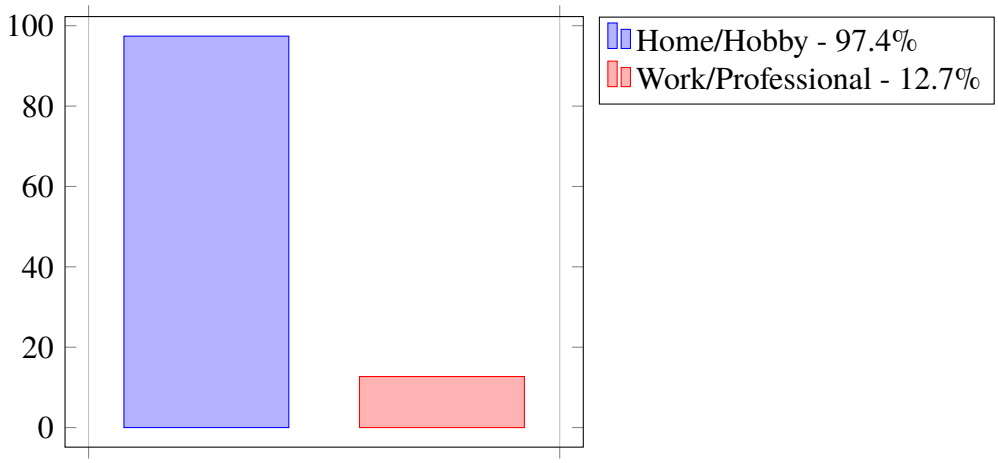
1. How old are you?



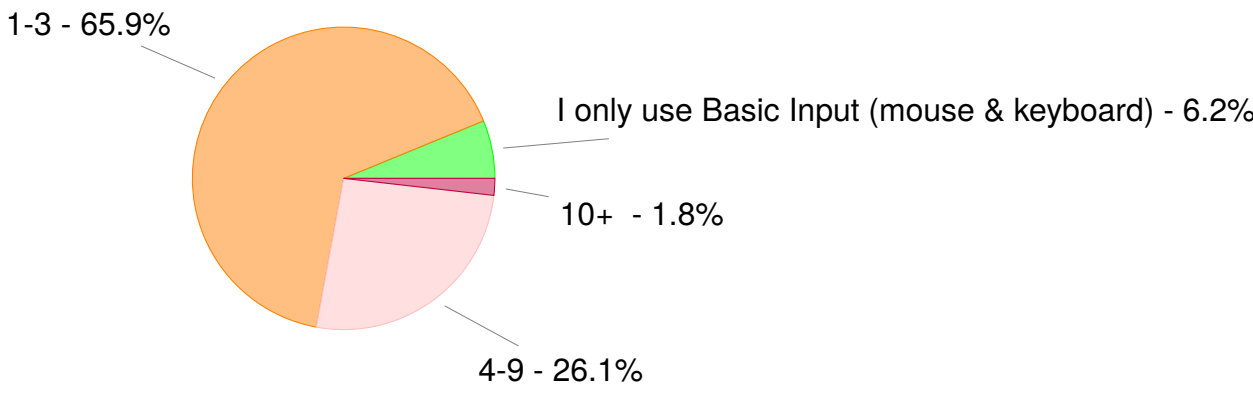
2. Gender?



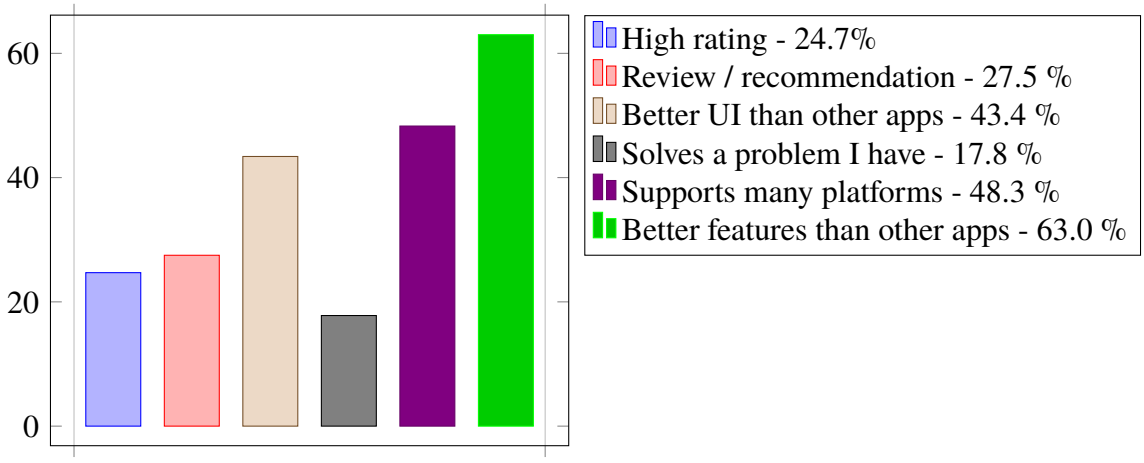
3. Where do you use Unified Remote?



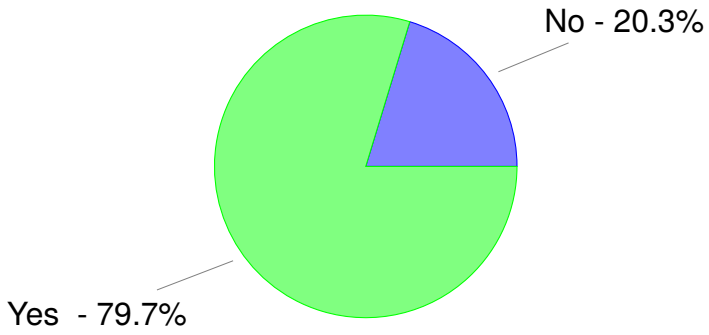
4. How many remote controls do you use on a regular basis?



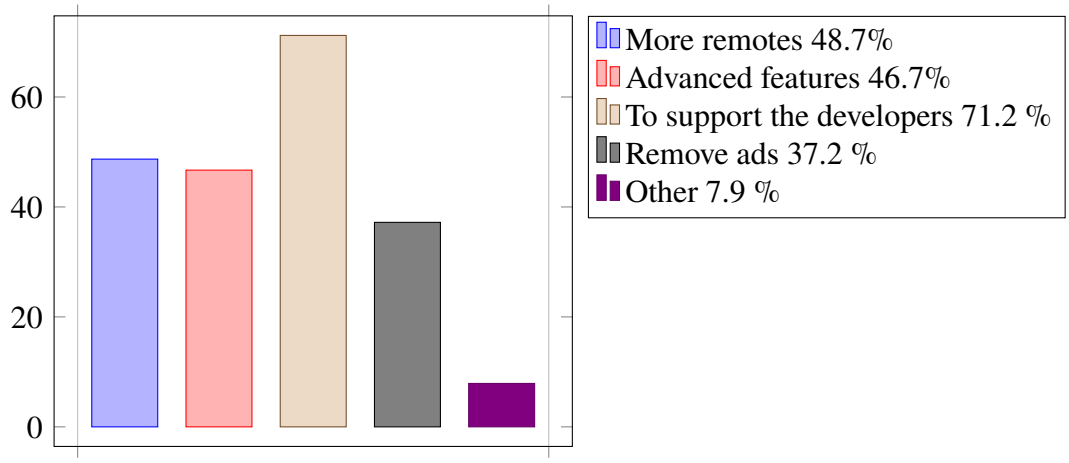
5. Why did you choose Unified Remote?



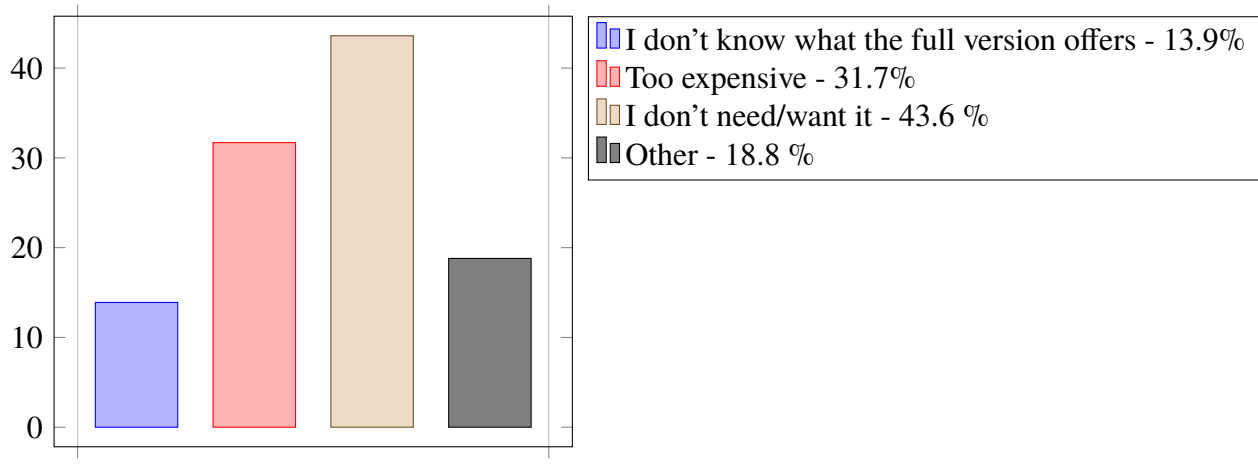
6. Have you purchased Unified Remote Full?



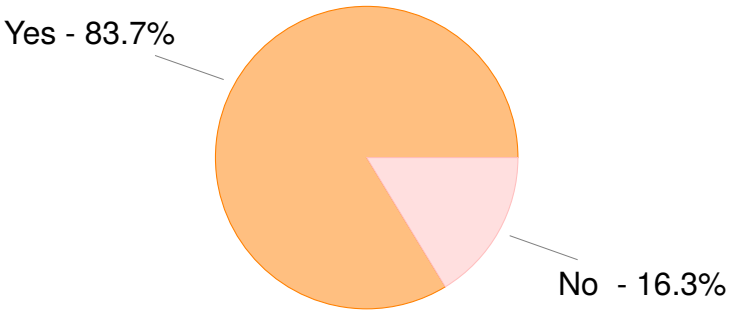
7. What made you choose to upgrade to the full version?



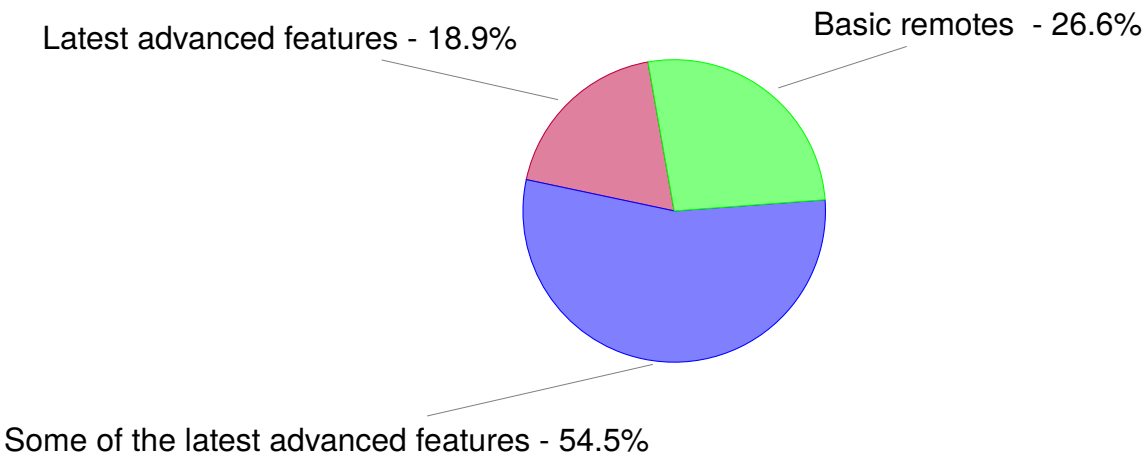
8. Have you considered upgrading to the full version?



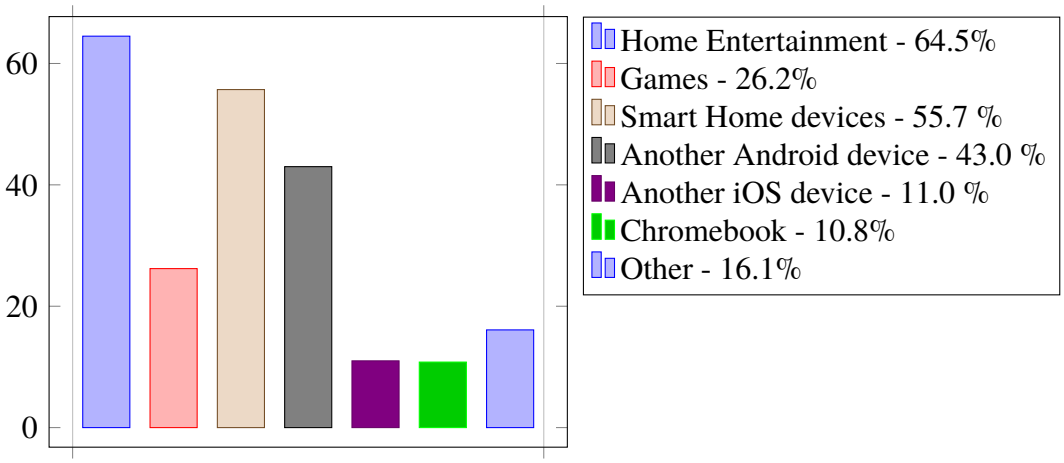
9. Are you an early-adopter of technology?



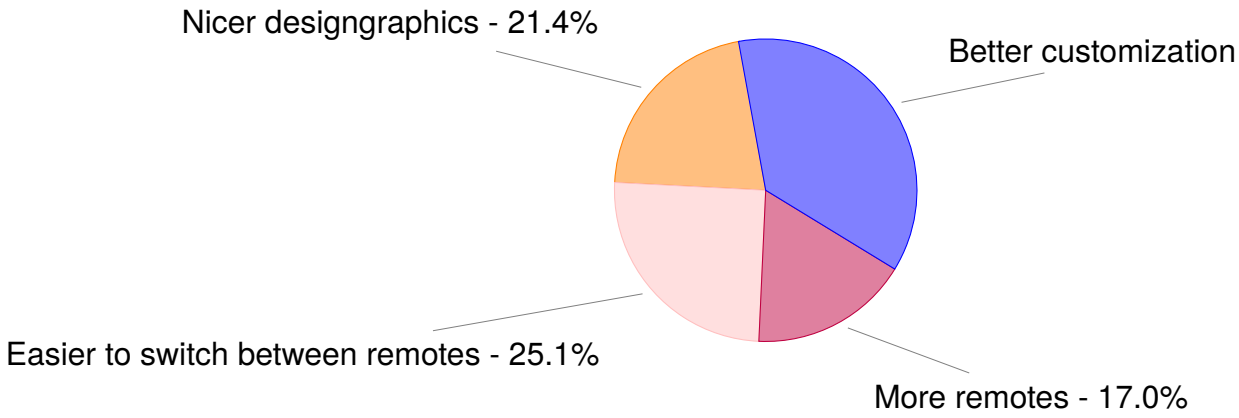
10. Do you consider yourself a Basic or Advanced user of Unified Remote?



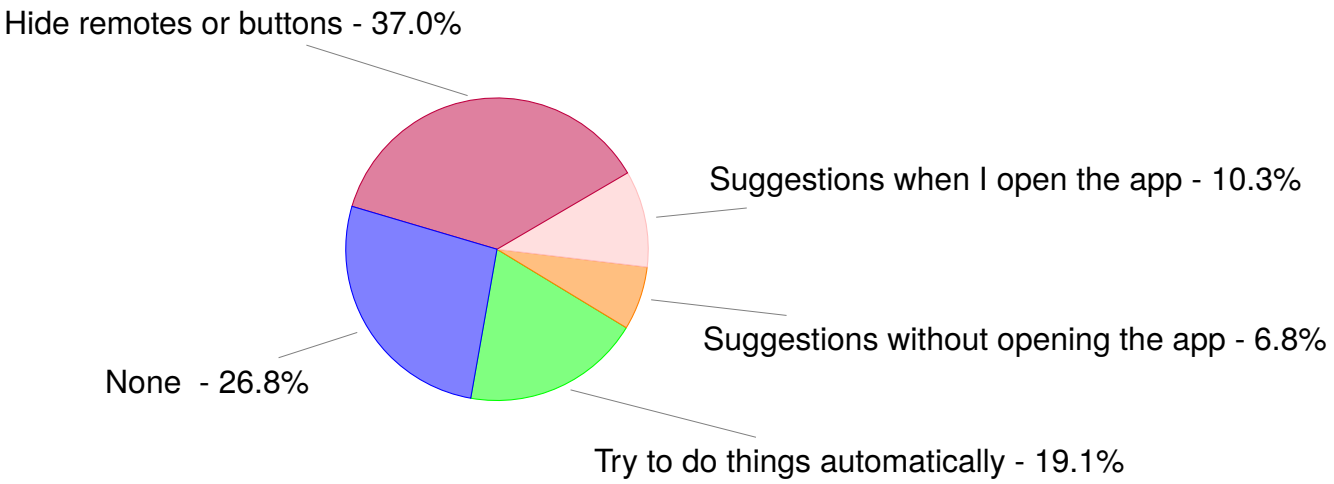
11. What else, if any, would you like to control using Unified Remote?



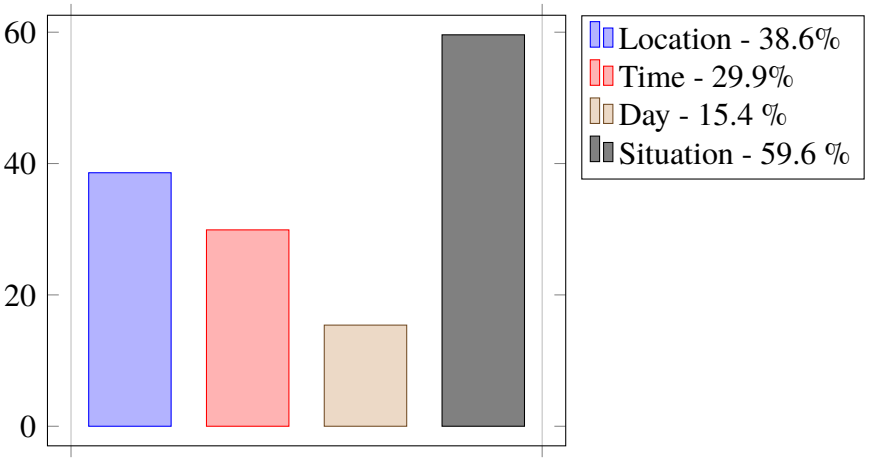
12. If you could pick one improvement, what would you choose?



13. Imagine that Unified Remote could learn from your behavior to make the app easier to use. What would you prefer?



14. Again, imagine that Unified Remote could learn from your behavior to make the app easier to use. What is most important for you?



B Interview questions

1. Is it okey if we record this interview? (for our own research, won't be shared)
2. Where are you calling from?
3. What time is it for you?
4. What do you do for a living?
5. How are you using Unified Remote today?
6. What is your favorite thing about Unified Remote?
7. What do you think could be improved in the app?
8. Have you created any custom remotes?
9. What kinds of 'smart home' devices do you have today? (Or do you plan to get any?)
10. Would you benefit from using a context-aware version of UR. (It knows where you are, what you are doing)
11. UR as it works today, pretty much on-demand
    UR will try to learn and execute actions for you, in a automagic manner.
12. If you think freely how would you like to use Unified Remote during a normal day.
13. Do you have any questions for us?

C Product prototype survey

1. Did you use the app that was sent to you?*
   (Yes, No)
2. How many times did you use the app?*
   (1, 2-5, 6-10, 10+)
3. Did the app change how you use Unified Remote?*
   (Yes, No)
4. Did you like the changes to the app?*
   (Yes, No, Did not notice any changes)
5. Did you find the new top bar of the recent actions useful?**
   (Yes, No)
6. Did you find it easier to find the remote you where looking for?**
   (Yes, Yes a little, No, it was the same, No, it was a bit harder, It was a lot harder)
7. Did you notice that the remote list sorting was time and usage based?***
   (Yes, No)

* Question used in all three surveys
** Questions used in the time and pattern survey
*** Question only used in the Time survey