AN INVESTIGATION OF SMARTPHONE APPLICATIONS:

Exploring usability aspects related to *Wireless Personal Area Networks*, *Context-awareness*, and *Remote Information Access*

A thesis submitted for the degree of Doctor of Philosophy

by

Jarle Hansen

School of Information Systems, Computing and Mathematics – Brunel University

June 2012

Abstract

In this thesis we look into usability in the context of smartphone applications. We selected three research areas to investigate, namely *Wireless Personal Area Networks, Context-awareness,* and *Remote Information Access*. These areas are investigated through a series of experiments, which focuses on important aspects of usability within software applications. Additionally, we mainly use smartphone devices in the experiments.

In experiment 1, *Multi-Platform Bluetooth Remote Control,* we investigated *Wireless Personal Area Networks*. Specifically, we implemented a system consisting of two clients, which were created for Java ME and Windows Mobile, and integrated with a server application installed on a Bluetooth-enabled laptop.

For experiments 2 and 3, *Context-aware Meeting Room* and *PainDroid: an Android Application for Pain Management*, we looked closely at the research area of *Context-awareness*. The Context-aware Meeting Room was created to automatically send meeting participants useful meeting notes during presentations. In experiment 3, we investigated the use of on-device sensors for the Android platform, providing an additional input mechanism for a pain management application, where the accelerometer and magnetometer were used.

Finally, the last research area we investigated was *Remote Information Access*, where we conducted experiment 4, *Customised Android Home Screen*. We created a system that integrated both a cloud-based server application and a mobile client running on the Android platform. We used the cloud-computing platform to provide context management features, such as the ability to store the user configuration that was automatically pushed to the mobile devices.

Acknowledgements

I would like to thank my supervisor, Gheorghita Ghinea, for all the help and assistance with my research experiments and the writing of this thesis.

Additionally, I am thankful for the support and work provided by my colleague, Tor-Morten Grønli, who helped me improve and expand on my own research.

Finally, I would also like to thank my family for all the support, which has been very important to me.

- Jarle Hansen, 15/04 - 2012

List of Publications

Journals

1. Grønli, T.M., Hansen, J. and Ghinea, G. (2012) "Mobility, Context and Cloud – Exploring Integration Issues in a Meeting Room Scenario". In *Journal of Interactive Mobile Technologies, volume 6, issue 2, pp. 13-22*
2. Grønli, T.M., Hansen, J. and Ghinea G. (n.d.) "The Tailored User Experience: A Multi Dimensional and Cloud Integrated Context-Aware Phone" In *Online Information Review* [Under Review]
3. Spyridonis, F., Hansen, J., Grønli, T.M and Ghinea, G. (n.d.), "PainDroid: An Android-based Virtual Reality Application for Pain Management". In *Transactions on Systems, Man, and Cybernetics-Part A, IEEE* [Under Review]

Conferences

1. Hansen, J. and Ghinea, G. (2009) "Multi-platform Bluetooth remote control*", IEEE International Conference for Internet Technology and Secured Transactions (ICITST 2009), London, UK, pp.1-2. IEEE*
2. Grønli, T.M., Hansen, J. and Ghinea, G. (2010) "Android vs Windows Mobile vs Java ME, A comparative Study of Mobile Development Environments*", ACM 3rd International Conference on PErvasive Technologies Related to Assistive Environments, Samos, Greece*.
3. Grønli, T.M., Hansen, J. and Ghinea, G. (2010) "Android, Java ME and Windows Mobile Interplay: The Case of a Context-Aware Meeting Room*", 24th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA).*
4. Grønli, T.M., Hansen, J. and Ghinea, G. (2010) "A Context-Aware Meeting Room—Mobile Interaction and Collaboration using Android, Java ME and Windows Mobile*",* In *2nd IEEE International Workshop on Middleware Engineering, Seoul, South Korea*.
5. Grønli, T.M., Hansen, J. and Ghinea, G. (2011) "Integrated Context-aware and Cloud-based Adaptive Application Home Screens for Android Phones*", 14th International Conference on Human-Computer Interaction*.

6. Grønli, T.M., Hansen, J. and Ghinea, G. (2011) "A Cloud on the Horizon: The Challenge of Developing Applications for Android and iPhone", *Proceedings of the 4th PErvasive Technologies Related to Assistive Environments. ACM*
7. Hansen, J. Grønli, T.M. and Ghinea, G. (2012) "Cloud to Device Push Messaging on Android: a Case Study". *Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications Workshops, (WAINA). IEEE Computer*
8. Spyridonis, F., Grønli, T.M., Hansen, J. and Ghinea, G. (2012) "Evaluating the usability of a Virtual Reality-based Android application in managing the pain experience of wheelchair users". *34th Annual International Conference of the Engineering in Medicine and Biology Society, 2012*

Book Chapters

1. Hansen, J. and Ghinea, G. (2012) "Multiplatform Bluetooth Remote Control: Implementation and Results*", Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications, IGI Global.*

Table of Contents

Source code: https://github.com/jarlehansen/brunel

Chapter 1 – Introduction

The area of information technology continues to experience considerable progress and innovation in recent years. Computers have evolved from large and very expensive devices, to mainstream products we take for granted in our everyday lives. Many people own multiple computing devices, from normal desktop computers to small mobile devices. We find mobile computing devices of particularly interest and this will be the focus of our study. We believe these devices, especially in the last few years, have evolved considerably. Not only do they have increased performance, they also provide more features than before, such as the new and improved touch screens.

In addition to the opportunities with mobile devices, they also present new challenges that are not present in standard desktop computing. Energy consumption, varying network coverage, and the relatively small screen sizes are examples of this. Moreover, in 2011 smartphone exceeded the PCs in terms of devices sold[1], which further highlights the importance of mobile devices and in this case smartphones specifically.

1.1 Mobile Devices

The technology surrounding mobile devices is one of the areas we have seen significant development in recent years. Today we have multiple devices with various form factors. Because the user has different requirements based on the situation where the device is used, the devices are usually designed for specific tasks. To provide a short overview of this area we have selected three common device types, namely *laptops*, *tablets*, and *smartphones*.

The first mobile device we wanted to introduce is the laptop, which is a traditional computing device. However, in contrast to desktop computers they are mobile. This means they are usually not dependent on any fixed infrastructure. Two important features of laptops are the ability to connect to the Internet with a wireless connection

[1] http://www.smartplanet.com/blog/business-brains/milestone-more-smartphones-than-pcs-sold-in-2011/21828

and to run on battery power. Laptop devices are comparable to desktop computers in that they function well for both media consumption and creation.

The next type of device we wanted to focus on is the tablet. These devices are normally smaller than laptops, and will usually have no physical keyboard. The iPad is one of the most well-known tablet devices currently on the market, and it provides a touch-based screen that the users interact with. Tablets are usually lighter and more practical to carry around compared to laptops. Even though it is possible to use them for work (media creation), they are best at media consumption, such as watching a movie, playing games, and listening to music. This is because it can be cumbersome, at least in comparison to a physical keyboard, to write for example a word document or e-mails using the touch-based input.

The last device type we want to introduce, and the most important in the context of this thesis, are the smartphones. Mobile phones have evolved considerably from their beginnings, when they were very expensive to use, had poor battery life, and were too heavy to carry around. Initially, the devices were only capable of making phone calls, and then text messages became popular, before cameras and sending pictures were included as mainstream features. Compared to tablets, the smartphone devices are smaller.

Within the area of smartphones/mobile phones, there are two categories, namely *feature phones* and *smartphones*. There is no standard definition of what constitutes a smartphone. However, in the context of this thesis we are using the following description: The smartphones have more technology and functionality incorporated compared to feature phones (Kang et al. 2011). This means that a smartphone provides normal mobile phone functionality and offers additional features, such as Internet connectivity and e-mail capability to name two common examples. Additionally, it is also normal for modern smartphones to offer the ability to install local applications, usually provided through an application marketplace.

The popularity of these devices has grown significantly in recent years. As stated by (Do et al. 2011), several billion individuals carry these devices today, and they are becoming reachable almost *anytime*, *anywhere*. According to Gartner (2012), the total

growth of smartphone sales was up 47.3%, with 149 million units sold, in the fourth quarter of 2011, when compared to the same quarter of the previous year. For the entire 2011, the smartphone sales worldwide reached 472 million units sold, up 58% from the previous year. With these numbers in mind, it is clear that smartphones are becoming increasingly popular and a mainstream product.

The smartphone devices are multipurpose and very important in our day-to-day activities. Not only do we rely on them for phone calls, we use them to browse the web, send and receive e-mails, and much more. This evolution, both on the hardware and software side, is a result of continuous research and development in the mobile computing area. We continue this introduction with a short presentation of how the hardware and software within the smartphones have progressed over the last few years, before providing an overview of the topic and content of this thesis.

1.1.1 Smartphones

According to Bagchi (2010), the hardware devices and wireless networks are evolving rapidly in the commercial landscapes of information technology, and hence they are enabling the realisation of high-end mobile computing applications. To provide an indication of the performance progress over the last few years we compare three smartphone devices, which are all shown side-by-side in Figure 1-1. On the left side, the first device is the Nokia E61, released in 2005. Five years later, the HTC Evo was released (shown in the middle), and in 2011 the iPhone 4S became available on the market (on the right).

Figure 1-1, Nokia E61, HTC Evo, and iPhone 4S comparison.

To demonstrate this a a DOM (Document Object Model) core performance test (*Hixie DOM Core Performance tests*[2]) was conducted on all of these devices. The performance test runs in a browser, and is a test of basic DOM operations, such as append and prepend. Figure 1-2 presents the results for the three devices.

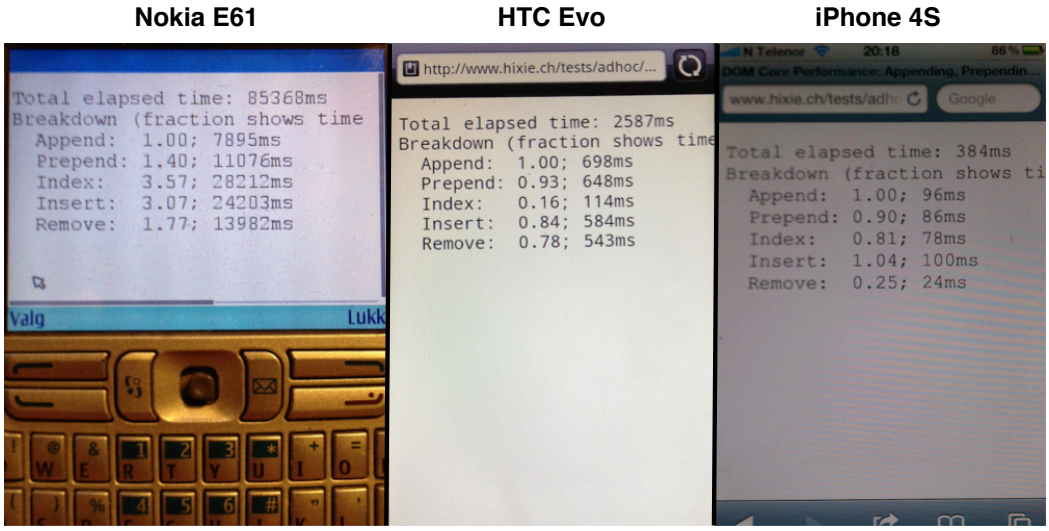


Figure 1-2, Hixie DOM Core performance tests.

It is important to note that the numbers only show one performance aspect, namely the DOM core performance. In any regards, the results are still quite remarkable. Nokia E61 used 85368ms, while the HTC Evo used 2587ms, and finally the iPhone 4S used only 384ms. The results show a performance improvement of 222.3x, when comparing the results from Nokia E61 and the iPhone 4S. These numbers are only an indication of the differences between the devices, and do not mean the iPhone 4S is over 200 times faster in every respect. When considering hardware resources such as memory, the iPhone 4S (512MB RAM) has about 8x the amount when compared to Nokia E61 (64MB RAM). Although these results are in no way considered a complete performance comparison, they demonstrate the improvements that have been made in the smartphone technology, both hardware and software, over the last six years.

With powerful hardware and software available, the field of mobile computing is quickly changing. The previous, and very strict, limitations are not considered the same

[2] http://www.hixie.ch/tests/adhoc/perf/dom/artificial/core/001.html

obstacle any more. With the increase in computing power and storage space, the devices are able to run applications that were previously limited to desktop computers. Indeed, nowadays smartphones are essentially small and powerful computers that can fit in ones pocket.

1.1.2 Applications and Development

The innovations in hardware capabilities open up new opportunities and challenges when developing systems that run on or integrate with mobile devices. When combined with the wireless capabilities of high speed Internet through EDGE, 3G and 4G as well as Bluetooth and WLAN, many new research possibilities appear. Furthermore, with the updated network infrastructure and more affordable payment options from the ISPs (Internet Service Providers), the *always-connected* devices are becoming mainstream.

While smartphones are becoming increasingly powerful, the software they run has also gone through some major evolutionary steps. Particularly the Android and iOS marketplaces have been a very important factor in the platforms' success. In 2011 both Android Market[3] and the App Store[4] reached over 500,000 available applications. The maturity of the platforms and the popularity of apps are giving businesses a new channel to promote products, offer new features and generally expand their methods of reaching out to potential customers.

Moreover, the ability to purchase and download native applications directly to the smartphones has proven to be a popular service for both consumers and developers. Developers are able to publish their applications quickly and users can navigate through a library consisting of many thousands of applications, providing everything from games and educational software to enterprise solutions. Additionally, the rating systems also provide end users with the ability to directly give feedback on the quality and price of the offered application. These features have certainly made people use their phone for more tasks than before. With the increase in usage and capabilities of the smartphones, new and interesting research opportunities have emerged.

[3] http://www.research2guidance.com/android-market-reaches-half-a-million-successful-submissions/
[4] http://www.apple.com/iphone/built-in-apps/app-store.html

1.1.3 Why focus on smartphones?

We have chosen to investigate smartphones as part of this thesis due to three main reasons, which are presented next.

1) The widespread use of smartphones.
There are a considerable research that points at the increasing popularity and importance of smartphones, and that they provide an important tool in the everyday lives of consumers. In addition to the report created by (Gartner 2012) that shows a significant increase in popularity of smartphones, several relevant research efforts also verify this trend: (Cheng et al. 2007; Kang et al. 2011; Palit et al. 2011; Xu et al. 2011).

Because of the increased popularity of smartphone devices, the general availability provides a major benefit. Systems utilising smartphones only require technology that is already widely accepted and used by a considerable number of people. Also, it is becoming increasingly important for businesses to offer support for smartphones, because more users are accessing the services/resources using these devices. Nah et al. (2005) state that mobile and wireless devices make it possible for organisations to conduct business more effectively.

2) Smartphones provide considerable hardware and software capabilities.
When Apple released the iPhone there was a big step forward, both in terms of hardware and software capability. The iPhone completely reset the user expectations for mobile experiences (Charland & Leroux 2011). There were smartphones available before the iPhone was released, however, these were very limited when compared to modern touch-screen devices, such as the iPhone and Android-based smartphones.

Apple released the first iPhone in 2007[5] and Google, under the Open Handset Alliance, released Android (*beta*) the same year (DiMarzio 2008). The innovations of these devices, such as the large touch screens and on-device sensors, opened up many new and interesting possibilities for application developers and researchers. The devices are

[5] http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html

now capable of more than making phone calls and sending text messages. Additionally, with increased hardware capabilities, better runtime environment, and improved developer tools, the previously impossible suddenly becomes possible in terms of creating new and useful applications. Cheng et al. (2007) point out that these devices are becoming increasingly popular because they provide all-in-one functionality by integrating traditional mobile phones with handheld computing.

3) The smartphones can utilise high-speed wireless networks.
Smartphones have the ability to connect to several different networks. EDGE/3G/4G are available in many smartphone models. Also, the devices will commonly have the capability to connect to local resources such as WLAN and WPAN. With the possibility to connect to the Internet and other devices, many new opportunities are created. One is no longer confined to only the information found locally on the mobile device. This means that the devices provide mobility combined with (often considerable) computing power, which is not possible with standard desktop computers. Moreover, it opens up many new and interesting research areas because of the mobility, such as the ability to register and utilise the context to improve the user experience. According to Coursaris and Kim (2011), the mobile devices are reaching mainstream status because of reduced technology fears among consumers and reduced cost. Features such as connectivity, communication, and data services are also important factors contributing to the increased popularity.

1.2 Usability in Smartphone Applications

As previously presented, with the increased possibilities on both the hardware and software side, there are indeed many interesting topics that are open for new research contributions. We have introduced smartphones and provided a justification for focusing on this mobile device type.

The research topic we wanted to investigate in context of using these devices is usability, and specifically usability aspects in smartphone applications. Usability testing is an evaluation to measure how well users can use a specific software system (Zhang & Adipat 2005). Furthermore, usability of a software product is often defined with three

aspects: *effectiveness, efficiency*, and *user satisfaction in a specified context of use* (International Organization for Standardization 1998).

As stated by Zhang and Adipat (2005) usability testing of applications developed for mobile devices is an emerging area. In contrast to the usability aspects of normal desktop software project, we are presented with some unique challenges. According to Kenteris et al. (2009), usability studies with mobile devices have to take limited bandwidth, the unreliability of wireless networks, and the ability to change context into consideration. These important aspects are usually not an issue in usability studies of normal computer software. Moreover, this is an area that has relatively few studies because of the novelty of mobile technology (Coursaris & Kim 2011).

The main reasons why we wanted to focus on usability in the context of smartphones are therefore: 1) the lack of previous research focusing on smartphone usability and 2) the unique challenges posed by mobile technology. We investigated this topic of usability with four experiments. These were completed to provide insight into specific usability issues of various smartphone applications.

We acknowledge that even though we have limited our research to usability in smartphone applications, it is still a fairly large and comprehensive area. To further narrow the scope of our work, we therefore selected three specific areas to investigate more closely. These areas all provide important features in regards to smartphone applications, and have provided a way for us to categorise the experiments we conducted as part of this thesis. The selected areas are: *Wireless Personal Area Networks* (WPANs), *Context-awareness*, and *Remote Information Access*.

WPAN was selected because it is an area that provides the convenience of mobility and flexibility to mobile devices (Li et al. 2005). Additionally, according to Oh et al. (2010), techniques in wireless communication that can provide high quality service has attracted considerable attention. In these situations WPANs provide one viable solution to achieve a fairly high data rate using low-power technology and it also provides convenient interconnectivity possibilities.

The second area is Context-awareness, which is highlighted as one of the important areas of research within pervasive computing (Saha & Mukherjee 2003). Moreover, Coursaris and Kim (2011) state that the area with the greatest potential for future mobile usability research is environment characteristics. By registering information about the environment the user is in, the application can automatically adjust the content.

Finally, the last area is Remote Information Access, which is a critical aspect of smartphone applications because a significant number of applications require information from external resources. Additionally, it is also an important aspect because of the rapid growth of mobile devices, which means that the demand for using the devices to access Internet resources is also increasing (Kenteris et al. 2009).

Therefore, the overall aim for this thesis is:
To investigate aspects of usability in smartphone applications related to three areas: *Wireless Personal Area Networks*, *Context-awareness*, and *Remote Information Access*.

We have introduced the overall aim for this thesis, and provided justifications for the selected research areas. The remaining chapters of this thesis are organised as follows: We continue with the literature review in chapter 2. This gives an in-depth look at the background information and relevant research areas. By providing information on related work, we are able to position our work in regards to existing research and also show the gaps in current research that we are trying to focus on. The next chapter (*3*) introduces the methodology used as part of this thesis. Finally, the experiments are explained in detail in chapter 4 (*Wireless Personal Area Networks*), 5 (*Context-awareness*), and 6 (*Remote Information Access*), before the conclusion is presented in chapter 7.

Chapter 2 – Literature Review

In this chapter we start by presenting background information on the mobile and pervasive computing areas and related topics. The content provides valuable information of the contributions that have influenced our own research as well as the entire mobile computing field. Additionally, by introducing a general view of the research area, we can position our own work and clearly state the justifications for the overall aim and objectives, presented at the end of this chapter.

We start by introducing four general research directions: *Distributed Computing, Mobile Computing*, *Pervasive Computing*, and finally *Internet of Things*. These are important areas within computing, and therefore important to include as background information. Then we present the topic of usability in smartphone applications, before introducing research areas closely related to our own work, which are *Wireless Personal Area Networks, Context-awareness,* and *Remote Information Access*.

2.1 Background

Mobile devices are integrated with an increasing number of tasks in our day-to-day activities. Not only is society very reliant on the devices themselves, but also the infrastructure. One relevant example in this context is an application developed by British Airways for handling airplane tickets. This application replaces, for many destinations, the check-in process and paper boarding passes completely[6]. A screenshot of the application is presented in Figure 2-1.

[6] http://www.britishairways.com/travel/iphone-app/public/en_gb

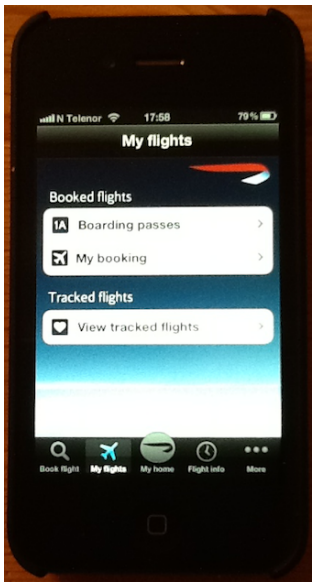


Figure 2-1, British Airways iPhone app.

Because mobile devices are usually carried around everywhere and have the capability to communicate with external resources, they provide an ideal match for these kinds of applications. It replaces other items and tasks, like boarding passes and check-in procedures, with a simple and well-integrated application that is more practical. Moreover, one does not need to print out a boarding pass or stand in queue at the check-in counter.

These features of the mobile devices are a result of many different components cooperating. Most applications communicate with various resources; these can be local to the phone, like sensors, or backend services that provide the wanted information. There are several research areas that have made significant contributions towards these technological advances we are able to use today. These areas are presented in the next section of this chapter. We will concentrate on four concepts that are particularly important when it comes to mobile devices and the integration of network communication, namely 1) *Distributed Computing,* 2) *Mobile Computing,* 3) *Pervasive Computing,* and 4) *Internet of Things.*

As presented in Figure 2-2, these majors steps in mobility and computing are related both in regards to technology and research challenges. When moving towards the right of the figure, there is a tendency to either add new problems or make existing ones more

challenging (Satyanarayanan 2001). On the figure, we have marked (with a stapled box) the most important issues that are investigated in this thesis.

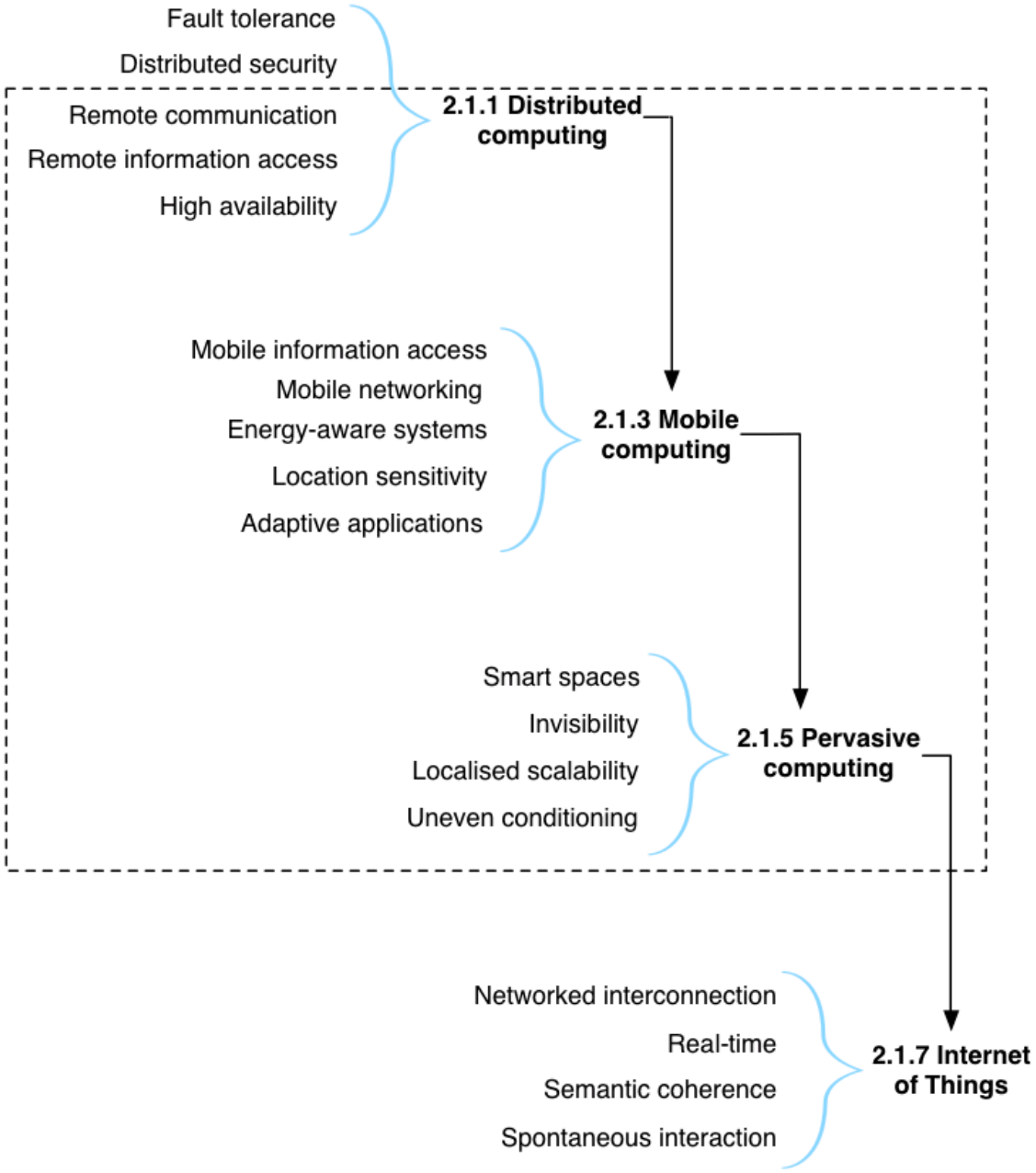


Figure 2-2, *Distributed Computing*, *Mobile Computing*, *Pervasive Computing*, and *Internet of Things* overview. Adapted from Satyanarayanan (2001) and Zhang et al. (2011).

Although new problems appear with the paradigms on the right, it is important to build on previous research findings. Distributed computing has a considerable knowledge base that helps both mobile and pervasive computing moving forward. Internet of Things is also building on knowledge learned from the previous paradigms. We will not

go into detail on the Internet of Things, as this is outside the scope of our work, but we include a general description to complete the overall picture of the four research areas.

To provide an indication of the popularity of these research areas, we have compared them in regards to search trend statistics from Google, shown in Figure 2-3. For pervasive computing we added both the search terms *pervasive computing* and *ubiquitous computing*, since these terms are used interchangeably.

The time period covered in the graph is from January 7th of 2007 to December 25th 2011. The black lines show the overall trend for each specific search term. The graph shows that mobile computing has fairly consistent search numbers. It is interesting to see that for distributed computing, the trend is falling slowly over the four years presented. This is what one would expect, since distributed computing is the oldest paradigm and the newer terms, such as mobile and pervasive computing, will attract more interest. Also, as previously mentioned, these terms overlap and take advantage of the enhancements made in previous research efforts. Pervasive computing shows a slight increase over the period, whereas the newest paradigm, Internet of Things, is increasing at a faster pace.

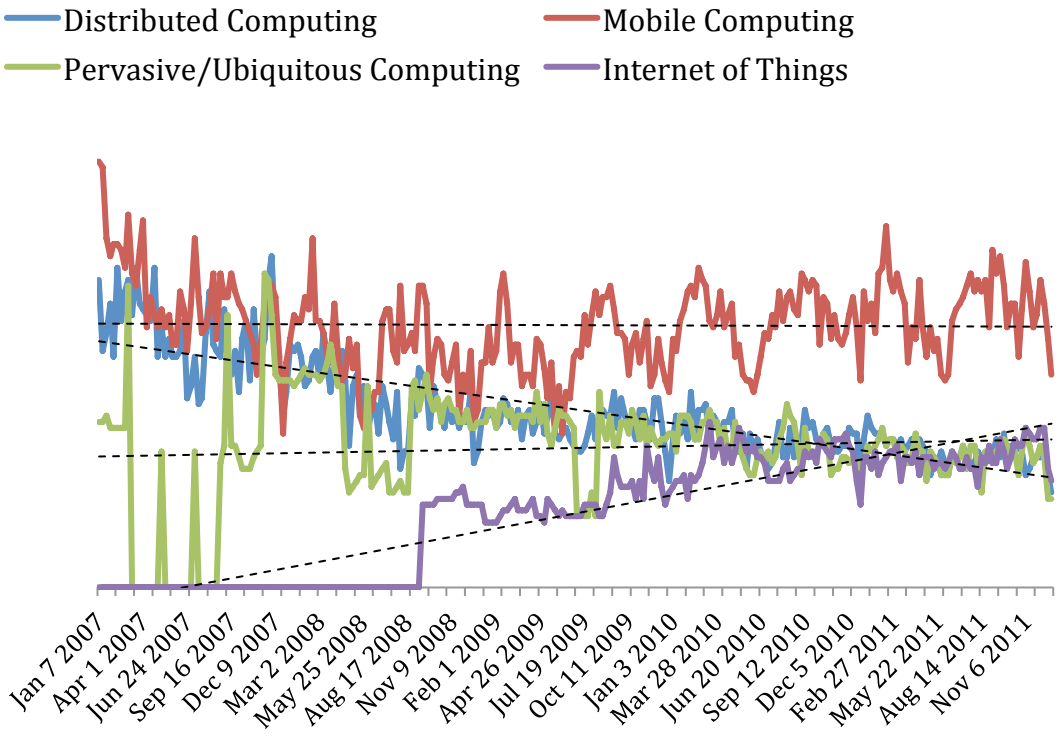


Figure 2-3, Google search trend for Distributed, Mobile, Pervasive/Ubiquitous Computing, and Internet of Things.

These statistics must be taken with a pinch of salt. They are in no way a perfect representation of the popularity of a certain technology. A potential source of errors is when a term has multiple search phrases and only one is included in the statistics. While we are fully aware of the limitations, we still believe this gives a good indication of the popularity of the terms we have presented. It is particularly interesting to see the interest for mobile computing. We believe it is due to the multiple paradigms mobile computing affects. This is especially so for distributed and pervasive computing, which are areas that overlap with mobile computing, where mobile devices are used to achieve the development of such systems.

In the next section we will go into more detail about these four general directions of research within the mobile space. By introducing these research areas we present a general introduction, before turning our attention to the main topic of this thesis.

2.1.1 Distributed Computing

Up to the late 1970s / early 1980s computing systems were usually centralised mainframe installations and access to these was done through a text terminal (Boger 2001). When the Personal Computer (PC) started to enter the mainstream market it provided local workstations. However, these devices usually provided limited communication options. Since the 1990s, when the Internet became a commercial success, the computers have been connected in a large network. With the communication options and availability of the Internet, it was possible to share resources, like servers and printers, and also enable a high degree of information sharing between users. With the Internet becoming popular, these local computers started to connect to external resources, and the initial distributed systems appeared.

A distributed system is (Tanenbaum & Van Steen 2002):
"… a collection of independent computers that appears to its users as a single coherent system".

This definition consists of two parts, which are the hardware and software. The hardware must cooperate to complete specific tasks, while the software should try to unify these hardware resources into one coherent system. Distributed computing thus

takes advantage of networked resources, trying to share information or even have different resources join forces to be able to achieve complex tasks that might take too long for just one standalone computer. Compared to implementing a single-machine system, distributed systems have their own challenges and issues.

Building on the work done by Satyanarayanan (2001), we identify four main areas of research in distributed computing. The areas are, with relevant research papers referenced: *Fault tolerance* (Borg et al. 1989) and (Maniatis & Chun 2011), *distributed security* (Gao et al. 2009), *remote communication* (Bagchi 2010), *remote information access* (Malik & Dustdar 2011), and *high availability* (Kale & Bharambe 2009). We will continue by going into more detail on these topics in the next sections.

2.1.1.1 Fault Tolerance

When creating distributed systems, one of the main challenges is error handling and fault tolerance. It is not easy to create systems that are able to keep running even though remote components have failed. Fault tolerance deals with partial failure and can in many cases provide a considerable benefit in distributed systems over a single-machine application.

The notion of partial failure is an aspect that differentiates distributed systems from the single-machine systems (Tanenbaum & Van Steen 2002). In a single-machine system a failure might impact the entire application. The concept behind partial failure is that only parts of the system are affected, while others can continue to operate as normal. A good example of this is an Internet banking website. Consider the case where the internal message system breaks down. This system handles the sending and receiving of messages from the bank to the customers. Messages can consist of content such as monthly reports and notifications that money has been transferred. If this, relatively small, part of the overall system breaks down, it is vital that other components, like the payment module, are not affected. The customer should be able to use all other features, and in the best-case scenario not even notice the failure at all.

To give a short overview of two research efforts related to fault tolerance, we include the work by Frank (1999), who describes a nested algorithm designed for updating

multi-databases with semantic ACID (Atomicity, Consistency, Isolation, and Durability). Additionally, Nouali et al. (2005) also conducted work in this area, and they proposed an alternative to the Two Phase Commit (2PC) protocol called Mobile 2PC. We will not go into detail on these research efforts because they are outside the scope of our work.

It is important that fault tolerance is present for the end-users. However, it can be difficult to implement. One design principle used by the engineers has been to create a system where remote calls are as close as possible to those of local procedure calls. RPC (Remote Procedure Calls) tries to hide most of the intricacies of message passing (Tanenbaum & Van Steen 2002). A well-known paper (*A note on distributed computing*), by Kendall et al. (1994), points out that there are several flaws in the RPC-model. There are fundamental differences between the interactions of distributed objects and the interactions of non-distributed objects. Their claim is that distributed systems that do not acknowledge and intentionally design the system with these differences in mind are doomed to failure.

It is important to acknowledge that a system is distributed. Relying on abstractions, from for example third-party libraries, can cause issues when there are failures in the system. It will also be difficult to debug the systems, since the actual network code is not implemented by the system, but in the abstraction. In our work, we utilised contract-first network protocols on a lower abstraction level than the RPC-model. This is explained in more detail in Chapter 4. The next section will introduce another challenging topic of distributed computing, namely *distributed security*.

2.1.1.2 Distributed Security

Security in distributed systems is crucial. Since data no longer is confined to one physical machine, but spread across multiple resources, it is very important to restrict access to sensitive information. In a single-machine system, the data can be protected by physically securing the hardware. In contrast, on a distributed system one will usually not have physical access to all hardware resources and the information is transferred on public networks, through routers and switches used by many other systems and end-users.

Security in distributed systems can be divided into two generic parts. The first part concerns the communication between users or processes, possibly residing on different machines (Tanenbaum & Van Steen 2002). The communication can be contained in a secure channel, which will provide the means to mutually authenticate the communicating parties, and protect messages against tampering during transmission. Harn and Huang (1994) have done research in this area, where they created a protocol for efficiently distributing session keys to establish a secure channel.

The second part of distributed security concerns authorisation, providing features to ensure that the process only get the entitled access rights to a specific resource in a distributed system (Tanenbaum & Van Steen 2002). In relevant research, Bertino et al. (1999) have created an authorisation model that can express a number of discretionary access control policies for relational data management systems.

There is a considerable amount of research focusing on security in distributed environments, such as a paper written by Georgiev and Georgiev (2001), which presents a multi-tier model for secure computing as a teaching platform. Other relevant research include Osmani and Slabbert (2009) who investigated an application-level decentralised security architecture for enterprise networks and Gao et al. (2009) who propose a distributed security MIPv6 (Mobile IPv6) model based on dynamic policies. Security is certainly a very important topic in distributed systems and computing in general, and in this section we wanted to give a short introduction to the topic. However, a more detailed investigation of distributed security, and security in general, is outside the scope of this thesis.

2.1.1.3 Remote Communication

One important concept within distributed computing is the network communication between devices. Different components of the system need to communicate, this is at the core of the distributed computing area. There are several attempts at making the development and overall design of such systems easier and more reliable.

One alternative for implementing distributed systems is the previously mentioned RPC technology. RPC was created to make distributed computation easy to implement. The ideas behind RPC were discussed in public literature as early as 1976 (Birrell & Nelson 1984). The goal was to create distributed systems that were more or less invisible to the developer. One common usage area for RPC is to build a distributed file system, where operations can be expressed as remote calls to a file server instead of having to use primitive message passing (Tanenbaum & Van Steen 2002).

In related work, Bagchi (2010) focused on RPC in mobile applications. They developed a platform supporting Inter-Process Communication (IPC) and RPC between the mobile client processes and the remote processes. Specifically, the focus was designing the software architecture for supporting distributed RPC and IPC (Inter-Process Communication) in an integrated fashion in order to support interactive mobile computing applications.

MARS, created by Cidon et al. (2011), utilised RPC in the first adaptive, online, and lightweight RPC-based remote execution scheduler, supporting multi-threaded and multi-core systems. The system uses a novel and efficient offloading decision algorithm, which takes the trade-offs between communication and computation delays and power consumption into account. MARS runs on the mobile devices and provides a lightweight design able to instantly adapt its decisions to the changing wireless resources. As previously stated, we did not use RPC in our own work. In the connectivity between the mobile devices and external resources we created services to enable easy access to the data through a pre-defined contract.

Stender and Ritz (2006), in related work, present a scenario where two organisations meet. Their idea is that both representatives at the meeting will have a mobile device and can access data from the central system. They demonstrated two different implementations of this system; one where all the application logic and data is stored at the server. The other example is a more peer-to-peer oriented architecture, where the data and logic is stored on the mobile device and then replicated to the server. They made a prototype where a mobile client prepares a quotation and can send it directly to the customer via Bluetooth.

From a distributed computing perspective it is interesting to see how the projects focused on both client-server and peer-to-peer architectures. In a peer-to-peer architecture there is usually no dedicated server. In the case of *Napster*, which was a very popular peer-to-peer system, it offered a centralised directory. This directory server knows which objects each peer was able to share (Kurose & Ross 2002). However, the actual transmission of information happens directly between the peers. There are also systems using a decentralised directory, like *Kazaa*, which distributes the content-location over the peers themselves.

2.1.1.4 Remote Information Access

The next issue we wanted to present background information on is remote information access, which provides a fundamental and important aspect of distributed computing. We go into more detail on how we have accessed information in our experiments when discussing the selected research areas later in this chapter. However, we start by presenting a general overview in the context of distributed computing.

There are several alternatives for distributing and receiving messages, for example publish/subscribe and request/reply. A publish/subscribe model consists of one publisher being able to provide event notification to all interested observers (Hohpe & Woolf 2003). In related research, Waluyo et al. (2005) have implemented a broadcasting mechanism to deliver information to a large number of mobile clients in a wireless environment. The broadcast mechanism refers to pushing information to clients using a broadcast channel. The main contribution from this work is the development of a two-tier data broadcast system using a wireless ad hoc environment. Another area related to remote information access is distributed databases. Stonebraker et al. (1996) have researched this topic and created a system called *Mariposa*, which is a wide-area distributed database system.

More closely related to our own work is the concept of organising services that offer remote information access. The Service Oriented Architecture (SOA) is a well-known model that enables the developer to reuse services without considering the programming language or any other implementation specific technology. It is based on the idea of

composing applications by invoking existing services available in the network rather than building new applications to accomplish some task (Zhou et al. 2010).

SOA promotes features such as loose coupling, service granularity, service description, and service discovery (Mukhtar et al. 2008). The concept behind SOA has been around for many years in the enterprise world; however, there have also been attempts to transfer the ideas into mobile software.

Mukhtar et al. (2008) discusses the problems of SOA in the mobile software environment. The main issue is the platform heterogeneity that can make it difficult to use the *write-once, use everywhere* approach. Developers have to take requirements of network bandwidth, software platform, and hardware capabilities into account when designing the system for a mobile platform. Mukhtar et al. (2008) have used the SCA (Service Component Architecture) model to handle these issues with mobile systems. The SCA is a realisation of the SOA model with the additional feature of defining a unified approach for heterogeneous technologies. The main idea behind SCA is the possibility to create applications across organisations, which are technology, protocol, and implementation independent.

The smartphone market today is a heterogeneous environment where the devices are spread across several platforms. For most native applications this has the consequence of requiring separate clients for each platform. Thus, a very important idea behind SOA in general is to find the commonalities, usually offered as reusable services, which can be extracted away from the client without loosing features.

2.1.1.5 High Availability

It is important not to hide the fact that the system is distributed from the developer behind an abstraction. This can cause major problems when trying to implement robust systems, where the failure in one part of the system should not affect the others. Hence, this aspect is directly related to high availability.

There are several methods for providing high availability of a distributed system. Server environment replication is one of them. Replication is used to have two identical systems running, providing both a backup and possibly increasing the performance. If one crashes the other one can seamlessly take over the responsibility until normal operation is resumed. In addition, a load balancer can create higher overall availability for the system by routing requests based on the traffic levels, utilising the hardware of each environment in parallel. Cloud computing takes high availability and server replication to a new level, where it is usually completely transparent to the developers and the users.

Replication in a mobile environment is a very important aspect of high availability, because users are often changing their position and the connection to the network is not always reliable. Ratner et al. (2001) outline the requirements for a replication service designed for the mobile context. These requirements include *any-to-any communication*, *larger replication factors*, *detailed controls over replication behaviour*, and *the lack of pre-motion actions*. They also provide a description of *ROAM*, which is a replication solution designed specifically for mobile computing and created to meet the mentioned requirements.

Khairullah and Al-Aama (2009) present another research effort focusing on data replication in mobile computing. They propose a solution to address the context-aware data replication problem. In their solution they use a shadow table, triggers, Remote Data Access, and a repetition frequency technique.

An important aspect of distributed systems is that they try to provide location transparency, where a user cannot tell where a resource is physically located in the system. This feature is achieved by assigning logical names to resources (Tanenbaum & Van Steen 2002). A well-known example is URLs (Uniform Resource Locator), generally used to identify resources available on the Internet. One major benefit with location transparency is the ability to mask server-side changes. If a resource is moved physically, the user might not even notice. Perhaps more importantly, no changes need to be implemented by the client application. The logical name is still the same, even though the location has changed. Both of these features, replication and location transparency, help provide high availability for distributed systems.

The next main research topic, which follows distributed computing, is *mobile computing*. Mobile computing builds on the body of knowledge generated by distributed systems. In fact, most mobile computing applications are distributed in some sense, where a server application receives requests from a mobile client.

2.1.2 Mobile Computing

Mobile computing started with the appearance of the laptop computers and WLANs (Wireless LANs) in the early 1990s (Satyanarayanan 2001). It emerged from the integration of cellular technology with the Web (Saha & Mukherjee 2003). Mobile computing, as the name suggests, is the process of computation on a mobile device.

Network communication is a very important part of mobile computing. Usually communication between the device and external resources is involved, where a device can request a specific set of services from back-end servers or other devices. One of the main goals of mobile computing is to offer mobility and computing power. This can be offered by providing decentralised and distributed resources on diversified mobile devices, systems, and networks that are interconnected via mobile communication standards and protocols (Kamal 2008). This aspect is very important and we will address this in more detail in several of our experiments.

Creating a distributed system for mobile devices is usually even more challenging than the standard client-server architecture, where the client is situated on a desktop computer. It is difficult to create systems where there is no fixed infrastructure available and devices can enter or leave the system at any time. Also, devices can be disconnected due to issues such as an empty battery or poor network coverage.

Satyanarayanan (1996b) identified four constraints that characterise mobile computing:

1) Mobile elements are resource-poor relative to static elements. Desktop computers and servers will usually have more hardware resources than laptop computers, smartphones, and especially lower-end mobile phones. With less hardware resources, certain considerations have to be taken when developing the systems, both on the server

and client side. There are constraints specific to the applications that are created to run on mobile devices, which would not be a consideration for normal desktop clients. Energy-consumption and varying network coverage are two examples.

2) Mobility is inherently hazardous. Security in mobile computing is considerably more difficult than in environments that have a fixed infrastructure. In mobile computing the devices are *on the move* and used in many different locations and settings. This increases the possibility of theft or to simply misplace the devices. Another challenge is privacy. Smartphones often record various data about the user and the device, for instance the location. It is crucial that this kind of sensitive information is not accessible by anyone other than the authorised users.

3) Mobile connectivity is highly variable in performance and reliability. Moving from an environment with good network coverage and bandwidth to environments that offer only low speed connections or no network connection at all is an important factor the systems must handle. There are also differences in the hardware capabilities of the devices. Some devices might only offer slow wireless connections, like EDGE, while others have the capability to connect to faster networks, with for example 4G. If the system does not recognise and adapt to these differences, it can impact the user experience. For example, if a system sends high quality video to a device with a very limited wireless connection, the result is long loading times and a poor user experience.

4) Mobile elements rely on a finite energy source. Battery technology has improved over time, but it still remains one of the main challenges for mobile computing. Although hardware components are created to be vey energy efficient, the smartphones are often equipped with large touch-screens. These large screens use a considerable amount of power. The concern for power consumption must span many levels of hardware and software to be fully effective. Applications should thus be able to adapt the content and settings on the device according to the battery level. One common example of a feature that is implemented to minimise the battery usage is the light sensor, which registers the amount of light in the room and adjusts the screen brightness accordingly.

Based on these characteristics of mobile computing, Satyanarayanan (2001) identified five main research challenges. These are *Mobile Information Access, Mobile Networking, Energy-Aware Systems, Location Sensitivity,* and *Adaptive Applications.* The next sections will go into more detail on each of these challenges.

2.1.2.1 Mobile Information Access

An important aspect of mobile computing is access to information. Mobile information access consist of challenges such as how information should be stored if the device disconnects from the network and information synchronisation when it is reconnected.

In relevant research, Chandrasekaran and Joshi (2002) created a framework for mobile information access called *MobileIQ*. The system addresses challenges in mobile environments such as mobility management and disconnection management, in addition to common issues such as content personalisation, bandwidth utilisation, download latencies, and user privacy. MobileIQ is a distributed system and service infrastructure that offers personalised mobile information access in a wireless network environment.

Satyanarayanan (1996b) presents another interesting research contribution, with a file system called *Coda*. Coda is an experimental file system, which aims to offer clients continued access to data in the face of server and network failures. In the evaluation of this system the results showed that Coda clients did experience various kinds of service failures, but the file systems were able to mask these failures effectively.

A more recent example is *Git,* a distributed version control system created in 2005 (Chacon 2009). We used git as part of our development process to store the source code and it provided a considerable benefit in cooperation between several developers. Git is a distributed version control system, which entails that the users fully mirror the repository. When working with source code, a developer is able to commit code locally, and eventually push the changes to a branch on a remote server. Since each developer will download the repository to the local machine, commits can be done offline. When the computer is connected to the network again the changes can be pushed remotely. In addition, Git has several features for handling conflicts. Conflicts can appear when several developers work on the same code and make different changes to the same files.

The system tries to merge the changes automatically. When this is not possible, a manual conflict resolution is required.

2.1.2.2 Mobile Networking

Communication between mobile devices and other resources is an important part of mobile computing. Without this capability, the device is restricted to the features and information stored locally. Within mobile networking there are challenges such as disconnection handling, varying bandwidth, limited device resources, variations on the server side, and changes due to mobility of the user and the device (Loke 2006).

To achieve mobility and network access, wireless communication is used. There are two broad types of wireless Internet access. The first is Wireless LAN (WLAN), where mobile devices transmit and receive packets from a base station. The radius of these networks is usually a few tens of meters (Kurose & Ross 2002). The second wireless technology is Wide-area Wireless Access Networks (WWAN), which usually offered by a telecommunications provider and covers a larger area.

Another technology used for wireless communication is created for close proximity networking, namely Wireless Personal Area Networks (WPAN). These networks commonly use a technology created for sending information over short distances, like Bluetooth or infrared. The short-distance networks are also used for device-to-device communication. *Ad hoc* networking is the creation of dynamic networks that are created by mobile nodes to fulfil their communication purposes (Perkins 1998).

One of the characteristics of mobile networking is that mobile devices frequently change their point of attachment to the network (Bhagwat et al. 1996). The need for a common networking protocol that can support network-wide mobility has generated research around *mobile IP*. Papers written by Bhagwat et al. (1996) and Perkins (1998) go into detail on this topic. Mobile IP is an open standard, defined by the Internet Engineering Task Force (IETF). It is based on the Internet Protocol (IP), and has the benefit that all media supporting IP also support mobile IP (Kamal 2008). Mobile IP lets the user maintain the connection to the destination without changing the device IP address even when connected to foreign networks (Chandrasekaran & Joshi 2002).

2.1.2.3 Energy-aware systems

Mobile devices face a growing demand to support computationally intensive applications like 3D graphics. However, these devices are inherently limited by processor power density and device battery life (Cidon et al. 2011). According to Fei et al. (2008), as a result of the slow battery capacity growth, it is increasingly important to develop techniques that achieve high-energy efficiency. Energy efficiency is explained as the amount of service work that the system can accomplish given a battery capacity constraint.

Flinn and Satyanarayanan (2004) show that energy-aware adaptation, the dynamic balancing of energy conservation and application quality is an essential part of comprehensive energy management solutions. The effective design of mobile software requires striking the appropriate balance between application quality and energy conservation. They measured the energy used by four applications, namely *a video player*, *speech recogniser*, *map viewer*, and *web browser*. Their study shows that energy-aware applications can substantially reduce the energy usage of mobile computers.

Fei et al. (2008) have conducted similar research, where they propose a novel dynamic software management framework. The goal behind this framework was to improve battery utilisation. The system explores quality-of-service adaptation to reduce system energy. It also employs a priority-based pre-emption policy for multiple applications to avoid competition for limited energy resources. The results after testing the system show significantly improved battery utilisation for some mobile applications, such as *video player*, *speech recogniser*, and *voice-over-IP*.

Moreover, Chary et al. (2006) present another energy-aware research effort. They have created a formalised architecture to allow systematic access and use of low cost sensors that are built into small handheld systems. In their work they explore the option to manage the power of mobile devices through the use of sensors.

The challenge in mobile computing is that the devices demands higher requirements and capabilities, while at the same time the users demand longer battery life. Although this topic is not our main focus, we look into this challenge when considering energy efficiency of different technologies for the Android platform in experiment 4 (*Customised Android Home Screen*).

2.1.2.4 Location sensitivity

Because the mobile devices provide the ability to move around while connected to the network, location sensitivity becomes another important issue. Devices are no longer bound to one location and have the ability to operate in many different environments. It is a fundamental aspect of mobile computing that users will change their location. These changes can be exploited by the system to proactively tailor services or present services according to the user's current situation (Loke 2006). This tailoring of information based on the environment provides context-awareness to the applications.

In a relevant research effort, Mantoro and Johnson (2003) explore how useful the location awareness history is for an office based low-cost context-awareness environment. They capture location awareness data into a relational database and also introduce a mechanism for accessing location awareness history using SQL (Structured Query Language) in the subject and environment's role architecture. The implementation described is the first stage of a complete system combines location sensing and deduction, camera and voice input, and sound output, releasing users from keyboard, stylus, and mouse and allowing them to interact with they computational environment more in the way they do with other people.

Another example of an application that uses location sensitivity is *OneBusAway*, created by Ferris et al. (2010). The application provided bus riders in the Seattle-area with real-time arrival information, a trip planner, a schedule and route browser, and a transit-friendly destination finder. The application is location-aware and was developed for the iPhone. It communicates with a back-end server to request information about stops in a given area, information about particular routes, and real-time arrival information for specific stops.

The area of context-awareness, which is affected by location sensitivity, is discussed in a later section when we go into detail on the selected research areas. The next section moves on to the topic of adaptive applications, which is an aspect introduced by mobile computing.

2.1.2.5 Adaptive Applications

Adaptive applications create the ability for a mobile client to sense changes in its environment, make inferences about the cause of these changes, and then react appropriately (Satyanarayanan 1996b). By integrating the applications with adaptive behaviour, they increase the predictably and the ability to run efficiently under a broad range of conditions. Additionally, Gross et al. (1999) state that adaptive applications are necessary for efficient execution and predictable response times.

Within this research area, Rasche et al. (2005) has implemented an adaptation framework called *Adaptive.NET*. It includes a monitoring infrastructure, a reconfiguration platform and tools for building adaptive applications. They used a proof-of-concept application to evaluate the connector architecture and study interoperability of Java, CORBA, and .NET objects.

Gross et al. (1999) have categorised adaptive applications in three dimensions. The first dimension they discuss is *choice of resources*. This is where the application is compiled for a variable number of nodes and the actual number of nodes for execution is determined at runtime. The second dimension is *time of adaptation*, where an application can be adaptive with regards to the number of processors or nodes it can use. The application can be created to only allow adaptation on start-up time, which is the simplest solution. A more complex scenario is when the application has the ability to adapt at runtime. Finally, the *interface between the application and the runtime systems* provide the last dimension. Applications need information on the status of the environment. In protocols based on implicit feedback the receiver monitors the incoming data stream and uses this stream to derive information about network conditions. A good example of this is TCP, where dropped packets are viewed as a sign of congestion, and the sender responds by reducing its rate. In contrast, with explicit feedback, some entity inside the network provides explicit information about network

conditions to sender. Adaptive applications have opened up the opportunity for a new kind of applications, which can take context into account. We go into more detail on context-awareness in section 2.2.1.3.

The next section introduces pervasive computing. Mobile computing deals with the implementation of many of the core networking ideas for pervasive computing, for example roaming between data networks (West 2011). Although pervasive computing uses many aspects of mobile computing, they represent two separate fields of research. According to Saha and Mukherjee (2003), pervasive computing is a superset of mobile computing. Whereas mobile computing deals with *anytime anywhere* that is a reactive approach to information access, pervasive computing takes a more proactive approach with the *all the time everywhere* goal. Moreover, in addition to mobility, pervasive systems have additional requirements such as *interoperability, scalability, smartness,* and *invisibility*.

2.1.3 Pervasive Computing

The initial move towards pervasive computing started in the 1970s, when the PC brought computers closer to people (Saha & Mukherjee 2003). However, it was not until the early 90s the idea of truly ubiquitous/pervasive computing started to take shape. In 1991, Mark Weiser wrote about what he envisioned the *computer for the 21st century* would be like. The work was based on research done at Xerox, where staff had working prototypes of what they called *ubiquitous computing.* The terms *ubiquitous* and *pervasive* computing are used interchangeably throughout this thesis. The idea was that computers blend into the environment, to the point where people would no longer notice their presence. Pervasive computing thus consists of a new class of devices that make information access and processing available for everyone from everywhere at any time.

Mark Weiser (1991) stated that: *"Prototype tabs, pads and boards are just the beginning of ubiquitous computing. The real power of the concept comes not from any one of these devices – it emerges from the interaction of all of them*". When looking at the current state of mobile computing it is quite impressive how this vision is getting increasingly closer to reality. Mobile phones, tablets, and e-readers are now very much a part of our everyday lives. Pervasive computing provides a rich diversity of

applications and can connect to worldwide networks, thereby providing access to a wealth of information and services. Pervasive computing builds on mobile computing, but adds characteristics such as transparency, application-aware adaptation, and an environment sensing ability (Hansmann et al. 2000).

In many situations, mobile devices require the ability to communicate with each other. This leads to a close connection of both distributed and mobile computing with pervasive computing. Pervasive systems require support for *interoperability*, *scalability*, *smartness*, and *invisibility* to ensure that users have seamless access to computing (Saha & Mukherjee 2003). All of these features and advantages create new challenges that did not exist before (Hansmann et al. 2000). In some cases, research done in distributed and mobile computing can be applied directly to the pervasive computing area. For others, the demands of pervasive computing are sufficiently different that new solutions have to be sought (Satyanarayanan 2001).

One important challenge in this research area is privacy. Privacy in pervasive computing environments can be a very difficult problem to solve (West 2011). According to Satyanarayanan (2001), privacy in pervasive systems is greatly complicated when moving from distributed and mobile computing. Tracking location and other possibly sensitive data resources can cause considerable challenges. Not only can this cause a problem technically, but also from a user acceptance perspective.

Another challenging aspect is in the hardware domain. Satyanarayanan (2001) and West (2011) mention the issue that mobile devices are becoming increasingly smaller and placing severe restrictions on battery capacity. One example of research done in this field is a system developed by Parkkila and Porras (2011). They use a method called *cyber foraging*, where the idea is to use nearby computers in the same local network to handle the heavy tasks. The issues previously mentioned, privacy and battery limitations, are outside the main scope of this thesis.

We present four main topics in the next sections, based on the research of Satyanarayanan (2001): *Smart Spaces, Invisibility, Localised Scalability,* and *Uneven Conditioning*.

2.1.3.1 Smart Spaces

A smart space, also called smart environment, can be identified as one that is able to acquire and apply knowledge about the environment and its inhabitants in order to improve their experience (Cook & Das 2007). According to Satyanarayanan (2001), a space may be an enclosed area such as meeting room or a corridor, or even a well-defined area such as a courtyard. With pervasive computing we can take advantage of the location information and adapt the content provided to the user. The environments can be saturated with sensors and handheld devices to smartly serve the users (Zhang et al. 2011). By using these sensors and devices, one can adapt the content and behaviour based on the environment the users are in (Cook & Das 2007).

There are several research efforts focusing on creating these smart spaces/environments. Jaimes and Miyazaki (2005) present an experimental Smart Conference Room, containing cameras, microphones, displays, and capture devices. Closely related, Chen et al. (2004) created a smart meeting room. The system provided features such as presentation, lighting control, and greeting service. They used Bluetooth to register the participants that were in the meeting room. Similarly, Ahmed et al. (2005) developed a meeting detector model that can determine the beginning and end of meetings. The general idea is to free meeting participants from worrying about distractions like mobile phone ringer mode and sending meeting minutes. Parviainen et al. (2006) also included the use of speech detection, and the system was utilised in a lecture room setting. They developed a source localisation system to identify the participant who is speaking and at which time. The next section continues with research on pervasive computing, where we focus on *invisibility*.

2.1.3.2 Invisibility

One of the major challenges within the domain of pervasive computing is invisibility (Cook & Das 2007). The concept of invisibility is exactly the idea behind pervasive computing, as devices blend into the background. Mark Weiser (1991) stated: "*They* (the devices) *weave themselves into the fabric of everyday life until they are indistinguishable from it*."

As these mobile devices are becoming ubiquitous in our lives, they are increasingly more invisible to us (Hansmann et al. 2000). We get so used to having them around for specific tasks, that the operating system and underlying technology are no longer important. The important fact is that they are created to do a specific set of tasks that is useful to the consumer.

West (2011) presents one example of invisibility, where he compares pervasive computing to writing. Writing was previously used solely as a tool to record and relay information. Today, writing is everywhere. On billboards, clothes, cars, buildings, and so on. Even though writing surrounds us, it does not steal all our attention. The writing lies in the background of our lives, but we use it regularly without thinking. It is not the device or technology that should be occupying the users' attention, but rather the task itself.

In research related to creating invisible systems, Waller and Johnston (2009) introduce two ways that pervasive computing can be truly available. Firstly, through manipulating the space of possible actions, like how smart spaces can adjust certain aspects in a room based on for example the number of people present. Secondly, there must be an indication of the possibility for action, involving the ability to let the user know what actions are possible in that place and at that moment.

Context-awareness, smart environments, and invisibility are closely integrated themes. A smart environment using context-awareness makes the technology and devices more invisible for the user. More tasks are done automatically, and the user interfaces can be adapted to the current situation and location that the user is currently in. The environment and objects in it must be able to tune themselves without distracting users at a conscious level (Saha & Mukherjee 2003).

2.1.3.3 Localised Scalability

Localised scalability is another aspect of pervasive computing, and it integrates crucial factors of both distributed and mobile computing. Information sources and destinations are widely distributed in the world of pervasive devices (Hansmann et al. 2000).

Localised scalability builds specifically on the idea that scaling should be based on proximity to a resource. It is important that the density of interactions declines as one moves away, otherwise, the computing system can be overwhelmed by distant interactions that are of little relevance (Satyanarayanan 2001). Using context and smart spaces help provide useful information, and enables scalability based on the proximity to a resource.

An important issue when dealing with scalability is the aspect of connectivity. According to Hansmann et al. (2000) connectivity is a fundamental paradigm within pervasive computing. To enable mobile devices to communicate with each other, it is important to provide a common standard. With open standards one opens the possibility for devices, applications, and data to be exchangeable. There are several standards available, with especially XML providing a widely accepted meta-language (Peltzer 2003). By integrating open standards for communication between devices, one can provide scalability and easily share data between computing resources.

The next section is also highly dependent on open standards. Uneven conditioning is the challenge where environments and devices have a difference in *smartness*. Providing open standards are critical to mitigate the problems when dealing with uneven conditioning.

2.1.3.4 Uneven Conditioning

There is a shift from the universal computers to diversified devices, which aim at meeting requirements of a group of users for a specific purpose (Hansmann et al. 2000). Consumers today have many different devices for specific tasks, from the small sized smartphones, to the larger tablets, all optimised for a particular context and environment. The heterogeneous environments that exist today, with a large variety of devices, creates important research challenges (Saha & Mukherjee 2003).

There are also a wide variety of device form factors. Display capabilities can often differ from device to device. Additionally, the set of available data input mechanisms such as touch-based, stylus, function key, speech or handwriting recognition, make the applications device specific (Hansmann et al. 2000). All of these devices need to

cooperate, and the pervasive system should hide this issue from the end user. The goal is to not bother the user with these technical details. It is not important for the user to know whether a system is programmed in Java and uses XML to send and receive information, what is important is the task one wants done. Masking the uneven conditioning is thus important to maintain the invisibility of the technology used in the system (Satyanarayanan 2001).

The Android platform is one example of where fragmentation is occurring due to the sheer number of device models. Devices have different capabilities, and not all devices will be equipped with the same hardware resources. Also, there is a fragmentation of Android versions, where newer versions of Android will provide more features than older versions. Table 2-1 presents data based on the number of Android devices that have accessed the Android Market from February 1st to February 14th, 2012.

Android version	Codename	Distribution
1.5	Cupcake	0.6%
1.6	Donut	1.0%
2.1	Éclair	7.6%
2.2	Froyo	27.8%
2.3 – 2.3.2	Gingerbread	0.5%
2.3.3 – 2.3.7		58.1%
3.0	Honeycomb	0.1%
3.1		1.4%
3.2		1.9%
4.0 – 4.0.2	Ice Cream Sandwich	0.3%
4.0.3		0.7%

Table 2-1, Android Platform Versions[7].

As Table 2-1 clearly demonstrates, there are many different versions of Android currently being used by consumers. Versions 2.2 and 2.3.3-2.3.7 have a combined total of 85.9% of the devices recorded. These older versions do not have the same features,

[7] http://developer.android.com/resources/dashboard/platform-versions.html

such as the Android Beam that offers NFC-based (Near Field Communication) sharing[8], included from Android version 4 onwards.

Developers are required to address this issue, without the need for end-user involvement. The Android SDK includes features to deal with these fragmentation issues, like the ability to express a minimum required version of Android for the device installing the application. There is also the possibility to disable features if the device installing the application does not support them. This works by testing for a certain feature, for example the existence of a sensor component. If it does not exist on the specific device, the feature is turned off and not used. This way the user does not get involved and will not have to do any manual configuration.

Smart Spaces, Invisibility, Localised Scalability, and *Uneven Conditioning* are all important issues in pervasive computing. Moving on from pervasive computing, the term *Internet of Things* has appeared. The following section gives a brief introduction to this topic.

2.1.4 Internet of Things

Internet of Things is a networked interconnection of everyday objects and is rapidly gaining popularity, thanks in part to the increased adoption of smartphones and sensing devices (Patel et al. 2011). Zhang et al. (2011) categorises the Internet of Things with four main challenges:

1. **Networked interconnection**, all physical objects must be mapped into the Internet.
2. **Real-time**, requires real-time searching techniques.
3. **Semantic coherence**, recognise objects accurately and support a lightweight representation to accommodate semantics across smart spaces.
4. **Spontaneous interaction**, handle each interaction in an efficient manner such that the entire system is scalable and real-time.

It is evident that Internet of Things builds on pervasive computing, where words like *smart spaces, scalability,* and *interconnection* are all described in the previous sections.

[8] http://developer.android.com/sdk/android-4.0-highlights.html

However, like the other research directions, Internet of Things adds its own challenges and opportunities.

One of these issues is scalability. As reported by BBC[9], the number of devices connected to the Internet is expected to reach 15 billion by the year 2015. This increase in network connected devices causes problems for the infrastructure, with issues like IPv4 addresses quickly running out. Even though the new IPv6, that was approved as early as 1998, is currently being pushed out to companies, it is a slow process. IPv6 is a network layer protocol, which was designed to increase the address space for nodes within the Internet (Perkins 1998).

The Internet of Things is an up and coming area, which will undoubtedly attract more research interest in the near future. The popularity is increasing, as shown with the Google trend graph (Figure 2-3). This area of research is, however, outside the scope of this thesis.

These four areas of computing (*Distributed Computing, Mobile Computing, Pervasive Computing,* and *Internet of Things*) were presented as background information, with relevant research efforts included. We wanted to provide a general view first because it introduces important contributions we are building on. The next section presents topics more closely related to our own work, namely usability and then moving on to the specific research areas selected.

2.2 Usability

Usability testing of software artefacts is important to understand the system from the perspective of the users. It can find problems with existing features and/or identify aspects that work particularly well. Usability testing is used to get feedback from relevant users, and in software this is usually in regards to a created artefact. It is important to understand what the user wants from the product and if they are satisfied with the tested features. Therefore, the general goal is to make the product easier for people to use or proving that it is easy to use (Krug 2009). Usability of a software

[9] http://www.bbc.co.uk/news/technology-13613536

product is usually defined as *effectiveness, efficiency,* and *user satisfaction in a specified context of use* (International Organization for Standardization 1998).

Zhang and Adipat (2005) have identified nine generic usability attributes that are important aspects of testing software applications:

1. **Learnability**, how easily can the users finish a task the first time using an application (*ease-of-use*).
2. **Efficiency**, how fast can users accomplish a task while using an application. The users should have some experience with the system.
3. **Memorability**, level of ease with which users can recall how to use an application.
4. **Errors**, the mistakes a user makes.
5. **User satisfaction**, attitude of users toward using the application.
6. **Effectiveness**, completeness and accuracy with which users achieve certain goals.
7. **Simplicity**, the degree of comfort with which users find a way to accomplish tasks.
8. **Comprehensibility**, how easily users can understand content presented in the application.
9. **Learning performance**, effectiveness of learning by using the application.

These provide some general guidelines on what a usability study can contain. It is not meant as a checklist that each study should complete, but a few pointers that can be useful to include in the usability evaluation. We include several of these suggestions, such as the efficiency and user satisfaction, in our own experiments that are presented in later chapters.

While there are some obstacles in usability testing, such as such as the ability to change the mind-set of developers towards a more user-centred focus (Bak et al. 2008), it has provided considerable benefits. Evaluating usability is documented to provide economical benefits with increased sales, increased productivity, and decreased training costs and need for user support (Bak et al. 2008).

We wanted to focus on usability within smartphone applications, and the next section continues by introducing this area of usability, before presenting the selected research areas that we have focused on within this general topic.

2.2.1 Usability and Smartphones

Research of usability within software has focused on a range of different areas, such as *mobile commerce* (Venkatesh et al. 2003), *games* (Korhonen & Koivisto 2006), and *websites* (Petrie & Power 2012). While usability for software in general is an important area, we wanted to look closer at usability in smartphone applications.

There are two main reasons for selecting this area. Firstly, there is still a lack of existing research focusing on usability within mobile applications (Coursaris & Kim 2011). Secondly, there are new constraints introduced by mobile devices, such as limited bandwidth, the unreliability of wireless networks, and environmental factors (Zhang & Adipat 2005).

Based on the work by Zhang and Adipat (2005), we have identified four important usability challenges:

- **Mobile context**, the ability to sense the environment and adjust the content based on the situation. This is also called *context-awareness*.
- **Multimodality**, combining various modes, such as providing multiple input mechanisms. For example using voice and touch-based input.
- **Connectivity**, how to deal with various network conditions.
- **Restrictive data entry methods**, small buttons and labels limits the users effectiveness and efficiency in entering data.

According to Zhang and Adipat (2005), these challenges for examining usability of applications are unique to mobile devices. We will look closer at these challenges through the work conducted as part of our experiments.

Even though we have focused on a particular aspect of usability, it is still a comprehensive and large area of research. Therefore, we wanted to further narrow the scope of our work to three specific research areas to investigate in regards to usability.

2.2.1.1 Research Areas

To categorise our experiments we selected three research areas, in which usability for smartphones are investigated. Based on the previously presented background information (*Distributed Computing, Mobile Computing, Pervasive Computing,* and *Internet of Things*) and the usability aspects introduced in this chapter, we have identified three important research areas to investigate.

These research areas are: *Wireless Personal Area Networks, Context-awareness,* and *Remote Information Access.* Figure 2-4 gives an overall view of how they integrate with smartphones.

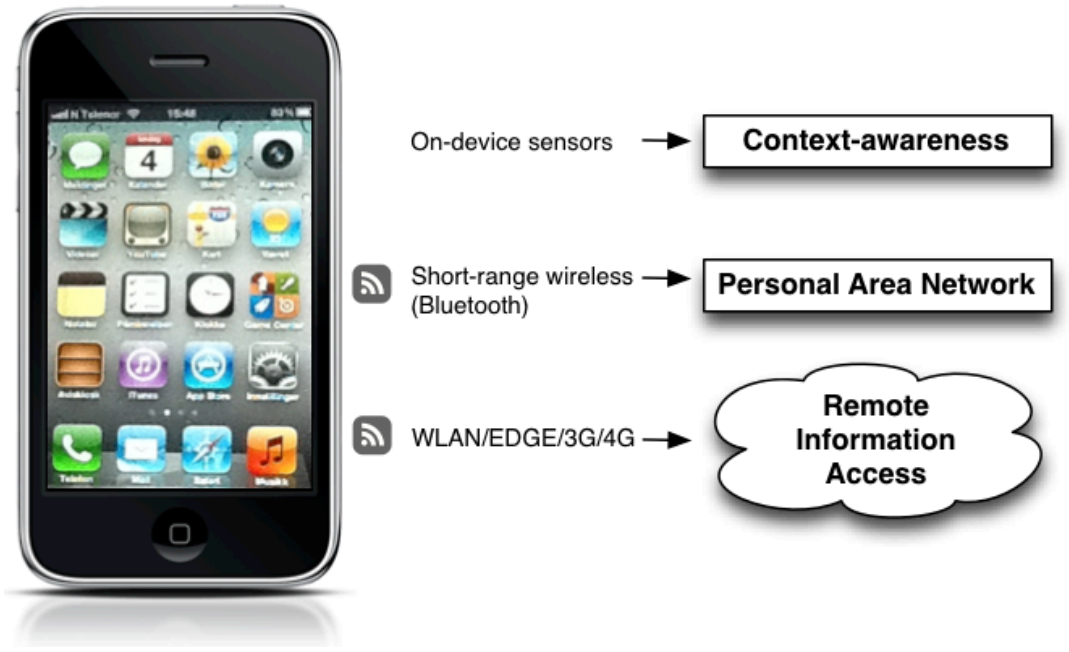


Figure 2-4, smartphone applications and the selected research areas.

In the next sections we go into more detail on these research areas, and present research efforts more closely related to our own work, starting with WPAN.

2.2.1.2 Wireless Personal Area Networks

Wireless Personal Area Networks (WPANs) have emerged as an alternative to both Local Area Networks (LANs) and Wide Area Networks (WANs) (West 2011). WPANs make it possible for communication between devices that are at short distances from each other (Kamal 2008). Bisdikian (2001) explains that the personal connectivity space resembles a communication bubble that follow people around, and that people can connect to other devices that enter this bubble.

We selected WPANs as one of the research areas because it provides the convenience of mobility and flexibility (Li et al. 2005). According to Oh et al. (2010), techniques in wireless communication that can provide high quality services have recently attracted considerable attention. In these situations WPANs provides one viable solution to achieve high data rate and provides the possibility for significant interconnectivity. This is further verified due to the increase in the number of devices using wireless network in small indoor areas (Oh et al. 2010). Many of the devices used today include short-range communication functionality. The short-range communication is usually radio frequency based, in contrast to the infrared technology used previously, since this allows high bandwidth non-line-of-sight communication (Moors et al. 2008). According to Moors et al. (2008), while the communication technology may change, the applications that demand this functionality are enduring, so they expect short-range radios to persist into the future.

There are a considerable amount of research efforts utilising WPANs in various configurations and environments. Fernández et al. (2006) present one example of work using Bluetooth for short-range communication. They propose a new wireless communication application using Bluetooth and push technology. The *Moviltooth* application delivers personalised information to the users. The server saves user profiles and only sends information that is interesting according to the active profile. Since there are many different display sizes, the server adapts the content of the message to the specific mobile phone types. This application is clearly an interesting idea, especially the fact that it does not need any local installation. Not only does this make it easier for the users, it also attracts the attention of more people.

In similar research, Chen et al. (2004) used Bluetooth to create a smart meeting room. The Bluetooth technology registered the participants that were attending the meeting. They conducted a demonstration of the implementation and received positive feedback. González-Castaño et al. (2005) present another research effort using Bluetooth and WPANs. They have created a system used in a museum where it delivers information to the visitors, with an implementation of a Bluetooth Location Network. The visitors use PDAs with external Bluetooth modems and get webpages sent to the device via a Bluetooth push protocol. Their research pointed out that there are enough visitors to justify the need of the system in the museum, and an analysis that showed that it is of practical interest.

Context-aware systems can also benefit from using short-range technology. Kwon et al. (2005) present the use of Bluetooth integrated in a context-aware system, where they created a personalised reminder system called NAMA (Need Aware Multi-Agent). The system is aware of the URL address used and interprets the content as context data. With this information it cooperates with a Web Service matchmaker to help the user purchase a certain product. The system uses proximity information to inform users about products they may find interesting, and alerts the user through a mobile device.

Another WPAN and Bluetooth-based system is called *Just-for-Us,* which provides a publicly available mobile Web Service (Kjeldskov & Paay 2006). The system uses a pervasive network of sensors to generate a digital layer of information about people, places, and activities adapted to each user's physical and social context. The system uses Bluetooth beacons to approximate other users' locations, and scans for other Bluetooth-enabled mobile devices to identify nearby friends.

Feldbusch et al. (2003) presents a different approach, with their implementation of the Bluetooth remote control. Specifically, they created a universal remote control system. One of its main purposes was to control consumer devices through the Internet. The system is based on a protocol called BTRC (Bluetooth Remote Control System). It is a simple request/response based protocol that aims at providing a simple, uniform and scalable way for information exchange between devices. It also provides services that require a secure connection with an encrypted channel and a certainty that the

commands are coming from an authenticated client. The idea of a flexible remote control system is an aspect we wanted to develop further in our own work.

We have used Bluetooth for two purposes in our experiments, namely as communication technology and proximity sensing, i.e. registering devices in the same room. In addition to proximity, there are several other options for smartphones to sense the environment. The next section introduces the area of *Context-awareness*, which makes it possible to use this information to enhance the user experience.

2.2.1.3 Context-awareness

We start this section by presenting definitions on both context and context-awareness in computing in order to establish a common understanding of the terms. Dey (2001) defines context as "*any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*". A computing system can thus be identified as context-aware if it uses context to provide relevant information and/or services to the user, and relevancy depends on the user's task (Dey 2001).

We selected context-awareness because it is highlighted as one of the most important areas of research within pervasive computing (Saha & Mukherjee 2003). Moreover, Coursaris and Kim (2011) state that the area with the greatest potential for future mobile usability research is environment characteristics. The ability to sense and adapt to the environment is a crucial aspect of context-awareness. According to Venkatesh et al. (2003), a key success in wireless context, such as systems running on mobile devices, is the ability to present content to users in a customised fashion.

Kamal (2008) explains that context refers to the interrelated conditions in which a collection of elements, records, components, or entities exists or occurs. When these are considered in regards to how they relate to each other and to the environment, they can provide a wider meaning. Accordingly, context-awareness in computing systems can be defined with three basic functionalities, as identified by Loke (2006). Figure 2-5 provides an overview of the context-aware system architecture.

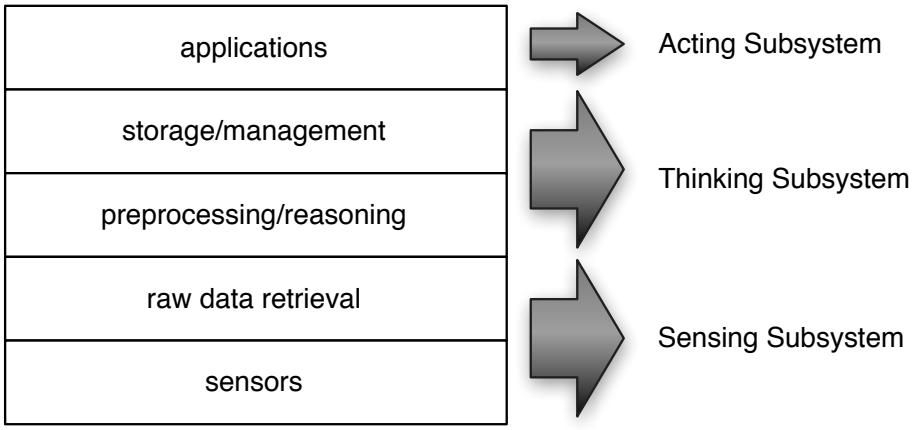


Figure 2-5, context-aware system architecture (Loke 2006).

The three identified functionalities are (Loke 2006):

1. **Sensing**, sensors provide a means to acquire data or information about the physical world or some aspect of the physical world. This knowledge can be used to determine the actions that are the most appropriate to the physical situation. Different sensors are usually used to collect context information, such as GPS, Bluetooth, and RFID.
2. **Thinking**, utilise the collected data and make sense of it.
3. **Acting**, take the appropriate actions.

We have further divided context-awareness into two main areas, namely *context-aware applications* and *smartphone sensors*.

Context-aware applications

According to Loke (2006), context-aware systems are sensitive to the real world and attuned to the purposes for which they have been developed. A context-aware application tries to understand the situation in which a device operates, resulting in better and more efficient computing strategies (Kamal 2008). There are two main ways to use context in computing systems (Chen & Kotz 2000). The first is to automatically adapt the behaviour according to the discovered context, called *active context*. The second way, called *passive context*, is to present the context to the user on the fly and/or store the context for later retrieval.

Within context-aware applications, it is often useful to hide the details of how a context type is acquired (Loke 2006). One example is systems that are able to find the location for a mobile device using several different methods, such as GPS and WIFI location tracking. Both of these technologies return the same type of information, but with different accuracy.

Location tracking is a widely used context source (Loke 2006), and a considerable amount of research has focused on using location. In addition to the research previously presented by Ferris et al. (2010), with the *OneBusAway* project, another example is a system called *Place-Its* (Sohn et al. 2005). Place-Its is a location-based reminder application that runs on mobile phones. The idea is to study how people use location-aware reminders throughout their daily lives.

Interesting work has also been done to utilise context-awareness in the healthcare domain. Kjeldskov and Skov (2007) have explored ubiquitous technologies, where they studied two specific features. Firstly, they created an Electronic Patient Record (EPR) system distributed across desktop and laptop computers throughout a large hospital. Secondly, they added an extension to this system in the form of a context-aware mobile computer terminal prototype called *Mobileward*. This application provided context-awareness in the sense that it recognises the location of the nurse and presents information and functionality accordingly. Their results indicated that the usefulness of a ubiquitous computing environment, supporting work activities in healthcare, benefit from context-aware mobile information access.

In addition to location, other context sources are available, such as time, current activity of the user, and data from sensors (Loke 2006). According to Schimidt et al. (1998) applications will benefit significantly from a wider notion of context. In this regard, Raento et al. (2005) have developed a software platform to help developers to build applications that integrate into both existing technologies and the everyday lives of the users. *ContextPhone* is thus a software platform consisting of four interconnected modules that runs on Symbian. By using this library, developers are able to easily add new components such as context data sources and sensors, and build new applications without rebuilding the entire system. *ContextPhone* can be used to sense, process, store,

and transfer context data. Their software supported four sensor types, namely location, user interaction, communication behaviour, and physical environment.

As explained by Jones (2005), a central theme in achieving context-aware information access is the combination of information retrieval with multiple dimensions of available context data. Various context sources can be exploited using a variety of technologies to create new and exciting possibilities for information access. Huebscher et al. (2006) described this in their paper, where they used context data of the same context type in combination to create higher quality results. They investigated the case where multiple providers of the same type of context in the home can be combined.

Smartphone Sensors

When working with smartphones, high levels of interaction are difficult and it is thus important to exploit whatever information is available to maximise retrieval precision (Jones 2005). Sensors provide an important aspect of context-aware systems because the sensors collect the data that is consequently used for computing the context-aware content. Sensors are electronic devices that have the ability to sense the physical environment (Kamal 2008). In mobile devices, sensors facilitate interaction between the device and the surroundings. A combination of sensor data, information on the device, and information from other devices or elements embedded in the infrastructure, can be used to create proactive features in the applications (Glesner et al. 2004). Due to advances in mobile technology, it is now possible to embed sensors in mobile devices at a low cost (Schmidt et al. 1998).

In work closely related to our research using sensors and context-awareness, Murao et al. (2011) captured data for 27 different kinds of gestures. They used a mobile device with nine accelerometers and nine gyroscopes. Their experiment investigated the effects on recognition accuracy when changing the number and positions of sensors, and additionally the number and kinds of gestures. Their findings show that the use of multiple sensors and sensors positioned at specific positions affected the accuracy.

As already mentioned, using multiple sensors can provide useful context information. Gellersen et al. (2002) investigated how to use a multi-sensor setup to achieve context-

awareness in mobile devices and smart artefacts. Their experiments included three projects; 1) development of an awareness module for augmentation of a mobile phone, 2) *Mediacup*, exemplifying context-enabled everyday artefacts, and 3) the *Smart-Its* platform for aware mobile devices, ad hoc sharing, and collective awareness across connected devices. They explored the prototypes in various applications to validate the multi-sensor approach to awareness. Their conclusion is that using sensors are a viable approach to obtain context representing real-world situations and context that captures interaction with everyday artefacts.

In our work we use normal smartphone devices. This is similar to Bhoraskar et al. (2012), who created a system they call *Wolverine*, which uses input from smartphones. They specifically collected data from the accelerometer, GPS, and magnetometer. Wolverine is a non-intrusive road condition estimation system that can identify braking events, where frequent breaking indicates congested traffic conditions, and/or bumps on the roads, which characterise the type of road.

(Kwapisz et al. 2011) presents another interesting research effort that uses standard smartphone sensors to register events. They created a system that used the accelerometer of smartphones to perform activity recognition. This was implemented to identify physical the type of physical activity, such as sitting, walking, and running, a user is performing.

We now move on to the final research challenge we wanted to address in as part of this thesis, namely *Remote Information access.* Specifically, we have integrated context-aware systems with external services to provide features such as flexibility and scalability. The next sections introduce *Remote Information Access* and presents important research conducted in this area.

2.2.1.4 Remote Information Access

The final research area we want to investigate is Remote Information Access. This is a critical aspect of smartphone applications, because most applications are required to retrieve information from an external resource. Additionally, it is also important

because of the rapid growth of mobile devices the demand for using these devices to access Internet resources is also increasing (Kenteris et al. 2009).

In our work, we specifically wanted to focus on the integration of smartphones and the use of external services. These services were created to provide important functionality to the application running on the phone. An important aspect in this area of research is SOA (Service Oriented Architecture).

There are two key concepts that SOA is built on. Firstly, it is usually made up of a centralised list of all available services. Secondly, there is an interface provided, commonly a WSDL (Web Service Description Language) or WADL (Web Application Description Language), describing the agreed upon contract for the service (Hohpe & Woolf 2003). Web Services, which is commonly used in combination with SOA, is usually developed as a request/response model, where a client calls a remote service and expects a result in return. It is commonly used with XML or REST-based (REpresentational State Transfer)[10] services that are sent over the HTTP protocol.

A considerable benefit in creating common services is the ability to reuse features across applications. According to Dan et al. (2008) this provides three key benefits:

1. **Improving agility** of solutions by quickly assembling new business processes from existing services to meeting changing marketplace needs.
2. **Reduction of cost** by not just avoiding duplication of code for enabling similar business functions across multiple business processes, but also throughout the SOA life cycle spanning service deployment and management.
3. **Reducing risks** by reusing well-tested code and runtime environments.

To implement the services used as part of our implemented systems as part of our experiments, we used cloud computing. Cloud computing refers to the applications delivered as services over the Internet and the hardware and systems software in the datacenters providing these services. The idea is built around an economy of scale, where the ultimate goal is to provide more resources, better scalability and flexibility for less money (Binnig et al. 2009). Cloud computing indicates a movement away from computing as a product that is owned and towards computing as a service (Khajeh-

[10] http://www.oracle.com/technetwork/articles/javase/index-137171.html

Hosseini et al. 2012). Service in this context is a concept that deals with the utilisation of reusable fine-grained components across a vendor's network, and cloud-based services are usually billed using a pay-per-use model (Vaquero et al. 2008).

The NIST (National Institute of Standards and Technology), provide the following definition of cloud computing (NIST 2011): *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".*

Although cloud computing is in its infancy (Elsenpeter et al. 2009), the technology cloud-based systems are built and rely on are getting to the point where they are mature enough to reach mainstream acceptance. This is validated by the fact that large companies, such as Microsoft, Google, and IBM all have initiatives relating to cloud computing technology and platforms (Mei et al. 2008). Some of the popular cloud-based services include *Microsoft Azure*, *Amazon EC2*, and *Google App Engine.* However, it is important to note that cloud computing is still an emerging platform, and there is a lack of defined standards, tools, and methods to efficiently use it (Calheiros et al. 2011).

The general idea of cloud computing is not new, however it does introduce some novel concepts (Mei et al. 2008):

1. Illusion of unlimited computing resources.
2. Elimination of an up-front commitment by Cloud users. The ability to start small and increase hardware resources based on need.
3. Pay for use of computing resources on a short-term basis. No need for a big up-front investment in hardware.

We have given an introduction to what cloud computing is; moving on, we present the benefits that cloud computing offers. Elsenpeter et al. (2009) mention a few good examples, such as *reduced cost*, *increased storage*, *automation*, *flexibility*, *better mobility*, and *better use of IT staff*. Furthermore, there is a growing interest in cloud computing, as highlighted by Gartner (2011), which included cloud computing in the

top-10 list of strategic technologies for 2012. Gartner stated that enterprises are evolving from the initial phase, where they were trying to understand how to utilise cloud computing, to making decisions on selected workloads to implement on cloud services. Additionally, there is a consideration on the need for private clouds. Gartner predicts that a hybrid model, which consists of a combination of both public and private clouds, will be a focus in 2012.

For cloud computing providers and server-side development, there is a need to look closely at how the integration between cloud computing and mobile devices, particularly smartphones. In the mobile computing domain, the rapid advancements have enabled a new generation of cloud-based and context-aware mobile applications (Paniagua et al. 2011).

Cloud-based service providers and developers are looking towards the mobile domain, having their expectations focused on the access and consumption of services from mobile devices (Paniagua et al. 2011). Hence, integrating an application running on a mobile device with cloud computing services is becoming an increasingly important factor. Moreover, high bandwidth wireless networks have become ubiquitous and are being used to connect mobile devices to the cloud. Potentially, by utilising such connectivity to offload computation to the cloud, we could greatly amplify mobile application performance at minimal cost (Cidon et al. 2011).

In our research, we see the potential for a comprehensive integration between smartphones and cloud computing service, such as services that are tailored toward the current environment and the context of the devices. By implementing this we can take advantage of the scalability and flexibility of a cloud-based environment. These are explained in more detail in Chapter 6.

In other cloud-related research, a system called *COSCA* has been created that is a cloud-based platform attempting to provide a component-based design with emphasis on adaptability and modularity. The system is created as a PaaS-based (Platform as a Service) platform (we describe the various cloud computing service models in Chapter 3). The authors, Kächele et al. (2011), suggest that developers should focus only on core application development and functional aspects. In their opinion a cloud-computing

environment should abstract from servers altogether and provide a central web interface that enables easy deployment of applications.

Also Xiao et al. (2011) have focused their research on a cloud-based system, namely a novel cloud-assisted context-aware power management framework called *CasCap*. It takes advantage of the processing, storage, and networking resources in the cloud to provide secure, low-cost, and efficient power management for mobile devices.

In a general overview of the cloud computing area, Mei et al. (2008) have found four research issues they find particularly important. The main research issues identified are:

1) **Pluggable computing entities to cloud applications.**
2) **Data access transparency.**
3) **Adaptive behaviour of cloud applications.**
4) **Automatic discovery of application quality.**

Of these areas we have chosen to focus on number 2 and 3, specifically on the seamless integration between mobile devices and cloud-based services. Finally, we also investigate, as part of our experiments, how the cloud can adapt and combine data from different sources to improve the user experience and usefulness of a mobile application.

2.3 Summary

The purpose of this chapter was to provide an in-depth introduction of relevant information in the research areas that are closely related to our own work. We started by introducing background information, with a presentation of *Distributed Computing, Mobile Computing, Pervasive Computing,* and *Internet of Things*. Next, we positioned our own work mainly in the areas of mobile and pervasive computing, and then introduced the topic of usability within smartphone applications.

We wanted to focus on usability in smartphone applications because of the lack of existing research (Coursaris & Kim 2011) and the new constraints introduced by mobile devices (Zhang & Adipat 2005).

Furthermore, in regards to usability in smartphone applications, there are three main research areas we wanted to investigate closer, namely *Wireless Personal Area Networks, Context-awareness* and *Remote Information Access.* We continued by presenting relevant research in each of these three areas. Within the topics of Context-awareness we further divided the content into two sub-topics: *Context-aware Applications* and *Smartphone Sensors*. Additionally, we provided justification for each chosen research topic. These justifications were grounded in previous research that we have identified.

The first topic we are investigating utilises Bluetooth in a heterogeneous WPAN environment, focusing on usability of a remote control application using short-range communication. In contrast to previous efforts, we created a very flexible and heterogeneous system using Bluetooth for the communication between the components. It is different than the proposed work done by Fernández et al. (2006), who focused on the server side implementation, while we used the mobile devices to persist the information, such as the user configuration. Additionally, it also differs from the universal remote control system implemented by Feldbusch et al. (2003), which was created to work over the Internet. Our solution is targeted towards a local service and was created for a meeting scenario, specifically to be used with devices in close proximity.

Within context-awareness, we created a heterogeneous system that takes advantage of Bluetooth to provide proximity information. The system consisted of a smart meeting room, where we implemented a novel feature to locate devices in close proximity and automatically established a connection. A significant difference between our work and previous research, such as Mantoro and Johnson (2003) and Ferris et al. (2010), is that we did not focus on using location as input to the system. Instead we took advantage of the short-range features of Bluetooth to provide proximity as a context input source. Additionally, in contrast to previous research efforts that created smart spaces, such as Jaimes and Miyazaki (2005) and Parviainen et al. (2006), we utilise standard laptops and smartphones, and not requiring the installation of external sensors, cameras, or microphones.

For the second part of context-awareness, we investigated the area of smartphone sensors. We implement a system for pain management, which is targeted at users with disabilities. Specifically, we wanted to use sensors on the device to provide an additional input mechanism. Moreover, we looked closer at the sensors on the Android platform and how they can be used in this scenario. In the proof-of-concept we created, we wanted to highlight challenges we experienced during the development and also present relevant examples of the implementation. Similarly, the research efforts by Bhorasker et al. (2012) and Kwapisz et al. (2011)6/19/12 8:14 PM also utilised smartphone sensors, however, in our work we focus on using the sensors as a control mechanism, whereas they used it as a passive input source for registering various activities. Another novel aspect of our work is that we provide a very practical approach to challenges and possibilities using the on-device sensors, which we have not seen in any previous research.

The last research area we are investigating is remote information access, where we implemented a system that stored context information on the cloud-based server application. A previous research effort in this area, conducted by Huebscher et al. (2006), combines context data of the same context type to provide higher quality results. We are, on the other hand, using various sources to provide context information to the system, such as calendar data and user configuration. In addition, we use a cloud-based server application to be responsible for the context management. By using this service we were able to take advantage of the flexibility and scalability, which are important and useful features offered by cloud computing. To integrate the mobile devices with the cloud-based server, we wanted to use push messaging. This also provides a novel aspect of our work, where we investigate the best-suited push messaging technology to integrate in our experiment in regards to *response times*, *stability*, and *energy consumption*.

As we have seen, whilst there has been a considerable amount of research dealing with various aspects of usability and usability in regards to smartphones, we are exploring areas that are not targeted by previous research efforts. An important aspect of all of our experiments is that we focus on the use of standard components, and we specifically focus on smartphones. We have also provided justifications for why we have selected smartphones, which include the computing power and availability of the devices.

2.4 Aim and Objectives

Having presented the research gaps that our work addresses, we are now in a position to formulate the aim of the thesis.

2.4.1 Aim

Accordingly, the overall aim for this thesis is:
To investigate aspects of usability in smartphone applications related to three areas: *Wireless Personal Area Networks*, *Context-awareness*, and *Remote Information Access*.

2.4.2 Objectives

The three thesis areas are explored in a series of four experiments. In Figure 2-6 we present the experiments, how they relate to the areas, and how they are linked together. For instance, experiment 2, *Context-aware Meeting Room,* uses a WPAN with Bluetooth to register meeting participants that are present in the meeting room. Therefore, the second experiment has an overlap with the WPAN area, even though its main focus is on context-awareness.

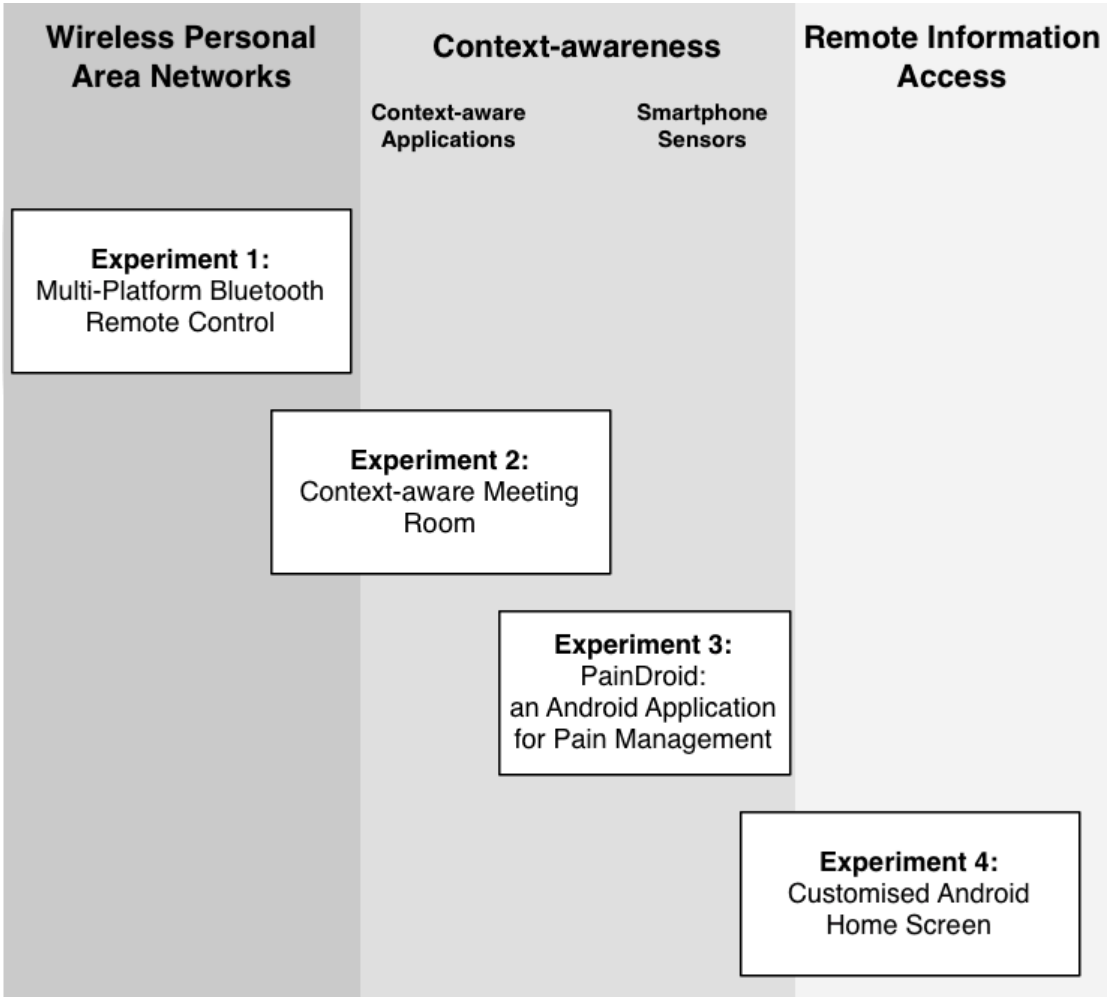


Figure 2-6, overview of experiments.

We have introduced the background for the area of mobile and pervasive computing, and then given an introduction to the research areas we have selected. Next, we continued by providing justification for the selected areas and link these with previous research efforts. Accordingly, we have created the following objectives:

Experiment 1:

Obj. 1-1. To provide an improved experience for presenters, specifically to improve the task of presenting.

Obj. 1-2. To create the possibility for each user to have their own independent solution (user specific configuration).

Obj. 1-3. To remove the need for additional hardware, such as cameras, microphones, and remote control devices.

Experiment 2:

Obj. 2-1 To handle time consuming meeting tasks automatically, such as meeting registration, information sharing, and meeting history.

Obj. 2-2 To provide context-aware and useful information without the need for additional hardware in the meeting room.

Experiment 3:

Obj. 3-1 To improve and expand the way users with dexterity problems can interact with smartphone (and tablet) applications.

Experiment 4:

Obj. 4-1 To find the overall best suited push messaging technology for integrating smartphone applications with cloud computing.

Obj. 4-2 To provide information that is relevant based on the current situation (filter content, cloud integration).

Chapter 3 – Methodology

At the end of the previous chapter we introduced the aim and objectives for this thesis. In this chapter, we continue by presenting the methodology used to accomplish the aim and objectives. The main goal is to clearly identify the link between our work and the methodology used.

We start with a general introduction of research and the relevant research perspectives, by presenting both interpretivism and positivism. This is followed by a more in-depth description of positivism, the approach used for our studies. Then, we continue by looking at design, and design science research specifically. We walk through the general steps that constitute design science research. In each of these steps, we will highlight how we conducted the work in our experiments. Towards the end, we present information on the material used, namely software components, before the conclusion.

3.1 Research

Research is described as an approach to promote knowledge enhancement or understanding (Gregg et al. 2001). Hevner and Chatterjee (2010) explain research as a process where the goal is to systematically try to find the answer to a question, the resolution of a problem, or a greater understanding of a phenomenon. Research can take many forms, but the main objective is to increase knowledge. The methods used are important to ensure the validity and rigour of the research conducted.

The research methodology explains how the process of research-related work is done. Methodology is described as "*a body of method, rules, and postulates employed by a discipline*"[11]. Since the areas of research are so diverse, the methods used in trying to improve the current knowledge in the specific fields are also different.

[11] http://www.m-w.com/

3.1.1 Research perspectives

All research is based on underlying beliefs. These beliefs are important for how the research is conducted. Orlikowski and Baroudi (1991) describe three beliefs underlying the conducts of research:

- **Ontology**
  - *"Whether social and physical worlds are objective and exist independently of humans, or subjective and exist only through human action"*
- **Epistemology**
  - *"Criteria for constructing and evaluating knowledge"*
- **Methodology**
  - *"Which research methods are appropriate for generating valid evidence"*

Each specific research perspective deals differently with these beliefs. Chen and Hirschheim (2004) present two main categories of research perspectives (paradigms) in their investigation of the paradigmatic and methodological progress made since 1991. They focused specifically on *positivism* and *interpretivism.*

Interpretivism can be briefly explained by evidence from a non-deterministic perspective, where researchers' engagement in the specific social and cultural setting is investigated and an analysis based on participants' viewpoints (Chen & Hirschheim 2004).

In contrast, positivism focuses on the formulation of a hypothesis, model or casual relationship among constructs (Chen & Hirschheim 2004). In this perspective quantitative methods are usually utilised to test the hypotheses and the focus is on the researchers' objective and value-free interpretation. Table 3-1 presents a more detailed comparison.

	Positivism	Interpretivism
Ontologically *(Existence)*	Reality exists objectively and independently from human experiences.	Emphasises the subjective meaning of the reality that is constructed and reconstructed through a human and social interaction process (Burrell and Morgan (1979), as cited in Chen and Hirchheim (2004), p. 201).
Epistemologically *(Grounds of knowledge)*	Concerned with the hypothetic-deductive testability of theories. Scientific knowledge should allow verification or falsification and seek generalizable results. As such, a causal relationship is usually presented and a tight coupling among explanation, prediction and control is expected (Orlikowski and Baroudi (1991), cited in Chen and Hirchheim (2004), p. 201).	Assume that scientific knowledge should be obtained, not through the understanding of human and social interaction by which the subjective meaning of the reality is constructed (Walsham (1995), cited in Chen and Hirchheim (2004), p. 201).
Methodologically	To test hypothetic-deductive theory, research should take a value-free position and employ objective measurement to collect research evidence. A quantitative method such as the survey is a typical positivist instrument.	To understand the meaning embedded in human and social interaction, researchers need to engage in the social setting investigated and learn how the interaction takes place from the participants' perspective. Field studies that engage researchers in the real social setting are appropriate for generating interpretive knowledge (Orlikowski and Baroudi (1991), cited in Chen and Hirchheim (2004), p. 201).

Table 3-1, Positivism and Interpretivism comparison, based on research by Chen and Hirchheim (2004).

3.1.1.1 Positivism

In our experiments we have adopted the positivistic research perspective. It is based on the assumption that research can be objective, and that the researchers are independent and the results are valid, reliable, and replicable (Pather & Remenyi 2004). The positivist research should ultimately result in an increased understanding through a process of confirmation/falsification of propositions (Gregg et al. 2001). Although not strictly necessary, the research in this perspective is usually quantitative.

According to studies done by Orlikowski and Barioudi (1991) and Chen and Hirschheim (2004), the positivistic perspective has dominated the research areas in Information Systems (IS). In their opinion the IS field would benefit from a broader range of research paradigms, but both the publication process and academic promotion system does not promote this. However, the critique is not based on trying to replace the positivistic perspective, which in many situations is the best choice. According to the authors, it is important to note that it is not the only choice and there are other equally valid research paradigms, such as interpretivism that we previously presented.

The reason we have chosen to use the positivistic research perspective is that we wanted to verify the validity of our objectives, which were implemented under various computing systems. To gather relevant results from the user experiments we conducted user evaluations. Additionally, we also included laboratory experiment in one case. Positivism is the best-suited perspective to handle these tasks, because it creates a solid foundation providing quantitative results and empirical measurements. By including user-provided and laboratory-test results, we can make statements based on the results and not rely on guesses, which can often lead to inaccuracies.

In the experiments conducted as part of this thesis, we have used the Design Science Research (DSR) methodology, using the positivistic perspective. The next sections present an overview of what design science research is, and give a description of how the experiments were evaluated in the context of the chosen research methodology. We will start by looking at design, which is an important factor in design science research.

3.2 Design

The general description of design is that it embodies the instructions for making *things*. However, it is not the *thing* itself (Hevner & Chatterjee 2010). One relevant comparison is how we design models of computer systems before and during the build process. This helps us get an overview of the system and an initial plan. The plan is not meant to be complete and it does not include all components of the actual running system. The designed model is not a one-to-one mapping, but is valuable due to the abstractions it makes, thus, helping both developers understand how the system is built, as well as being a tool used to communicate with domain experts that may not be technical.

Hevner and Chatterjee (2010) state that the challenge of building large-scale software systems is very different from building large physical systems. IS design does not involve the same process as for example the construction of a building. A building requires detailed planning, to the smallest detail, before the construction process has started. All issues not figured out in the planning phase can have severe economic consequences and cause, potentially, large delays. In software this is very different. The process of actually assembling software is so *cheap* in terms of time and cost that we can do it in minutes, depending on the project size, and throw away the result if it's not what we wanted. The major cost is in the design and construction process of software development. This has some very interesting implications when looking at the design of software. Big up-front designs are avoided; instead a small planning phase is usually done before the implementation. Evans (2003) explains that instead of making elaborate up-front design decisions, a continuous series of small discrete design changes are preferred. It is a hands-on approach, where we learn more about the problem domain and issues as we go along.

Developing software-based systems is a challenging task, and many times the final goal of the system is very hard to predict. Starting with one idea during the development stage, it can change for several reasons. New features that were not part of the original plan appear when more experience in the problem domain is gained. Also, the runtime constraints on the system can be altered. New hardware or even new platforms can be introduced, adding pressure to create an adaptable design. This is particularly a

challenge in highly heterogeneous environments, where multiple platforms have to be supported. The main objective is to handle the trade-offs when designing systems, making a conscious choice among other alternatives that places constraints to utility and resources (Hevner & Chatterjee 2010).

To summarise, design in IS deals with building software artefacts, which solves human problems. Artefacts are categorised as several components, including *constructs, models* - abstractions*, methods* - algorithms and practices*,* and *instantiations* - implemented and prototype systems (Hevner & Chatterjee 2010). It is also important to note that the accumulation of experiences and knowledge acquired during the development of the system represents a viable research goal in and of itself (Gregg et al. 2001).

The activities in a general design cycle are shown in Figure 3-1. It begins with the problem definition and moves on with suggestions on how to solve it. Development and evaluation proceeds, with the possibility of moving back to the problem definition stage. Finally, a conclusion is drawn where the results are the final output.

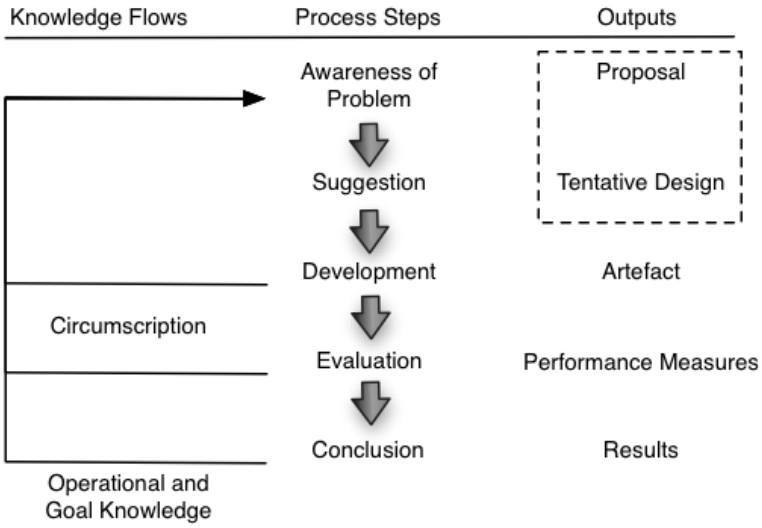


Figure 3-1, Design Science Research (Hevner & Chatterjee 2010).

Designing and developing artefacts is not limited to design research, it is a big part of professional design as well. However, there is a clear distinction between the two: design research provides an identification of a contribution to the body of knowledge and clearly communicates the results, whereas professional design is the application of

existing knowledge to organisational problems (Hevner & Chatterjee 2010). Even when the research does not provide the expected results, the knowledge gained can be very useful. The following section will explore *Design Science Research* in more detail.

3.3 Design Science Research

Design science research is based on the concept of a designer trying to answer questions related to human problems. The creation of innovative artefacts is an important part of the process, whereas the designed artefacts are useful and fundamental in understanding the problem(s) (Hevner & Chatterjee 2010).

Hevner and Chatterjee (2010) define three design cycles, presented in Figure 3-2, which provide a general overview of the design science research process. The *relevance cycle* integrates the environment of the research project with the research activities. Connecting the design research process with the knowledge base is a task of the *rigour cycle*. Finally, the *design cycle* consist of the iterations during the development of the artefact and the evaluation.

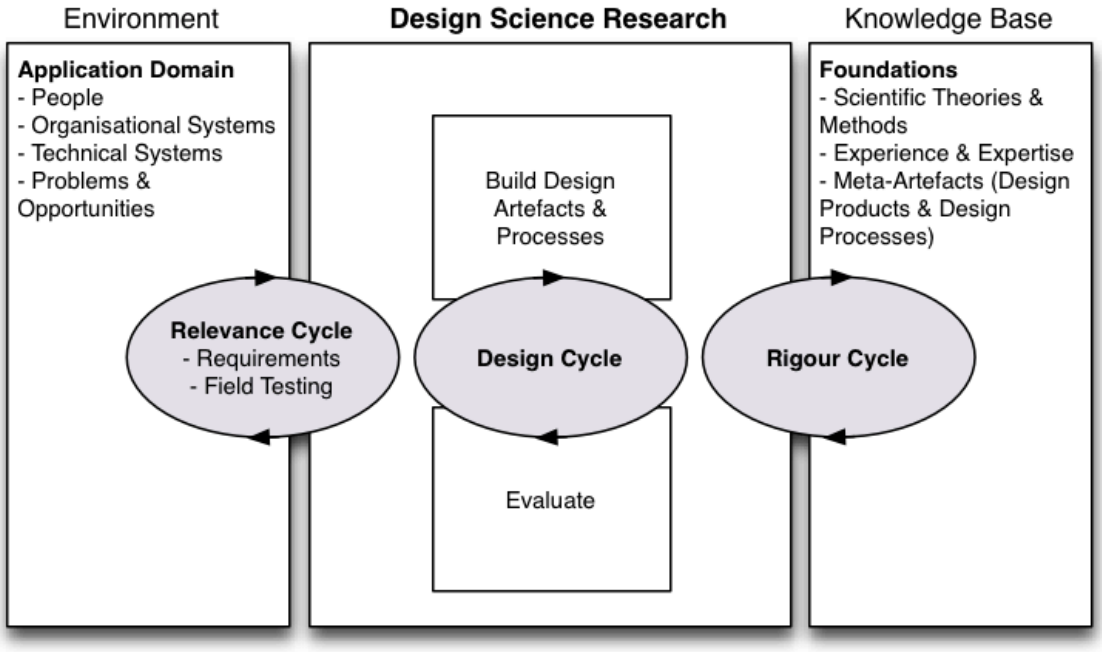


Figure 3-2, the cycle of design science research (Hevner & Chatterjee 2010).

In the experiments we conducted, we followed the ideas of the design science research cycles. Each one of the three cycles in this process, *relevance, design,* and *rigour,* are often done in iterations. Our process started with the *rigour cycle,* where we investigated related work and positioned our own efforts within the body of knowledge. Next, we continued with the *relevance cycle*, where we created the specific requirements for the system and how it should integrate with the environment and application domain. The next step is to initiate the development process, with the *design cycle*, where the focus is on building the artefact. Finally, the user evaluation is conducted.

Closely related to these cycles, are the five general steps proposed by Hevner and Chatterjee (2010), which are shown in Figure 3-3. These are essentially providing the same general ideas for the design science research process as in Figure 3-2, but they are more detailed and present each step in sequence. We continue with a closer look at each one of these steps.

The first step is to identify a problem and the motivation for solving this problem, referred to as *Construct a Conceptual Framework*. The problem definition is used as requirements for the development of an artefact. According to Hevner and Chatterjee (2010), this step accomplishes two factors: 1) It motivates the researcher and the audience to pursue the solution and accept the results, and 2) helps communicate the reasoning associated with the researcher's understanding of the problem.

Knowledge of the state of the art in the specific field of research is important to be able to identify the problem. Also, it provides the motivation behind trying to solve it. Reading similar and relevant research published in conference proceedings and journals are essential, which we presented in the Chapter 2 (*Literature Review*). Insight into the field of study is important to understand what previous research has produced. Not only does this increase ones' knowledge in a specific area, it helps to justify the research question. It is important to be as confident as possible that the research contains novel aspects not yet investigated to avoid reinventing the wheel. Often times, ideas on what current research is missing are discovered when looking at other researchers work. This might uncover lacking aspects of previous research, or provide the foundation for a continuation of work already completed.

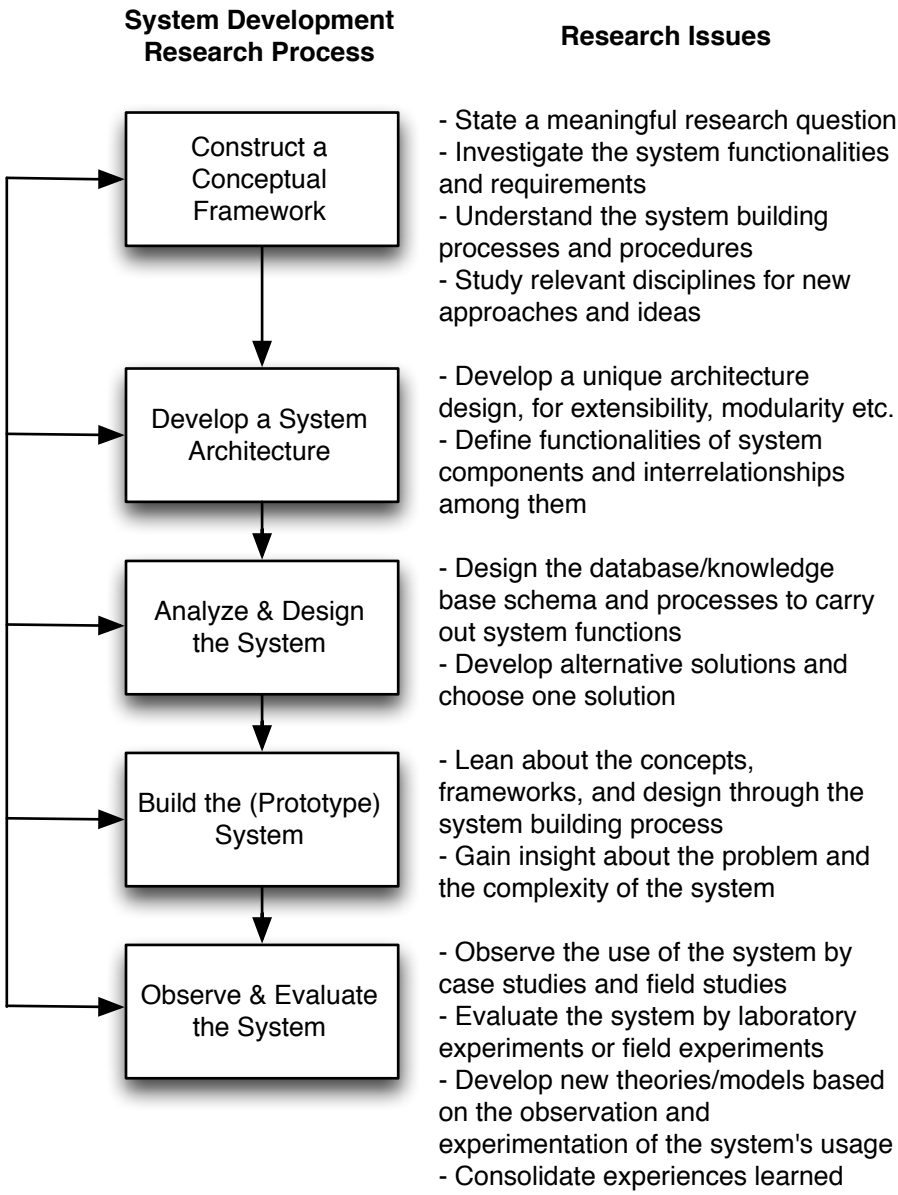


Figure 3-3, System Development Research Process (Hevner & Chatterjee 2010).

For each experiment, we studied related work to the topic we were investigating. In several cases, the problem identification was a result of a previous experiment. Such as, when we were doing an experiment, we noticed a crucial part missing in the literature and we wanted to expand by conducting a new experiment that focused on these issues.

Figure 3-4 presents an overview of how our experiments are integrated with the design science research process. It is linked to the five stages in the system development framework. In each step we have included a short description of how the various

experiments were created and evaluated. The text included in the figure only show the most important parts for each experiment, because of limited space. In later chapters we will look more closely at the specific experiments and present how they related to the process steps.

The next sections continue with the design science process steps identified by Hevner and Chatterjee (2010). We have already presented *problem identification and motivation*, the next steps are: *objectives for a solution, design and development, demonstration, evaluation,* and *communication*. In each section we give general background information on what the steps are, and then link them to our own experiments.

3.3.1 Objectives for a Solution

The objectives address the tasks that are needed to solve the problem, i.e. the goal of the research. Objectives can be *quantitative*, namely how a solution would be better than the current one, or *qualitative*, that is a description of how a new artefact is expected to support solutions to problems not yet addressed (Hevner & Chatterjee 2010).

Quantitative research utilises numerical analysis and tries to illustrate the relationship among factors in the phenomenon studied (Chen & Hirschheim 2004). The alternative is qualitative research, which deals with the description and understanding of the situation behind the factors. Whereas quantitative research often deals with surveys or experiments, qualitative research can use methods such as field studies.

Quantitative research is often mentioned in combination with empirical studies. Empirical methods are commonly used in the evaluation process (Hevner & Chatterjee 2010). Empirical studies rely on observation and data (Alavi et al. (1989), cited in Chen and Hirchheim (2004), p. 205). The nature of quantitative research matches well with the requirements for evidence and data that an empirical study requires. Non-empirical studies, on the other hand, emphasise ideas and concepts. Non-empirical and qualitative research is usually combined. The experiments conducted as part of our studies used, in most cases, a quantitative approach, gathering data from user evaluations and laboratory experiments. We also included a few qualitative research aspects, such as open-ended

questions where we wanted the users to express their own opinions. This, since, as stated by Fitzgerald and Howcroft (1998), qualitative techniques can complement quantitative ones.

	Wireless Personal Area Networks	Context-awareness		Remote Information Access
Process	**Exp. 1**	**Exp. 2**	**Exp. 3**	**Exp. 4**
1 Construct a Conceptual Framework *(Goal of research)*	Create a heterogeneous system using WPAN technology.	Construct a system supporting proximity and system input, both contributing to the context-awareness.	Utilise smartphone on-device sensors to improve the usability and provide a context input source.	Create a context-aware system using multiple context sources and integrate with a cloud computing environment.
2 Develop a System Architecture *(Components of the system)*	Bluetooth, short-range communication. Java ME, Windows Mobile.	Bluetooth, Internet resources. Java ME, Windows Mobile, Android, iPhone.	Smartphone on-device sensors. Accelerometer, magnetometer. Android.	Smartphone on-device sensors, Internet resources. Android.
3 Analyze & Design the System	2 Mobile clients 1 Local server Focus on integration and availability.	4 Mobile clients 1 Local server 1 Cloud-based server Focus on context-awareness and proximity.	1 Mobile client Focus on supporting a variety of device and input types.	1 Mobile client 1 Cloud-based server Focus on multiple context sources and context management.
4 Build the (Prototype) System *(Gain insight into the problem)*	Bluetooth integration on multi-platform systems.	Supporting a wide variety of platforms. Introducing context-awareness with proximity (using Bluetooth).	Increasing usability and introducing context-awareness through the use of on-device sensors.	Integrating multiple context-sources. Using a cloud-based application as context management.
5 Observe & Evaluate the System	System demonstration and user evaluation.	System demonstration and user evaluation.	System demonstration and user evaluation.	System demonstration and user evaluation.

Figure 3-4, overview of research process and experiments.

When the objectives are clearly stated, the next step in the process is the *Design and Development* phase.

3.3.2 Design and Development

The research continues with the design and development of the artefact that at this point has a clear purpose (problem identification) and defined requirements (objectives). The artefacts can take different forms depending on the research conducted, namely *constructs*, *models*, *methods*, or *instantiations* (Hevner & Chatterjee 2010).

The activities in this phase include an understanding of the desired functionality and architecture. In software development, the functionality of the artefact is often documented in the form of *use cases*. Martin Fowler (2003) explains, "*Use cases are a technique for capturing the functional requirements of a system. Use cases work by describing the typical interactions between the users of a system and the system itself, providing a narrative of how a system is used*". In the experiment chapters, we provide an overview of the most important requirements as use cases.

Usually, an overview of the designed system provides important information. Not just for planning and development, but also as a tool of communication. In a business setting, an abstract model of the system provides the customer with an overview of how the system works. For research projects, models can inform researchers about the system and focus on how the most vital parts of the system cooperate.

In our development process we used steps from *scrum*, which is an agile framework primarily for software development. The idea behind scrum is that the project team creates and uses a product backlog. The tasks in the backlog are prioritised and then implemented in sprints, which are iterations that usually last between two and four weeks[12]. At the end of the sprint, the work should be potentially shippable. These iterations repeat until the product backlog has been completed, the budget is depleted, or a deadline arrives.

[12] http://www.scrumalliance.org/learn_about_scrum

In our projects we created a backlog for each project, which contained the important elements we had to implement to meet the requirements and a few lower-prioritised elements, which were *nice-to-have*, but not essential. Additionally, we conducted the work in iterations (sprints), each consisting of a short period of time, usually between two and four weeks. During the development we did the work over several iterations, each step implementing certain parts of the desired functionality. Note that the requirements can change after development has started. The initial idea is built upon incomplete knowledge of the research area, and only after the actual implementation has started will small details appear that were not considered in the planning phase. Scrum is able to handle frequent changes, because it uses short iterations and prioritises tasks for each sprint. This was useful in our work because we were able to handle changes in the requirements quickly. We were often creating novel systems, and therefore we expected frequent changes in design and requirements for the system as the process moves on.

3.3.3 Demonstration

The researchers will continue by demonstrating the use of the artefact to solve one or more instances of the problem identified (Hevner & Chatterjee 2010). The demonstration can involve several different approaches, for example with *experiments*, *simulations*, or *case studies*.

In our experiments, we proceeded with a demonstration phase. All experiments included a test from the developers, testing the features to make sure the designed artefact works and that we were addressing the research problem identified. The next step was a prototype demonstration. This included a user evaluation, but the number of users was small, usually only one or two relevant users. Relevant users in this context were people who knew the project and the problem domain, but were not part of the development effort. This allowed us to get feedback on the functionality and find bugs that we had not yet encountered. The prototype test also gave us an indication if the questionnaire was providing the information we wanted. Kitchenham and Pfleeger (2002a) state that pilot studies, demonstrations in this context, are intended to identify any problems with the questionnaire. In a few occasions we decided to proceed with new iterations, going back to the design and implementation phase to develop new features and fixing bugs.

Also, we altered the questionnaire a few times after the initial demonstration. These iterations were very useful and helped us focus on the research questions we wanted to provide answers to.

When we were satisfied with the demonstration and the prototype tests, we moved on with the user evaluation.

3.3.4 Evaluation

The evaluation is a crucial part of the research process, observing and measuring how well the created artefact supports a solution to the specific problem (Hevner & Chatterjee 2010). At the start of the project, research questions should be created, whereas the evaluation is completed to answer these. The end result is a product of comparing the objectives of the designed solution to the actual results from the demonstration.

There are several different forms of evaluation, as classified by Chen and Hichheim (2004), based on earlier research conducted by Orlikowski and Baroudi (1991):

1. **Survey** – Gather data from questionnaires.
2. **Case study** – Studies that are involved with a single site or a few sites over a certain period of time.
3. **Laboratory experiment** – Studies that take place within a designed, controlled environment and usually involve special treatments of different groups to contrast the precise relationship among variables.
4. **Field experiment** – Special treatments are used to study two or more controlled groups. Conducted in a real-world setting.
5. **Action research** – Often embodied in a case study, researchers are an integral part of the phenomenon under study. The researchers' input will often influence the outcome of the phenomenon.
6. **Others** – Includes work that is practitioner oriented, non-empirical, or descriptive.

Furthermore, Hevner and Chatterjee (2010) explain the evaluation process in design science as: observing and measuring how well the artefact supports a solution to the

problem. We have identified the problems we want to address with the objectives. Moreover, we evaluated our objectives using *questionnaires* and a *laboratory experiment.* In the next section, we go into more detail on the evaluation, starting with questionnaires.

3.3.4.1 Questionnaires

"*A questionnaire is a set of questions for obtaining information from individuals*"[13]. According to Kitchenham and Pfleeger (2002a), one important issue when creating the questionnaires is the consideration of the impact of our own bias. They also mention how it is possible to avoid bias in the questionnaires:

- **Develop neutral questions.**
- **Ask enough questions to adequately cover the topic.**
- **Pay attention to the order of questions.**
- **Provide exhaustive, unbiased, and mutually exclusive response categories.**
- **Write clear, unbiased instructions.**

In our evaluations, we tried to follow these steps. However, it is very difficult to clearly verify these suggestions. Furthermore, according to Johns (2010), statements, which we used in the close-ended questions, can potentially contain persuasive assertions. However, before the actual user tests, we conducted pilot tests and also had other people related to the research work we were conducting read through the questionnaire. While we acknowledge the possibility for inadvertently creating biased questions and statements, we feel that we took the necessary precautions to avoid it.

When we did the evaluation, we divided the users into small groups, each consisting of about five people or less. There were several reasons for this; we wanted to do an initial presentation of the created system to each group, answering any questions that were asked about the system. The presentation consisted of an overview of the designed system, and what the system was trying to solve and the problem we had identified. The second reason was that we did not have enough mobile devices for every user to do the experiment simultaneously.

[13] www.m-w.com

We included both close-ended statements, with a fixed set of responses, and open-ended questions, where the users are free to write their own answer. For the close-ended statements we used a *Likert scale*. According to Kaptein et al. (2010), Likert-type scales are used to obtain quantified data regarding participants' attitudes, behaviours, and judgements. We used the Likert scale to measure positive and negative statements. A Likert item consists of two parts, namely the statement and response scale. The key point is that Likert items are intended to capture the extent of agreement or disagreement with an idea. According to Johns (2010), the main advantages of the Likert format are *simplicity* and *versatility*. Additionally, in our questionnaires, we randomly placed negative and positive statements, with a roughly equal number of statements in each type. This was done to minimise the bias during the user evaluations. Moreover, the statements in the questionnaire were grouped together into topics, which is suggested by (Kitchenham & Pfleeger 2002b).

Questionnaires were used for the evaluation process in all four experiments. Additionally, we also conducted a *laboratory experiment* as part of experiment 4 (*Customised Android Home Screen*).

3.3.4.2 Laboratory Experiment

Laboratory experiments are studies that take place within a designed and controlled environment. It usually involves special treatments of different groups to contrast the precise relationship among variables (Galliers (1991), cited in Chen and Hirschheim, (2004)). Laboratory experiments can provide precise measurement and control of variables, but at the expense of the naturalness of situation. This is because the real-world intensity and variation may not be achievable (Fitzgerald & Howcroft 1998).

In our research we created a laboratory experiment when doing a performance test of push messaging technologies for the Android platform. The evaluation was done with test applications set up in a controlled environment. Each device was running a specific application created to run a performance test, continuously measuring the performance of different push messaging technologies. We compared the response times, stability, and energy consumption of the different technologies. An important part of the laboratory experiment was to create an identical environment for each test.

After the evaluations are conducted and the results are gathered, the next step in the design science research process is to analyse the collected results.

3.3.4.3 Analysis of Results

We used SPSS to analyse the results from the user evaluations. The results were exported from the web-based questionnaire or manually added when we used paper-based questionnaires, and imported into SPSS. The mean and standard deviation were collected from the responses.

In the cases of simple calculations we used MS Excel. This included calculations of average, maximum, minimum, and standard deviation. In the final results, we also included graphs created with MS Excel from several of the experiments. The use of MS Excel was mainly limited to the laboratory experiment, where we had a considerable amount of data that we needed to generate graphs for.

At the end of this evaluation phase, researchers must decide on whether to do another iteration, starting again with design and development, or continue on with the next step, which is *communication*. Communication consists of sharing the results from the research conducted with the research community. According to Hevner and Chatterjee (2010), the problems and their importance, the artefact, its utility and novelty, the rigour of its design, and its effectiveness should be communicated to researchers and other relevant audiences. Our research contributions, with the list of publications, are presented in the introductory pages.

We have presented an introduction to the methodology we have chosen and linked this to our research. Next, we move on to the material used in the experiments.

3.4 Material

In this section we will present the software and hardware platforms used as part of our research experiments. We describe the various platforms selected and provide the reason why they were chosen.

3.4.1 Software Platforms/Systems

This section presents an overview of the software platforms/systems we have used in our research. We have divided the software platforms/systems into three, namely *WPAN Technology, Mobile Operating Systems/Platforms,* and *Cloud Computing Platforms.*

3.4.1.1 WPAN Technology

A vital part of WPANs is the short-range communication, both between devices and between device and other components such as a laptop computer. The main technology we used to provide this functionality is Bluetooth. The Bluetooth name is taken from the name of a Danish king (Kamal 2008) and emerged from an effort began by Ericsson in 1994 (Hansmann et al. 2000). Bluetooth IEEE 802.15.1 is a WPAN standard and operates in the 2.45 GHz unlicensed radio band. It includes a feature called frequency hopping, where the Bluetooth technology is able to minimise the effects of interference from other signals (Kurose & Ross 2002). Low powered Bluetooth devices commonly have a range of about 10 meters. Additionally, the technology provides a basic data rate of 1 Mbps (Hansmann et al. 2000). Bluetooth's current use includes interconnecting a wide range of devices including headsets, printers, phones, computers, and music devices (West 2011).

There are several technologies available for WPANs, such as Bluetooth, infrared, and ZigBee. We have chosen to use Bluetooth in our experiments. According to Morak et al. (2012), Bluetooth is the most popular personal area network technology available in mobile phones. This technology was selected because it does not need the line-of-sight that is required by infrared, and it has a range that was suitable for our research experiments. In addition, most smartphones and laptop computers are today equipped with Bluetooth, providing good availability. The WPANs make it possible to connect a collection of devices using Bluetooth as the underlying technology (Loke 2006). The main characteristics of Bluetooth are: *low power, low cost,* and *ability to support high-speed ad hoc networking.* There is a close synergy between mobility and wireless technology (Perkins 1998). Mobile devices are dependent on wireless communication to

enable interaction with external resources. The possibility for wireless data communication enables mobility and network access at the same time.

There have been some concerns about security related to Bluetooth in several research efforts. Although security is outside the scope of this thesis, we want to provide some background information on this topic. According to Naqvi and Riguidel (2004), work remains to address the lack of security with Bluetooth. Hager and Midkiff (2003) provide an evaluation of the Bluetooth security vulnerabilities. Their results show that the Bluetooth technology has vulnerabilities relating to improper validation, exposure and the potential for improper randomness. A specific problem is the use of short PIN numbers. Another issue is highlighted by Shaked and Wool (2005), who identify the possibility for an attacker to eavesdrop on the authentication process, and then be able to use a brute force algorithm to find the PIN used. While Bluetooth has certain security issues, the threats can be limited because of the short-range communication. The consequences are smaller than compared to the long-range wireless or wired technologies. Also, if the users create connections with a long PIN number, the chance of a security breach decreases.

3.4.1.2 Mobile Operating Systems/Platforms

The most important part of our work is the use of smartphones. The devices used in our experiments had a variety of mobile operating systems. These systems enable a programmer to develop applications without considering low-level issues such as drivers (Kamal 2008).

There are several major platforms available on the market. Table 3-2 presents the top six operating system competitors in the smartphone market (Gartner 2012). In this heterogeneous environment, we selected a total of four platforms for the work conducted as part of this thesis. The platforms are: *Java ME*, *Windows Mobile*, *Android*, and *iOS*. Covering five of the top six operating systems, the only one not represented is the newly released Bada created by Samsung, which supports C/C++. This was not included due to the fact that it is a new platform and it currently only has a minor market share when compared to the bigger and more mature platforms. Java ME is supported by both Symbian and Blackberry, which is produced by Research in Motion.

Operating System	4Q11 market share (%)	Diff. from 4Q10 (%)
Android	50.9	20.4
iOS	23.8	8.0
Symbian	11.7	-20.6
Research in Motion	8.8	-5.8
Bada	2.1	0.1
Microsoft	1.9	-1.5

Table 3-2, smartphone operating systems market share.

We included Java ME and Windows Mobile to support the heterogeneous mobile device environment. This is particularly important because there are still a considerable amount of older and low-end devices on the market. Also, iOS was included in several experiments, where we wanted to provide a truly multi-platform system that supported a wide range of devices.

Our main focus, however, was on the Android platform. There are three main reasons behind this decision. Firstly, we acknowledge the major impact Android is having, both in terms of the impressive market share and momentum of the platform. Not only is the user base rapidly growing, but also new features of the platform are quickly being developed as new versions are released. Secondly, even though there are a wide range of Android devices on the market, the compatibility issues are usually handled by the platform. Although there are corner cases where we have experienced differences in behaviour between device models, our overall impression is that Android has solid support for dealing with this fragmentation. This is especially true for later versions of the Android platform, where support for multiple screen resolutions have been improved. Thirdly, and perhaps most importantly, the fact that Android is an open platform, based on open source software, is ideal for research. This openness is especially important when prototyping applications that previously have not been attempted. An open platform gives insight into how the platform itself works, and a much greater freedom when trying to adapt the behaviour. Also, a strong community of developers backing the platform, willing to share knowledge and best practices, is a very important benefit when working with novel implementations.

In the next sections we go into more detail on the different platforms used, starting with *Android*.

Android

The Android (*beta*) was released by in 2007, under the Open Handset Alliance, a group of hardware and software companies, all working towards a more open mobile phone environment (DiMarzio 2008). The Android operating system is an open source software stack for mobile devices that is led by Google[14]. It is based on the Linux kernel, and supports applications written in Java syntax. This is one of the main advantages to the Android platform, because the programming language and tools will be familiar to Java programmers. The main tool, which is also provided by Google, is a plugin created for the Eclipse IDE (Integrated Development Environment) called ADT (Android Development Tools)[15]. Other available IDEs, for instance NetBeans or IntelliJ IDEA, also have support for Android development. Additionally, several third part libraries are also added to the platforms, such as the Apache Commons projects (Murphy 2011).

In contrast to standard Java, the source code is not tied to the JVM (Java Virtual Machine), but runs on the Dalvik Virtual Machine. The Dalvik VM executes dex-files (Dalvik Executables), which consists of classes compiled by a Java language compiler that have been transformed into dex-format by the *dx-tool*[16].

iOS

Unlike the openness surrounding the Android platform, the iPhone, with its iOS platform is owned and controlled by Apple (Hall & Anderson 2009). It is based on Mac OS, which is found on Apple desktop and laptop computers.

Apple offers a development tool called Xcode to implement applications for iOS. With Xcode it is possible to create applications for Mac, iPhone, and iPad[17]. The language

[14] http://source.android.com/
[15] http://developer.android.com/sdk/eclipse-adt.html
[16] http://developer.android.com/guide/basics/what-is-android.html
[17] https://developer.apple.com/technologies/tools/

used when creating iOS apps is Objective-C[18], which is defined as a small set of extensions to the standard ANSI C language. It is specifically designed to provide object-oriented capabilities to C[19].

To run custom written applications on the iPhone device and add them to the App Store, one needs to register for the developer program and pay an annual fee of $99 (Kochan 2011).

Java ME

Java ME (Micro Edition) was made in an attempt to create a common programming language for mobile phones. It is a platform on which very small and flexible Java application environments can be defined. The platform uses a concept of profiles, which specify the language subsets for different groups of devices (Hansmann et al. 2000).

The MIDP (Mobile Information Device Profile) combined with CLDC (Connected Limited Device Configuration) provides a standard Java runtime for mobile information devices, such as mobile phones. MIDP and CLDC supports the core functionality required by mobile applications[20].

It is also worth mentioning that we experienced compatibility issues that are especially problematic with the Java ME platform. According to Huebscher et al. (2006) and Greenhalgh et al. (2007) there are issues with platform interoperability, and Java ME in particular. Their research discovered that Java ME applications and especially those using the optional APIs, suffered from stability and compatibility issues.

Windows Mobile

Windows mobile is developed by Microsoft, and is based on the Windows CE 5.x series. It is used for a variety of devices, including PDAs and smartphones[21].

[18] https://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AppDesignBasics/AppDesignBasics.html
[19] https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html
[20] http://www.oracle.com/technetwork/java/overview-140208.html
[21] http://msdn.microsoft.com/en-us/library/ms920098.aspx

Windows Mobile applications can be written in managed (Visual C# / Visual Basic .Net) and native (Visual C++) code. The .Net Compact framework, which is supported by Windows Mobile, runs with access to many operating system features on the device. These features enable sophisticated and feature rich applications. The .Net compact framework and .Net framework shares several abilities such as a Common Language Runtime (CLR), which is similar to the Java Virtual Machine (JVM) on the Java platform.

3.4.1.3 Cloud Computing Platforms

To offer remote information access for the implemented mobile clients, we used cloud computing as the host of the back-end services. Cloud computing builds on other computing paradigms, such as distributed computing. As such, distributed computing has influenced cloud computing with technology, including the ability to cope with the distribution of server nodes. Cloud-based systems usually consist of many physical machines, which to the user appear as one single resource. The scalability is transparent and therefore the users can transfer data without the need to know where it is stored (Mei et al. 2008).

Additionally, service-oriented computing has also proven to have important features utilised by cloud computing. Service-oriented computing, which has a strong link to SOA, is useful in modelling functionality and providing flexible services (Mei et al. 2008) . A service is a function with well-defined, machine-readable interfaces and uses standardised message formats to communicate with external entities. Cloud computing and service-oriented computing share the same interaction sequence, with the commonly used input/output feature of cloud computing resembling that of service computing, i.e. using request/response interactions.

Cloud computing has also benefitted from work done in the pervasive computing area. Pervasive computing, as previously described, provides self-adaptive capacity to software with respect to environmental contexts (Mei et al. 2008). Similar to pervasive computing, cloud computing entities should be able to plug into a cloud dynamically. One example presented by Mei et al. (2008), is when a large cluster of computer

workstations and business services are attached to a cloud. This situation can change the availability of computing entities radically.

The last paradigm we want to introduce is *grid computing*. Grid computing is a type of parallel and distributed system. It enables sharing, selection, and aggregation of geographically distributed resources dynamically at runtime (Buyya et al. 2009). Grid computing applies the resources of numerous computers in a network to work on a single problem at the same time (Elsenpeter et al. 2009).

There are a few key differences between grid and cloud computing that are important to keep in mind. Dillon et al. (2010) present these differences and we have compiled these together in Table 3-3.

	Grid Computing	**Cloud Computing**
Resources	Emphasises resource sharing to form a virtual organisation. A large project is divided among multiple computers (Elsenpeter et al. 2009).	Cloud is often owned by a single organisation. Allows multiple smaller applications to run at the same time (Elsenpeter et al. 2009).
Performance and Capacity	Aims to provide the maximum computing capacity for a huge task through resource sharing.	Aims to suffice as many small-to-medium tasks as possible. Provides on-demand computing, scale up and down, in and out, and at the same time optimising the overall computing capacity.
Requirements	Trades re-usability for scientific high performance computing.	Directly pulled by immediate user needs driven by various business requirements.

Table 3-3, comparison of Grid and Cloud computing.

There are several types of cloud computing platforms and they offer different development and runtime environments. First, we present the service models in cloud computing, before looking at the deployment models, and then finally an introduction to the *Google App Engine*, which we used in our experiments.

There are three main service models in cloud computing (NIST 2011):

- **Software as a Service (SaaS):** The service consumer is able to use the provider application running on a cloud-based infrastructure. The consumer does not manage or control the underlying infrastructure.
- **Platform as a Service (PaaS):** The service consumer is able to deploy onto the cloud infrastructure, but the consumer does not manage or control the underlying infrastructure. The control is limited to deployed applications and possibly configuration settings or the application-hosting environment.
- **Infrastructure as a Service (IaaS):** The capabilities provided to the service consumer are to provision processing, storage, networks, and other fundamental computing resources. It opens up the possibility to deploy and run arbitrary software, which can include operating systems and application. The consumer does not manage or control the underlying infrastructure, but has control over operating systems, storage and deployed applications.

According to NIST (2011) there are three main deployment models for cloud computing:

- **Private cloud:** The infrastructure is provisioned for exclusive use by a single organisation.
- **Community cloud:** A specific community of consumers provisions the infrastructure for use.
- **Public cloud:** The general public provisions the infrastructure for open use. The service exists on the premise of the cloud provider.
- **Hybrid cloud:** A composition of two or more cloud infrastructures (private, community or public). The infrastructures remain unique entities, but are bound together by standardised or proprietary technology that enables data and application portability.

There is also a categorisation of how clouds are commonly utilised. Elsenpeter et al. (2009) have create these three general categories:

- **Compute clouds** allow access to highly scalable, inexpensive, on-demand computing resources that run the code that they are given. Examples include Amazon EC2 and Google App Engine.
- **Cloud storage** allows storage of data on a vendor's equipment. This is an ideal solution if one wants to maintain files off-site, e.g. Amazon S3.

- **Cloud applications** represent variations of Software as a Service (SaaS) and include systems such as web applications that are delivered to the users via a browser. These systems offload hosting and IT management to the cloud, e.g. Google Apps.

The cloud service we used in the cloud computing experiments was the *Google App Engine.* The Google App Engine is a Paas service, which lets you run web applications on Google's infrastructure. It has support for applications running in one of three runtime environments: *Go*, *Java*, or *Python*. It provides a secure environment that has limited access to the underlying operating system. Additionally, the Google App Engine supports a NoSQL data storage service that features a query engine and transactions. Other features include authentication and the ability to send email using Google accounts, which are provided with APIs. Automatic scaling and load balancing is also supported[22].

There are a few limitations worth noting. Applications cannot write to the file system and the applications must return response data within 60 seconds. Also, a request handler cannot spawn a sub-process or execute code after the response has been sent.

We chose to use the Google App Engine in our experiments because it provides a close integration with other relevant Google services, such as authentication, calendar, and e-mail, which we also utilised. Additionally, we found the PaaS service model to be well suited for our research because it allowed us to focus on our software system and not the operational aspects of the system, which were automatically handled by the Google App Engine.

3.4.1.4 Hardware Devices

We included a total of four different mobile platforms in the experiments. Each platform was tested with several device models. In general, these devices were chosen because they are all mobile phones. In some of our experiments we also needed

[22] http://code.google.com/appengine/docs/whatisgoogleappengine.html

hardware support for technologies, such as Bluetooth and WLAN. This also impacted the choice of devices.

In two of our experiments, Chapter 4 and 5, we focused on creating heterogeneous systems. Therefore, we included several device types with various operating systems. The selected devices included the most popular smartphone operating systems, as shown by (Gartner 2012). The experiments creating a heterogeneous system also included lower-end devices to create a realistic scenario where the users installed the application on their own devices.

In other experiments, where we wanted to focus on a particular platform, we only included a few device types with the same OS installed. In these situations we focused on the Android platform. We have previously described the reason for selecting Android in these cases, most importantly because of the openness of the platform. The devices selected for these experiments were usually high-end smartphones, because of the features offered by these devices, e.g. support for multiple sensors and high-speed wireless networks. All devices used in our experiments are listed in Appendix A.

3.5 Summary

This chapter presented research methodology, where we started by introducing a description of research and relevant research perspectives. We continued by providing background information and justification for the selected methodology, which is design science research. Design science research was linked to our experiments, with the five design science process steps identified by Hevner and Chatterjee (2010). We also explained the methods used for gathering data in the evaluation step, namely questionnaires and a laboratory experiment. Towards the end we presented the material used as part of our experiments.

In the next chapters we will continue by presenting the experiments that were conducted as part of this thesis.

Chapter 4 – Wireless Personal Area Networks

We begin the presentation of our experiments by going into detail on *Wireless Personal Area Networks*. Within this topic we are investigating the usability of a flexible remote control application, created to take advantage of the capabilities of Bluetooth in a heterogeneous environment. The experiment conducted within this research area is called *Multi-Platform Bluetooth Remote Control*.

4.1 Experiment 1 – Multi-Platform Bluetooth Remote Control

The main focus in this experiment was to take advantage of the capabilities of Bluetooth in a heterogeneous environment, using standard equipment such as mobile phones and laptop computers. We implemented a flexible remote control application, which allows easy integration with Bluetooth-enabled computers and a novel custom map key feature. The server application was implemented as a general-purpose remote control receiver.

We created three main objectives for this experiment:

- Obj. 1-1. To provide an improved experience for presenters, specifically to improve the task of presenting.
- Obj. 1-2. To create the possibility for each user to have their own independent solution (user specific configuration).
- Obj. 1-3. To remove the need for additional hardware, such as cameras, microphones, and remote control devices.

In the next section we present a short summary of the relevant literature that was used as part of this experiment. We continue by going into more detail on the design and architecture, before presenting the material used in our experiment. Next, a description of the procedure is presented, and we look at the results and the discussion based on these results. Finally, the conclusion is presented.

4.1.1 Literature Synopsis

WPANs make it possible for communication between devices that are at short distances from each other (Kamal 2008). Bisdikian (2001) explains that the personal connectivity space resembles a communication bubble that follows people around, and that people can connect to other devices that enter this bubble.

The technology used to create the WPAN for our experiment was Bluetooth. According to Morak et al. (2012), Bluetooth is the most popular personal area network technology available in mobile phones. This technology was selected because it does not need the line-of-sight that is required by infrared, and it has a range that was suitable for our research experiment. In addition, most smartphones and laptop computers are today equipped with Bluetooth, providing good availability.

One of the main research efforts we want to highlight is the work conducted by González- Castaño et al. (2005), who used Bluetooth and WPANs. They created a system used in a museum, which delivered information to the visitors, with an implementation of a Bluetooth Location Network. The visitors use PDAs with external Bluetooth modems and get webpages sent to the device via a Bluetooth push protocol.

Chen et al. (2004) also utilised Bluetooth in their work, where they created a smart meeting room. The Bluetooth technology registered the participants that were attending the meeting. They conducted a demonstration of the implementation and received positive feedback. However, some of the users were concerned about the privacy and information security in the system.

In research closely related to our experiment, Fernández et al. (2006) present one example of work using Bluetooth for short-range communication. They propose a new wireless communication application using Bluetooth and push technology. The *Moviltooth* application delivers personalised information to the users. The server saves user profiles and only sends information that is interesting according to the active profile. Since there are many different display sizes, the server adapts the content of the message to the specific mobile phone types. This application is clearly an interesting

idea, especially the fact that it does not need any local installation. Not only does this make it easier for the users, it also attracts the attention of more people.

We wanted to focus on a system that could provide useful features for meeting presentations. Feldbusch et al. (2003) created a universal remote control system that was designed to work over the Internet. However, we wanted to take a different approach, focusing on the potential for a remote control application that used short-range communication.

The next section continues with a description of the scenario used in our Multi-Platform Bluetooth Remote Control experiment.

4.1.2 Scenario

The overall scenario of the system created for this experiments was as follows: A user installs an application on the mobile device and a server application on the laptop computer. Next, the client application sends a request to connect to the server. This request triggers a dialogue on the server application, asking to either accept or reject the incoming connection. When accepted and the devices are connected, the user is able to send key actions that contain keyboard commands to the server. Finally, the server application installed on the laptop will receive this information over the Bluetooth connection and transform it into commands that are performed on the server. The main use of the system is for presentations, where it makes it easy to move between slides or even switch between open applications. The system supports mouse movement and most keyboard events, with the exception of some potentially harmful commands such as delete. The general use of the system is shown in Figure 4-1.

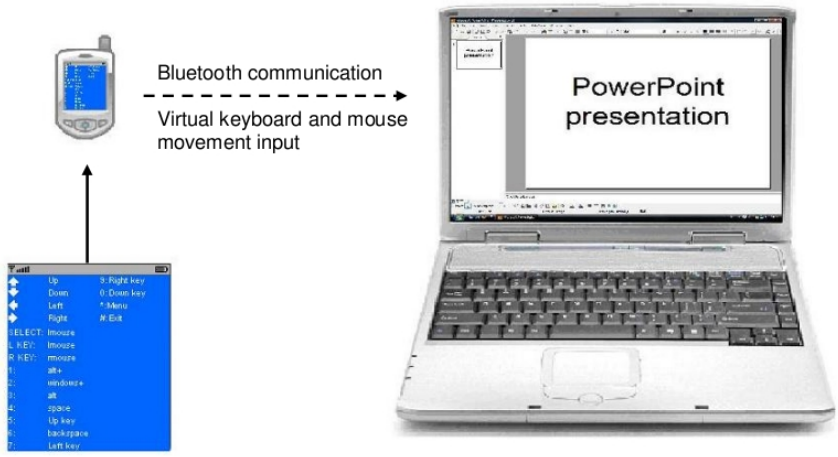


Figure 4-1, use of the remote control.

4.1.3 Design and Architecture

We start this section with a list of the most important requirements, shown as use cases, for the system. Then we provide an explanation of the components and architecture created to implement these requirements in the next section.

Use case	Configure the key mapping of a button on the phone.
Primary actor	Mobile device user
Preconditions	The application is installed and running on the mobile device. The user is inside the *Custom map keys* menu.
Basic flow	1. The user selects the key (button) on the mobile device he/she wants to change. 2. The user pushes *Add key/pointer speed*, *Remove key action* or *Set action type*. 3. A list of actions is presented. The user selects the key action to add or remove. If the user selects a mouse movement, a pointer speed has to be added. 4. Finally, push *Save* to store the configuration in the on-device database.
Post conditions	The configuration is updated on the mobile device. This will change the action executed on the server application when the button is pushed on the mobile device.

Table 4-1, Use Case 1 – configure the key mapping.

Use case	Send key action to server application. A key action object contains information about the action to be performed on the server computer.
Primary actor	Mobile device user
Preconditions	The mobile device must be connected to the server application.
Basic flow	1. The user pushes a button on the mobile device. 2. The server application receives the KeyAction object and transforms it into action.
Post conditions	The action has been executed according to the key mapping sent from the mobile device.

Table 4-2, Use Case 2 – send key action to server application.

Use case	Accept incoming connection to the server application.
Primary actor	Server application user
Preconditions	A mobile client is trying to connect to the server.
Basic flow	1. A new dialogue window is shown when a client is trying to connect. 2. Press *Yes* to accept the connection.
Post conditions	The client and server applications are connected and the mobile device is able to send key actions.

Table 4-3, Use Case 3 – accept incoming connection to the server application.

4.1.3.1 System Components

To target the main requirements, as presented in Table 4-1, Table 4-2, and Table 4-3, we created a system consisting of two main components, namely 1) *the mobile clients* and 2) *the server application*.

The mobile clients were written for Java ME and Windows Mobile. The server application was written in Java SE, and supported connections from both client types. Both mobile clients supported the same features, which included a flexible and user-friendly customisable key mapping system. The user was able to change the actions performed on the server when a button on the mobile device was pushed. While the mobile applications were flexible and allowed a considerable amount of customised configuration, we also added a default set of key mappings. These are shown in Figure 4-2.

Softbuttons:
1. (Left softbutton): Left mousebutton.
2. (Right softbutton: Right mousebutton.

Directional buttons:
Up: Mouse up.
Down: Mouse down.
Left: Mouse left.
Right: Mouse right.

Select: Left mouse button, press/release. Makes it possible to select text.

Numeric keys:
1: Alt + Tab, switch programs.
2: Windows + D, minimize all windows.
3: Alt, useful to access toolbar options.
4: Spacebar.
5: Up arrow (Keyboard).
6: Backspace.
7: Left arrow (Keyboard).
8: Enter.
9: Right arrow (Keyboard).
***:** Esc.
0: Down arrow (Keyboard).
#: Exit.

Figure 4-2, mobile client screenshot with default key actions.

When a button is pushed on the mobile device, the user interface changes the background to black and only shows the action(s) in white text in the middle of the screen, shown in Figure 4-3. This makes it easy for the user to see what action was performed on the server computer. After two seconds the list of all key mappings is shown again.

Figure 4-3, key pressed GUI.

The last component in the system was the server application. The server application listens for incoming connections from the mobile clients. When a device tries to

connect, a pop-up dialogue is shown on the server computer. If the connection is accepted, the client is able to send key actions to the server. The server GUI (Graphical User Interface) includes features such as a network communication window, active connections list, device information, and the ability to disconnect users. All of these features are presented in Figure 4-4.

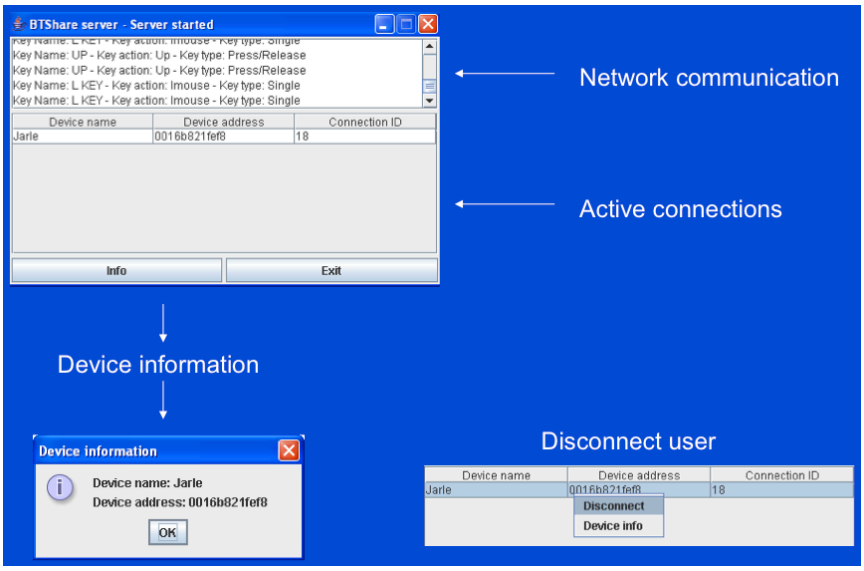


Figure 4-4, server application features.

4.1.3.2 System Architecture

The system was built with focus on three main technical features: 1) *the use of Bluetooth for all communication*, 2) *multi-platform support*, and 3) *allow multiple clients to be connected simultaneous*. Both clients consisted of the same general architecture. Figure 4-5 shows an overview of the system.

The client applications were written for Java ME-based operating systems and Windows Mobile, with C#. For the client applications we were relying on local storage for the key mapping configuration and also the paired devices that were saved. When connecting the client to the server, there was a pairing process. The client saves the Bluetooth address of the server, making it easy to reconnect after the initial setup was completed.

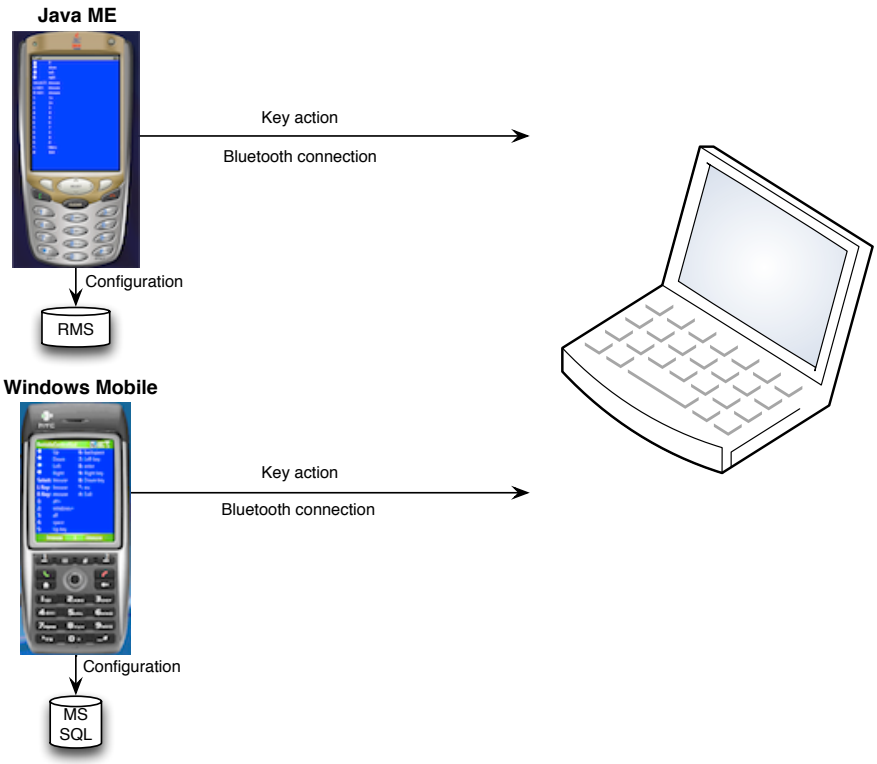


Figure 4-5, Multi-Platform Bluetooth Remote Control system overview.

The clients stored their own configuration, while the server application was only responsible for turning received key actions into events on the server computer.

The client-server model is often a main part of a distributed system, where a client sends requests to a server for a specific service, for example, file system or a database service. A client sends the request and subsequently waits for the server's reply. This is known as request-reply behaviour (Tanenbaum & Van Steen 2002). The client and server components of a system can be organised in different ways, as shown in Figure 4-6.

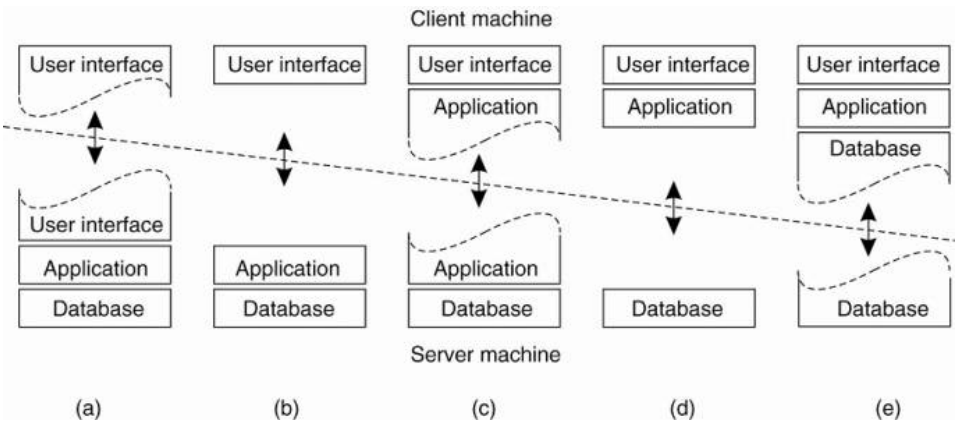


Figure 4-6, client/server organisation (Tanenbaum & Van Steen 2002).

Since our implemented system consisted of business logic divided between the clients and server, we used the *(c)*-client/server organisation. The difference in the system we created for this experiment and the model created by Tanenbaum and Van Steen (2002), is that we did not include any database or other persistent storage on the server application. This was done as a conscious choice to make it possible for multiple mobile devices to share the same server application without the need to share configuration as well.

For the server application we ended up using third-party library called BlueCove[23], which makes it easy to use Bluetooth communication for Java SE applications. One of the main challenges with this application was to support multiple connections from different mobile platforms. We implemented the multi-platform feature by creating a new thread for each incoming connection. During development, we tested the application on four devices simultaneously without any technical issues. This meant we were able to support multiple devices, connected at the same time. Because we also wanted to support two platforms, we created different thread-classes for Java ME and Windows Mobile. The input was handled in slightly different ways on the two platforms. Specifically, we had to incorporate a custom end to the messages sent from Windows Mobile. This was because the Java-based server application was not able to identify the end of the messages and would simply hang, expecting more data to be sent, without the custom implementation.

4.1.4 Material

This section presents the devices used in the experiment. We have chosen to focus on the mobile devices, since it is a vital part of this thesis. It is worth noting that we also did include a laptop computer in the experiment, however, it was not the focus of the experiment or evaluation. The only requirement we had for the laptop equipment was that it supported Java, at least version 1.6, and had Bluetooth capabilities.

[23] http://bluecove.org/

4.1.4.1 Hardware Devices

We tested the system on a range of different devices and the following Java ME-enabled mobile devices were used:

- Nokia 6230i
- Nokia 6233
- Nokia 6680
- Sony Ericsson k600i
- Sony Ericsson w810i

For Windows Mobile we included these devices in the experiment:

- Qtek 8020
- Qtek 9100
- HTC S710
- HTC MTeor

The characteristics of these devices have are presented in Appendix A. The evaluators owned all the devices used as part of this experiment. These devices ranged, at the time, from low-end feature phones (*Nokia 6230i*) to high-end smartphones (*HTC MTeor*). This provides a realistic test scenario to investigate the system.

4.1.5 Participants

The application was tested in two stages. We conducted an initial pilot evaluation where we included one expert user. During this stage, we were able to do a thorough evaluation of the features of the system, both client and server, and also get some initial feedback on the user documentation (Presented in *Appendix B*, for Java ME, and *Appendix C*, for Windows Mobile). We received valuable feedback that resulted in fixing several bugs and altered the default configuration that is initialised when the application is first installed on the mobile device. We were also able to improve on the menu system in the client application during this initial evaluation process.

In the main evaluation, we conducted the user experiment in a large Norwegian IT company (5000+ employees) located in Oslo. We had a total of 16 users, all of which

were computer literate and hailed from an industrial background. The age of the users were between 20 and 50 years. We continue to the next section, where we present the procedure used in the evaluation, before presenting the results.

4.1.6 Procedure

The participants were asked to install the mobile client on their device and copy the server application to their own laptop. They were given a brief system documentation (see Appendix B and C) explaining how to install the applications, the main features of the system, and also how to set-up Bluetooth properly. When the installation was completed, the participants completed a set of tasks relating to the main features of the system. All the tasks for the user evaluation are presented in Table 4-4.

Installation	
1	Read the user documentation to understand how to do the initial pairing and installation.
2	Add a new Bluetooth device. Do the initial pairing with the phone and computer.
3	Install the application on the phone. Java platform: *BTShareMobile.jar*, Windows Mobile platform: *BTShareMobileSetup.cab*.
4	Install the server application. Copy the files *BTShareServer.jar*, *bluecove.jar* and *run server.bat* to the same folder and then click on *run server.bat*.
Normal Use	
5	Read the user documentation to understand how to do the initial set up of the system and connect to the server application.
6	Connect to the server application.
7	Test the default keys to see if everything works properly.
8	Disconnect from the server application using the phone.
Use of Features	
9	Read the user documentation to understand how to custom map a key.
10	Create a new Windows shortcut (see user documentation to find examples) on the key *'8'*. Remember to first remove the key that is already there and set the action type to combination.
11	Connect to the server application again and test the new shortcut by pressing 8.
12	Change the pointer speed on the Left (Mouse left) movement to 50.
13	Connect to the server application again and test the new pointer speed by pressing left on the directional key.

Table 4-4, user instructions.

Number	Statement
Installation	
1	In general, the user documentation is easy to understand.
2	The initial pairing was not explained well enough in the user documentation.
3	It is easy to install the application on the phone.
4	The installation of the server application could be faster.
5	The installation of the server application could be explained better.
Normal Use	
6	It is difficult to set up the system (Search for and save the Bluetooth devices).
7	It is easy to connect to the server application.
8	The default keys are not very useful.
9	Some of the default keys do not work properly.
10	The user interface on the remote control is helpful.
11	The user interface on the remote control is easy to understand.
12	The server application user interface is difficult to understand.
13	The server application does not provide the options I would like to use.
14	The Remote Control is responsive with very little delay.
15	I think the application is useful for presentations.
16	I would like to use the application in the future.
Use of Features	
17	The user documentation has a good explanation of the features.
18	The *custom map key* menu is difficult to understand.
19	It is difficult to understand the *action type* menu.
20	The user documentation does not explain how to use the *action type* menu well enough.
21	The *custom map key* feature is flexible and useful.
22	The *pointer speed* menu is easy to understand.

Table 4-5, WPAN questionnaire

fter the tasks were completed we gave each user a questionnaire containing 22 statements. It consisted of a roughly equal number of positive and negative statements to avoid bias. Moreover, the statements were categorised, as suggested by (Kitchenham & Pfleeger 2002b). We selected three groups: *Installation*, *Normal Use*, and *Use of Features*.

The statements were created to investigate several of the common usability attributes (International Organization for Standardization 1998), such as effectiveness.

Furthermore, we focused on a usability challenge for mobile devices identified by Zhang and Adipat (2005). In this experiment we investigated *connectivity*, which in the context of this experiment was the integration of mobile devices and a laptop using WPAN technology.

For each statement the users indicated the answer on a six-point Likert scale, ranging from *1-Strongly Disagree* to *6-Strongly Agree.* We selected an even numbered scale to avoid the possibility of users only answering neutral to all statements.

In the next section we present the results gathered from all the statements (Table 4-5).

4.1.7 Results

The overall results are presented in Table 4-6. We calculated the average score and standard deviation. In this section we continue by going into more detail on each statement.

Number	Avg. score (out of 6)	Standard deviation
1	4.8	0.62
2	3.3	1.66
3	5.2	0.64
4	4.8	1.06
5	4.5	1.44
6	5.1	0.85
7	5.5	0.63
8	5.2	0.56
9	5.2	1.17
10	5.3	1.01
11	5.1	0.90
12	4.3	1.78
13	5.3	0.70
14	5.4	0.72
15	5.6	0.62
16	5.1	0.93
17	4.8	0.60
18	5.0	0.96
19	5.2	0.87
20	4.3	1.15
21	5.2	0.70
22	5.5	0.78

Table 4-6, overall results from user experiment.

To investigate the results of from the evaluation, we categorised the statements into four groups. According to Zhang and Adipat (2005) these usability attributes, which we have grouped the statements into, are important aspects of usability in software applications.

Learnability

We start with statements regarding learnability and in this experiment these statements mostly dealt with performance of specific tasks completed by the users and how well the system was explained in the user documentation. One challenge in this experiment was to make the system easy to learn. Also, we wanted to provide various features that had to be explained in detail in the user documentation. The installation of the components in the system, both the mobile client and server application, were tasks conducted as part of our evaluation.

We categorised these statements based on a specific task, for example *installation*, and how this task was explained in the user documentation. In statement 1 (*In general, the user documentation is easy to understand*) the user documentation in is the main focus. The average score based on the result from the user test is 4.8 out of 6. We see the same trend in statement 5 (*The installation of the server application could be explained better*), 17 (*The user documentation has a good explanation of the features*), and 20 (*The user documentation does not explain how to use the action type menu well enough*), where the average scores were 4.5, 4.8, and 4.3 respectively. These parts of the system and the user documentation, including installation of server application and the explanation of features, were explained satisfactory in the user documentation as shown in the results.

However, for statement 2 (*The initial pairing was not explained well enough in the user documentation*) we received a few comments on difficulties understanding the task of pairing the Bluetooth devices. This was also reflected in the average score received (3.3 out of 6). The specific problem several users had was related to installation of the system due to incompatible Java versions.

Statement 6 (*It is difficult to set up the system, search for and save the Bluetooth devices*) further investigates the tasks of searching and saving Bluetooth devices. This

was done to make it easy to connect to the same server multiple times because the Bluetooth address was persisted on the client. The average score was 5.1 out of 6, and it shows that the users managed to set up the system and did not find it too complicated. However, a few evaluators commented that the Bluetooth pairing process was not explained well enough in the user documentation.

We see two main improvements that can be made based on our results within learnability, namely to update user documentation (Bluetooth pairing better explained) and simplify the installation process. This proved too cumbersome for many users and we need to clearly specify the required Java version.

Effectiveness

The next group of statements we want to present is effectiveness, which deals with how effective the users are when using the system.

We focused on two specific issues, where statement 3 (*It is easy to install the application on the phone*) and 4 (*The installation of the server application could be faster*) focused on installation, and specifically of the client and server application. The statements received an average score of 5.2 and 4.8. The server application was designed to be very simple to install (simply copy a few files) and did not require any local storage. We believe this feature was well received, and provides a fast and easy way for users to install the application. This is s also verified by the results from the evaluation. For the mobile client, there were some issues with transferring the application to the mobile device. After the file was available on the device, the installation did not cause any further problems. For future updates to the application, the use of application marketplaces, such as the App Store and Android Marketplace, will improve the user experience of installing applications on the mobile devices.

The next statement investigated within this topic is number 14 (*The Remote Control is responsive with very little delay*), where we wanted to know if the remote control was perceived as responsive. It is very important that the application has little delay, i.e. it should seem almost instant to the user. We had tested this previously with good results, and the score from the questionnaire, with an average score of 5.4 out of 6, verifies that the users experienced the application as responsive during the evaluation.

Comprehensibility

Comprehensibility focuses on how easy it is for the evaluators to understand the information presented in the system. These statements target specific features, such as the user interface. Statement 10 (*The user interface on the remote control is helpful*) and 11 (*The user interface on the remote control is easy to understand*), deals with the user interface on the mobile phone when it is connected to the server application. The average score for these statements were 5.3 and 5.1, indicating that the user interface was well received by the evaluators and it was a helpful part of the application.

Moving on the specific features on the mobile client, the users were asked to rate how easy it is to comprehend the *custom map key* menu (statement 18, *The custom map key menu is difficult to understand*), the *action type* menu (statement 19, *It is difficult to understand the action type menu*), and the *pointer* menu (statement 22, *The pointer speed menu is easy to understand*). The custom map key menu is where the user can configure each button on the phone, the action type menu makes it possible to change the sequence of actions for a specific button, while the pointer menu provides an option to change the mouse speed. These statements received an average score of 5.0, 5.2, and 5.5, which indicates that the users generally understood these features in the evaluation.

In the final statement in this category, 12 (*The server application user interface is difficult to understand*), we focused on the user interface on the server application. The server is a very simple application running on a Bluetooth-enabled computer. For this application the user interface is only one window that contains information about connected users and network traffic. With an average score of 4.3, most users found the interface easy to use. However, a few users did comment on the difficulties getting the application installed on their laptop and we believe this affected to overall result in regards to this statement. This issue is directly related to the previously mentioned problem with incompatible Java Versions.

User Satisfaction

With user satisfaction we wanted to see the attitudes of the users towards to system. This is a very important aspect, dealing with issues such as how easy they were able to use certain features of the implemented remote control system.

As part of the system, we wanted to provide a useful application that included a flexible and highly configurable key mapping system. However, it was also very important to add a default configuration that was targeted at presentations, since this was one of the main objectives for this experiment. Statement 8 (*The default keys are not very useful*) and 9 (*Some of the default keys do not work properly*) target this issue, and asks about the default configuration that is included when the application is installed. These statements received an average score of 5.2, indicating that the users were generally satisfied with the features.

Statement 7 (*It is easy to connect to the server application*) asks about the process of connecting to the server applications. After the initial pairing was completed, this was a simple task of selecting the server from a list of Bluetooth addresses persisted on the client application. With an average score of 5.5, the majority of users found this task easy to complete.

The next statements we want to highlight within user satisfaction, is statement 15 (*I think the application is useful for presentations*) and 21 (*The custom map key feature is flexible and useful*). Both statements focus on the usefulness of the system, in regards to the use in presentations and the configuration options. Average scores of these two statements were 5.6 and 5.2 out of 6, which is an encouraging result. This indicates that the users found the application useful and that it works well for presentations.

Finally, the last two statements ask the users their general opinion on the system, and if they would like to use in the future. Statement 13 (*The server application does not provide the options I would like to use*) and 16 (*I would like to use the application in the future*) received an average score of 5.3 and 5.1, which indicates that the users were satisfied with the features and also that they would use it in the future. The interest in the system during the evaluation was generally high, and this is certainly a positive result that would encourage further development of the system.

Finally, we also received a few general comments from the evaluators, which provides a good summary of the results:

1. The application is easy to understand and use.

2. The application is useful and most of them (the evaluators) would use it for presentations.
3. The main problem was the installation. The lack of specifying the version needed on the computer created most of the problems. Also, the error messages could be more informative.
4. The initial pairing of the devices caused problems for some users. Not everyone understood the meaning of setting the Bluetooth device to discoverable. This is something that should have been explained better in the user documentation.

4.1.7.1 Limitations

We conducted the user evaluation with 16 users. If there were significantly more users evaluating the system, the results might be different. However, we believe that we were able to gather enough data to provide decent results. Additionally, it is also important to note that we were able to provide results from employees of a large company, which are in the target group of the application because it was mainly focused on meetings and presentations.

4.2 Discussion

The overall goal identified for this experiment was to implement a system to provide useful features in a meeting room setting. Specifically, we targeted presentations and the ability to use standard hardware (mobile phones and laptop) to offer a flexible and highly configurable solution.

Saha and Mukherjee (2003) identified *heterogeneity* and *integration* as two of the main challenges within pervasive computing. We created a remote control system, using Bluetooth to integrate normal mobile devices with laptop computers. Moreover, the heterogeneity aspect was included as part of the multi-platform support. We wanted to create a system supporting multiple platforms, and at the time the experiment was conducted Android and iOS was not available on the market. This is the reason why we chose to use Java ME and Windows Mobile.

Additionally, we also wanted to investigate *connectivity*, which is identified as one of the important challenges of usability within applications running on mobile devices (Zhang & Adipat 2005). In our experiment we connected the devices using short-range communication. We also supported multiple simultaneous connections, which added a useful feature where several presenters could use their own mobile device at the same time. The results from the evaluation were also positive towards the connectivity of the system.

The implemented system used the same basic design and architecture for both the Java ME and Windows Mobile client. This meant we could reuse all the work and experiences gained from creating the Java Me client, when we implemented the Windows Mobile applications. An important aspect of software projects is the overall design and architecture of the system. We found that in most cases the majority of this design work was platform independent and could simply be reused on all the platforms. However, we also acknowledge that in certain corner cases this aspect does not apply, such as differences in how the platforms supported Bluetooth that meant we needed to add platform specific features. We experienced these issues when it came to integrating the Windows Mobile and Java ME platforms on the same server, and specifically with the way the two platforms handled the Bluetooth communication differently. By including different entry points to the server for each mobile platform, we were able to provide a solution for this problem.

The main idea behind the server application was to create a very simple system that requires minimal user input and does not store anything on the computer. It simply receives commands from the phone and transforms them into actions. Because we wanted to keep the server application as simple as possible, we added the ability to store all configuration values on the mobile devices. The main drawback from this solution is that the application requires storage space on the mobile device and we also received feedback that an import/export feature would be a useful upgrade to the system. Additionally, we acknowledge the need to add a touch-based user interface to support multiple smartphone models.

One of the issues we had to deal with when creating the server application was how to authorise incoming requests for connections from various clients. We solved this by

including a simple pop-up dialog on the computer the server application was running, where one had to either accept or reject the incoming connection, shown in Figure 4-7. This is different from how, for instance, Windows has solved this issue, where it accepts the connection automatically once the Bluetooth devices are paired.

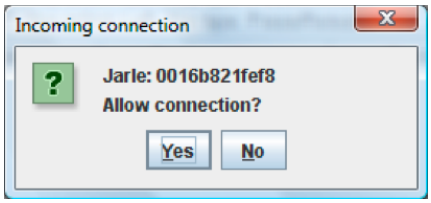


Figure 4-7, accept incoming connection.

For the server application, we see the potential to improve the user documentation and thereby eliminating some of the confusion that were experienced during the evaluation. It is also a possibility to use technology such as *Java Web Start* to minimise the manual work with the installation process of the server.

4.3 Conclusion

In this experiment we focused on the area of WPAN, where we evaluated usability of applications running on mobile devices. For our experiment, *Multi-platform Bluetooth Remote Control*, we defined three objectives.

The first objective (*To provide an improved experience for presenters, specifically to improve the task of presenting*) was to add useful features in a meeting room setting. We focused on the task of presenting, where we created a very flexible remote control system that offered a customisable key-mapping feature. A novel aspect to our work was the created heterogeneous system using Bluetooth for the communication between the components. It differs from previous research efforts, such as the work done by Fernández et al. (2006) who focused on the server side implementation. Additionally, our research also provides a remote control system that is different than the universal remote control implemented by Feldbusch et al. (2003). They created a system that utilised the Internet, while our solution targeted a local service and was created for a meeting scenario, specifically to be used with devices in close proximity.

One of the main contributions was the previously mentioned flexible custom map key feature, where each button on the phone could be configured to any keyboard combination on the receiving laptop device. Furthermore, as shown in the results section of this chapter, the general user feedback on the usability was positive. The users found the application easy to understand and use. Moreover, the features, the flexible key mapping and the overall integration between the devices, were perceived as useful by most of the evaluators. The main concern raised during the evaluation was user documentation issues, such as the lack of specifying the required Java version and not enough detail on the Bluetooth pairing process. Creating an updated version of the user documentation for a next release of the system would provide a better user experience.

For the second objective (*To create the possibility for each user to have their own independent solution*) we provided a solution where each user could connect to the system and use separate configuration values. The overall goal was to create a system where the configuration settings were stored independently. We were able to implement this feature by making the clients responsible for all the persisted data. Moreover, this also had the benefit of making the server application very simple, since it did not require any database or file storage.

One of the main aspects of the implemented solution was that we were able to connect multiple users, even at the same time, to one server application. And importantly, each user had separate configuration. The values stored on the clients, both Java ME and Windows Mobile, were the key-mapping configuration and Bluetooth addresses. During the evaluation, the users found this feature useful. It provided a very simple way of including user-based configuration and offers an easy installation process. One aspect that we would like to improve with this feature is to allow for import/export of configuration values, this will make it possible to keep the configuration values when changing to a new mobile device.

The final objective (*To remove the need for additional hardware, such as cameras, microphones, and remote control devices*) was another important aspect of our work. We wanted to focus on using standard components, namely mobile devices and a Bluetooth-enabled laptop. These were implemented to run without any fixed

infrastructure, using short-range communication technology. With the market offering a variety of mobile platforms (Gartner 2012) and the trend toward heterogeneous environments (Ryan & Gonsalves 2005), this is certainly an important aspect. In our system we had support for two platforms, namely Java ME and Windows Mobile. We have not seen existing research in this area focusing on creating a heterogeneous system.

One of the features of the system we found particularly appealing with the system was that it only required a normal mobile phone and a Bluetooth enabled computer. The idea of only needing standard equipment is an aspect we wanted to explore further. Therefore, we include it as a main theme in this thesis, where the experiments use standard smartphone devices.

Chapter 5 – Context-awareness

This chapter presents the topic of Context-awareness, and we have chosen to look at it from two different angles. We start by presenting the topic of *Context-aware Applications* with an experiment called *Context-aware Meeting Room*. After this, we introduce a second experiment, *PainDroid: an Android Application for Pain Management*, which provides information on the topic of *Smartphone Sensors*, and how these can be used to gather context data and provide context-awareness in a system. A discussion and conclusion on the overall topic of Context-awareness is presented towards the end of this chapter, taking both experiments into consideration.

In the next section, we start by looking at the *Context-aware Meeting Room*. Later in this chapter, section 5.2, we go into more detail on the experiment investigating smartphone sensors.

5.1 Experiment 2 – Context-aware Meeting Room

In regards to the three basic context-aware systems functionalities, as presented by Loke (2006), *Context-aware Applications* are closely related to *Thinking* and *Acting*. *Thinking* utilises the collected data and makes sense of it, while *Acting* takes the appropriate actions. These two functionalities are important aspects in this experiment, where we created a Context-aware Meeting Room.

The main idea behind the project was to create a system that used proximity information to register devices in the meeting room, and provide features such as automatically pushing meeting notes to the participants. Hence, to fulfil the goal of this experiment we created two main objectives:

- Obj. 2-1 To handle time consuming meeting tasks automatically, such as meeting registration, information sharing, and meeting history.
- Obj. 2-2 To provide context-aware and useful information without the need for additional hardware in the meeting room.

In the next sections we will provide more information about the experiment, starting with related work.

5.1.1 Literature Synopsis

An important idea behind context-aware applications is the attempt to understand the situation in which a device operates, thus resulting in a better and more efficient computing strategy (Kamal 2008).

In computing systems, one of the most widely used context sources is location (Loke 2006). Research efforts in this area include work conducted by Mantoro and Johnson (2003), Sohn et al. (2005), Eagle and Pentland (2006) and Ferris et al. (2010). Others have focused on providing additional context sources, such as the research presented by Raento et al. (2005) with the *ContextPhone* project. The *ContextPhone* platform was implemented to help developers create applications that integrate into existing technologies and the everyday lives of the users.

There is also a considerable amount of research focusing on creating smart spaces, which is closely related to our experiment. Jaimes and Miyazaki (2005) present an experimental Smart Conference Room, containing cameras, microphones, displays, and capture devices. In similar work, Chen et al. (2004) created a smart meeting room, where the system provided features such as presentation, lighting control, and greeting service. Additionally, Ahmed et al. (2005) developed a meeting detector model that can determine the beginning and end of meetings. Parviainen et al. (2006) also included the use of speech detection, and the system was utilised in a lecture room setting. They developed a source localisation system to identify the participant who is speaking and at which time.

The previous research efforts that focused on smart spaces require additional equipment such as cameras, microphones, and external sensors installed in the rooms/spaces. This is different to our approach, where we did not require any other equipment other than standard smartphones and a Bluetooth-enabled laptop. Additionally, instead of focusing on location we utilise proximity in combination with calendar data to provide context sources for the system.

5.1.2 Scenario

In a business environment, it is an everyday occurrence to attend meetings. There is a considerable amount of work administering these meetings, including keeping track of the daily schedule, meeting participants, write meeting minutes, and so on. We wanted to implement a system that can help the users by providing useful features, specifically relying on context to automatically present information for both the meeting presenter and participants.

The system works as follows: the user registers the device in the system, entering information such as first and last name, and the Bluetooth address of the mobile device, as presented in Figure 5-1.

Figure 5-1, user registration.

For the next step in the process, a meeting is scheduled in the online calendar. Just before the meeting is about to start, the participants approach the meeting room. When in proximity of the room, a welcome message is automatically sent to the mobile device. After all meeting participants are present, the presenter can start the meeting. When the presenter begins the presentation, the registered participants receive an updated message on their mobile device containing additional and useful information related to the first slide.

The system works by sending out notes, which the presenter has added to the presentation, to all the registered devices that are present in the meeting room. The client installed on the mobile devices also gives the participants the opportunity to move forward or backward in the received meeting notes. Additionally, it permanently stores notes on the devices, making it easy to review notes from previously held meetings.

5.1.3 Design and Architecture

When designing the system we created a list of requirements, called a backlog in the scrum framework. In this section we continue by presenting the most important requirements, which are shown as use cases. These requirements are directly related to the overall objectives. However, they provide a more detailed view of how the system works and what features it provides.

Use case	Receive meeting notes automatically when the presenter proceeds to the next slide.
Primary actor	Meeting participant
Preconditions	The participant must be registered for the meeting and in proximity of the meeting room.
Basic flow	1. The presenter moves forward in the presentation to the next slide. 2. A message is sent to all connected and registered mobile devices, containing the current meeting note.
Post conditions	A meeting note is shown on the mobile device and has been stored locally in the history feature.

Table 5-1, Use Case 1 – receiving meeting notes.

Use case	View previous meeting notes using the history functionality of the client.
Primary actor	Meeting participant
Preconditions	The participant must have attended a meeting using the client application. The client application is running.
Basic flow	1. From the main menu, the user selects *History*. 2. A list of previous meetings is shown, identified by the time when the meeting was held. The user selects a meeting. 3. A new list of the meeting notes is presented on the device. The user can select a specific note to view the entire text.
Post conditions	A meeting note from a previous meeting is shown on the mobile device.

Table 5-2, Use Case 2 – meeting note history.

Use case	Automatically find and connect to the registered devices.
Primary actor	Meeting room server
Preconditions	The device must be registered in the system and for the current meeting.
Basic flow	1. The meeting room server is started, and the presenter logs in with his e-mail. 2. A list of upcoming meetings is shown, and the presenter selects the current meeting and pushes *Start meeting*. 3) The meeting room server begins to search for registered devices by pinging their Bluetooth address. 4) Each device that is found is added to the list of *Connected participants.*
Post conditions	All devices are found and the continuous search for devices within close proximity is stopped.

Table 5-3, Use Case 3 – find registered devices.

5.1.3.1 System Components

To fulfil the requirements described in the use cases, we designed a proof-of-concept system consisting of three main components: 1) *presenter application*, 2) *participant applications*, and 3) *the server applications*.

The presenter application was created for the Android platform. It was implemented to provide useful features for the meeting presenter, such as moving to the next or previous slide, and starting and ending the meeting.

The second component consisted of the participant applications. These applications were able to receive meeting notes for each new slide displayed by the presenter. During the meeting, the participant can browse back and forth in the received notes, but never further than the most recent slide. The participant applications were developed for Java ME, Windows Mobile, iPhone (iOS), and Android. The Java ME and iPhone applications are shown in Figure 5-2.

Figure 5-2, participant applications, Java ME and iPhone.

The last component of the system is given by the server applications. We implemented two server applications, namely the cloud-based server and the standalone server running on a computer within the meeting room. Both applications were written in Java SE (Standard Edition). For the cloud-based application we used the Google App Engine. Responsibilities for this component included storing information to the datastore and integrating with external systems, including Google Calendar and Google Docs. It also handled all the business logic surrounding the delivery and status of meeting note messages. We chose to use the Google App Engine for our proof-of-concept because it provides a close integration with other services that we utilised, such as calendar and authentication.

The second server application was running on a computer inside the meeting room. It was responsible for connecting to the registered meeting participants in close proximity. Additionally, this application also handled the connection with the presenter application, providing features such as the option to move to the next and previous slide. The user interface of the server application included a list of calendar appointments, meeting details, and connected participants, as shown in Figure 5-3. The list of registered participants was retrieved from the cloud-based server.

The next section presents more technical details about the system, presenting the *System Architecture*.

Figure 5-3, meeting room server.

5.1.3.2 System Architecture

The system was built using a client/server model, where we used Protocol Buffers[24] as the data format for the network communication. Protocol Buffers are a contract-first technology, meaning that we started by defining a contract (proto-file in this case) and then generated source code based on the contract. In our experience, one of the main benefits of using a contract-first service was easy maintenance. This is because the shared contract loosely couples the components, i.e. the server and clients in our system. If a change was needed in the contract, we simply edited the proto-file and regenerated the code for each platform.

When we started implementing the proof-of-concept all platforms except Java ME had support for Protocol Buffers. To enable Java ME support, we created our own open source implementation[25], making it possible to use Protocol Buffers on Java ME-enabled devices.

As presented in the previous section, the system contained three main components. Figure 5-4 shows an overview of the system, and includes how the various parts of the system communicate with each other.

[24] http://code.google.com/p/protobuf/
[25] http://code.google.com/p/protobuf-javame/

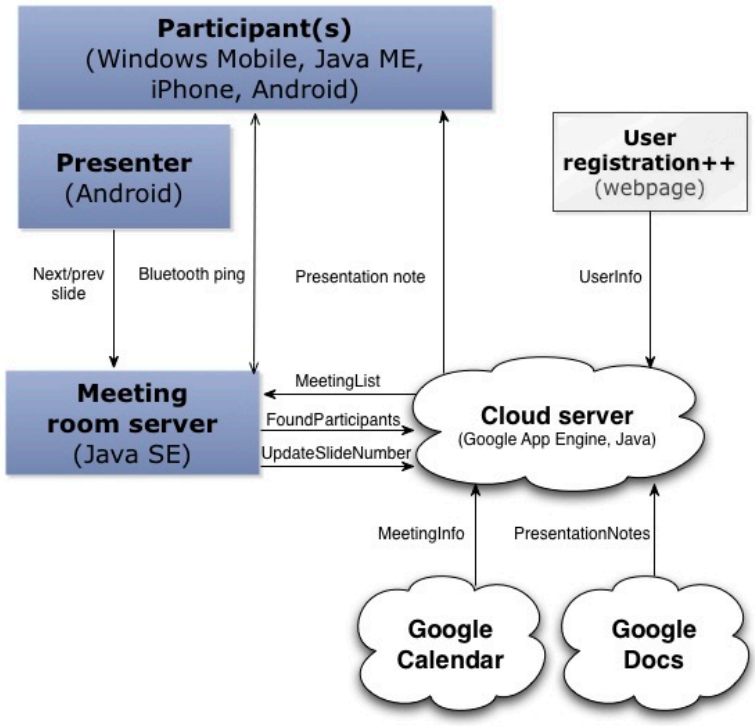


Figure 5-4, Context-aware Meeting Room system overview.

One of the most important aspects of the system was the use of Bluetooth to discover meeting participants. We took advantage of the short-range features of Bluetooth, which is usually around 10m, and performed a continuous search from the server located inside the meeting room. We called this feature *Bluetooth ping* (See Appendix D for source code). It worked by pinging, i.e. sending a request to a specific service on the devices, and waiting for a response. The Bluetooth addresses were stored in the system when the users were registered. By pinging the *Bluetooth hands-free service*, we were able to find the devices within communication reach. This had the advantage of not requiring any installation or configuration on the mobile clients, since the hands-free service is usually available on mobile devices. All devices used in our evaluation supported this feature. Additionally, we were also able to support devices that are hidden, meaning that they cannot be found with a normal Bluetooth search. This was particularly important for the Android and iPhone client, since these devices cannot maintain a permanent visible status. These platforms will automatically switch the Bluetooth status to hidden after a short period of time.

After the clients were found by the Bluetooth ping feature and connected to the system, we used WLAN communication for sending the meeting notes to the client. When we

knew the clients were registered and connected to the system, there was no practical need for Bluetooth. With WLAN we had considerably higher bandwidth, as 802.11b and 802.11g support bandwidth of 11 and 54 Mbps respectively, compared to Bluetooth that provides a basic data rate of 1 Mbps (Hansmann et al. 2000).

When the clients were connected and the meeting had started, we provided the meeting note features. The cloud-based server handled both updating current slide status and sending out notes to the participants. Specifically, it was responsible for sending the content to the mobile devices, and we had the advantage of the flexibility and scalability due to the cloud service (Google App Engine). Additionally, we found it very useful to provide a cloud-based service that did the heavy lifting when it comes to retrieving the calendar information and combining this input with the current presentation and the registered participants. By including this on the common cloud-based server, we could reuse the service for all client implementations.

5.1.4 Material

This section presents the most important devices and platforms used as part of the experiment. They are also presented in more detail in chapter 3, section 3.4.

5.1.4.1 Cloud Computing Platform

The Google App Engine was used to host the cloud-based application. We selected this cloud-based platform because it is well integrated with other services, such as authentication, calendar, and Google Docs, that we also used as part of the system.

With our server application we used the storage feature and provided access to the system through HTTP. The data sent from the cloud-based server to the mobile devices were in the Protocol Buffer format. By installing the server in a cloud-computing environment, we were able to use the service on different networks without requiring any local network configuration, such as opening firewall ports. This feature was particularly useful when we accessed the system using EDGE/3G in addition to WLAN.

5.1.4.2 Hardware Devices

The devices used in our test were running four different platforms, namely Java ME, Windows Mobile, iOS, and Android. The Java ME enabled mobile device was a *Nokia E61*. Windows Mobile was included with a device called *HTC Tytn II*, and we also included an *iPhone 2G* in the experiment, using iOS. All of these devices were used for demonstrating the participant clients. Both the presenter and participant applications were available for the Android platform with the *HTC Magic* device. The devices are all listed in Appendix A.

The devices were all smartphones, and they were selected because of the features they offered. For example, network communication and Bluetooth capabilities were two very important aspects that each device supported.

5.1.5 Participants

We started with an initial pilot evaluation, which was done with 12 users, and consisted of a full demonstration of the system as well as a review of the questionnaire. During this process we were able to fix a few technical issues and improve the questionnaire, removing two ambiguities in the statements by rephrasing them appropriately. In the main user evaluation, there were a total of 40 users participating. All of them were aged between 20 and 55, and all had previous knowledge of mobile phones and mobile communication. Moreover, the evaluation participants were recruited from both an academic and corporate environment.

5.1.6 Procedure

As part of the evaluation, we asked each participant to complete a set of tasks that consisted of the steps presented in Table 5-4.

Before Meeting	
1	Open the browser on the following URL: http://context-aware-meeting-room.appspot.com/meeting
2	Register your mobile device with the provided Bluetooth address and your personal email address.
During the meeting	
3	When entering the meeting room, verify that the welcome message is received.
4	During the presentation, verify that meeting notes are received when the presenter changes slide.
5	Verify that the meeting note received is in synch with the current slide.
6	Move to a previously received meeting note.
7	Move to most recent meeting note.
8	Disconnect the mobile client from the current meeting.
After meeting	
9	After the meeting has ended, use the history feature to view notes from a previously held meeting.
10	Navigate back and forth in the notes from this meeting.

Table 5-4, user instructions.

After the initial registration and setup, we started a test meeting where we held a 30-minute presentation that included meeting notes from 12 slides pushed to the participants' devices (Figure 5-5). The evaluation was conducted on a shared-WLAN, which was in active use by other students not part of the evaluation in addition to our test devices. This provided a realistic scenario, where the wireless network was shared for various tasks.

Figure 5-5, evaluation presentation.

After the presentation was completed, we asked each participant to answer a questionnaire. The questionnaire contained 15 statements and consisted of a roughly

equal number of positive and negative statements, randomly divided between the categories.

The statements in our questionnaire were categorised, which is recommended by (Kitchenham & Pfleeger 2002b). In this experiment we included the following groups: *User Interface*, *Performance*, *User Features*, *Context-aware Information*, *Application Features*, and *Overall Impression*.

For each statement, the users indicated the answer on a four-point Likert scale, ranging from *1-Strongly Disagree* to *4-Strongly Agree.* Similar to the previous questionnaire, we chose an even number to avoid the evaluators taking a neutral position to all statements. Moreover, we used a four-point scale instead of six, which was used in experiment 1, because we found it difficult to differentiate between *somewhat disagree/agree* and *disagree/agree*.

As with the previous experiment, we investigated common usability attributes (International Organization for Standardization 1998). Additionally, we also focused on *mobile context* and *restrictive data entry methods*, which are identified by Zhang and Adipat (2005) as important and unique challenges to mobile devices.

Table 5-5 presents the statements provided in the questionnaire we used in the evaluation. The following section continues by presenting the results gathered from the user evaluation.

Number	Statement
	User Interface
1	It is easy to see the available functions.
2	The features of the application are hard to use.
	Performance
3	Browsing back and forward in the current meeting notes is quick.
4	The presentation notes displayed on the mobile device was synchronized with the audio presentation/presenter's narrative.
	User Features
5	I was not able to register as a participant in the web application.
6	I was able to look at next and previous notes during the presentation.
	Context-aware Information
7	When the presenter started the meeting my mobile device was found and I began receiving meeting notes once in presentation mode.
8	The close integration with Google calendar is an inconvenience. I am not able to use the system without changing my existing or creating a new e-mail account at Google.
9	I do not like sharing my personal information (like my name and e-mail address) to a service that stores the information in the cloud.
10	Storing meeting information in the Google cloud and combining this with personal information on the device is a useful feature.
	Application Features
11	I find the ability to navigate notes during the presentation useful.
12	I do not find the *meeting note history* functionality useful.
13	I do not find the presenter/participant model useful for a meeting environment.
14	I would like to receive more content during the presentation (images, videos, files, web-pages etc.).
	Overall Impression
15	I find the context-aware meeting room application useful.

Table 5-5, Context-aware Meeting Room questionnaire.

5.1.7 Results

The overall results from the user evaluation are presented in Table 5-6. It is worth noting that we have calculated the average score for each question based on positive statements. This means, for the negative statements we gave *strongly disagree* a score of 4, while for positive statements *strongly agree* also had the same score. This has the advantage of making the numbers easier to compare, because they are all presented on the same scale.

Number	Avg. score (out of 4)	Standard deviation
1	3.58	0.55
2	3.50	0.56
3	3.58	0.55
4	3.13	0.69
5	3.75	0.59
6	3.78	0.48
7	3.83	0.39
8	3.18	0.90
9	2.93	0.69
10	3.38	0.49
11	3.55	0.50
12	3.25	0.63
13	3.33	0.62
14	3.53	0.64
15	3.33	0.57

Table 5-6, user evaluation results.

To investigate the results of the evaluation we have divided the statements into three groups. The groups are created in regards to usability attributes, identified by Zhang and Adipat (2005), and provide important aspects of usability in software applications.

Effectiveness

The first category of statements deals with effectiveness. In these statements we wanted to investigate how effective the users were with specific tasks in the implemented proof-of-concept.

When the evaluation started, the users were asked to complete an initial registration. Statement 5 (*I was not able to register as a participant in the web application*) covered this aspect, and the average score recorded was 3.75 out of 4, indicating that the majority of users were able to register without any problems.

After the initial registration, the users joined a meeting. Statement 7 (*When the presenter started the meeting my mobile device was found and I began receiving meeting notes once in presentation mode*) asked the users whether the application started receiving information when the presenter began the meeting. For this statements respondents were unanimous in giving positive answers, with an average score of 3.83.

Finally, when the meeting was started the presentation notes were pushed to the mobile devices of the users. Statement 4 (*The presentation notes displayed on the mobile device was synchronized with the audio presentation/presenter's narrative*) asked about the notes being received on the mobile devices when the presenter changed slide in the presentation. This feature is certainly an important aspect of the overall system. If the notes received are not in sync with the current slide, the information provided can prove to be less useful to the meeting participant. With a score of 3.13 out of 4, there is a positive bias based on the answers given by the users.

The statements presented show that the users were able to register and receive the presentation notes. The issues experienced by certain users in this part of the evaluation were caused by network latency. We conducted the test on a shared wireless network, and in a few occasions we experienced stability issues. However, these minor issues did not cause any major problems because we had added fall-back mechanisms, such as an alternative method for registration.

Both statement 4 and 7, which were previously presented, are directly related to objective 2-1 (*To handle time consuming meeting tasks automatically, such as meeting registration, information sharing, and meeting history*) where we took advantage of information from the online calendar and proximity registration to automatically provide functionality. These features, as shown with the results, were well received by the evaluators.

Additionally, statement 3 (*Browsing back and forward in the current meeting notes is quick*) and 6 (*I was able to look at next and previous notes during the presentation*) also deals with the meeting note features. With average scores of 3.58 and 3.78 out of 4, the results indicate that the notes functionality performed satisfactory during the evaluation.

Comprehensibility

The second category of statements investigates how easily users can understand content presented in the application.

Statement 1 (*It is easy to see the available functions*) and 2 (*The features of the application are hard to use*) asks the users if it is easy to see and use the features of the system. With an average score of 3.58 and 3.50, the results indicate that the majority of the users found the features easy to use.

We created specific client-side features we wanted to include, such as history and meeting notes browsing, which were displayed in the participant applications installed on the mobile devices. The overall user interface was created very simple, and one can argue that the responses, as presented from statement 1 and 2, are expected because of the simplicity of the interface and features. However, this was a conscious design decision because we did not want the application to be a hindrance due to complexity during the meetings. Both Hansmann et al. (2000) and Evans (2003) also highlight the importance of providing a simple design and architecture.

User Satisfaction

Finally, we wanted to investigate how the users perceived the system. With user satisfaction we look into the attitude of the users towards the application.

As previously explained, the proof-of-concept we created is tightly integrated with the Google App Engine, which provides the cloud platform as well as the advanced calendar and authentication functionality used in the system. In our questionnaire we presented three statements covering this topic.

Statement 8 (*The close integration with Google calendar is an inconvenience. I am not able to use the system without changing my existing or creating a new e-mail account at*

Google), 9 (*I do not like sharing my personal information to a service that stores the information in the cloud*), and 10 (*Storing meeting information in the Google cloud and combining this with personal information on the device is a useful feature*) asked participants to take a stand vis a vis the close integration with the cloud-based services, and whether or not they felt uneasy regarding the amount of personal information stored externally with a cloud provider. The results show that almost all evaluators agreed that they were comfortable with the information being stored in the cloud. This is somewhat surprising considering the potential privacy concerns that could arise in this respect, such as lack of transparency highlighted by Vigfusson and Chockler (2010). It is worth mentioning that in statement 9, one user responded that he/she did not like sharing the personal information, such as name and e-mail, in a cloud-based service. Nevertheless, the overall results indicate that the users generally did not have any concerns regarding information sharing in regards to cloud computing, with the results for statement 8, 9, and 10 providing an average score of 3.18, 2.93, and 3.38.

The next statements we wanted to look closer at focuses on the usefulness of different aspects of the system. Statement 11 (*I find the ability to navigate notes during the presentation useful*), asks about the ability to navigate in the received notes during a meeting. The next statement, 12 (*I do not find the meeting note history functionality useful*), looks closer at the history functionality of the notes stored from previous meetings. And finally, statement 15 (*I find the context-aware meeting room application useful*) asks about the perceived usefulness of the system. The average scores for these statements were 3.55, 3.25, and 3.33. Based on these results, we see that the evaluators did find the notes and history functionality useful and also it is especially encouraging that they saw the potential in the overall context-aware meeting room system.

Furthermore, statement 14 (*I would like to receive more content during the presentation*) asked the users about the idea of adding more content, such as files, images, and videos, during the presentations other than just meeting notes. Unsurprisingly, the majority of the users saw it as a useful feature to include more content, with an average result of 3.53. For future development of the system, the idea of adding new content is certainly interesting and our result also indicates that it is a feature that the users would find useful.

5.1.7.1 Limitations

The experiment has a few limitations that are important to be aware of. The performance of the network communication in our user tests was not only affected by our implementation, but also the fact that we were running the system on a wireless network used by other students simultaneously. As shown in the results of statement 4 and 5, where we experienced latency-related issues, this did to a certain degree influence the overall performance of the test. However, we still believe it is an important aspect to test the system in a realistic environment, where a WLAN is usually shared for many different tasks.

Additionally, a potential issue was the close integration with the Google cloud and Google services, such as Calendar and Docs. If these services were to go down, the system could not function correctly. Even though we cannot rule out the possibility of a service disruption with these services provided by Google[26], we find this scenario unlikely. We also acknowledge that with a considerably higher number of evaluation participants, the results gathered from the user evaluation might be different. However, we believe our test with 40 participants gives a decent and valuable insight into the world of mobile context-aware systems.

Having presented the experiment in the topic of *Context-aware Applications*, we now move on the *Smartphone Sensors* experiment before presenting the discussion and conclusion for the overall research area of context-awareness.

5.2 Experiment 3 – PainDroid: an Android Application for Pain Management

In the experiment presented in this section we wanted to investigate the use of sensors, and specifically target the Android platform. The Android platform was selected because of its openness and popularity, as reported by Gartner (2012). As already presented, sensors provide a crucial component of context-awareness, and we also see

[26] http://code.google.com/appengine/sla.html

the benefit of hardware availability and capability of the smartphones, which are now usually equipped with various sensors.

To show an example of the different sensors integrated in an Android device, we created an application that generates a list of available sensors. Figure 5-6 shows an application is running on the HTC Evo device, which was used in several of our experiments.

Figure 5-6, list of available sensors on HTC Evo.

As can be seen, the smartphone has a total of eight sensors installed. These sensors are available to developers through an API (Application Programming Interface) and make it possible to sense different aspects from the environment. We believe the area of smartphone sensors is an important topic to investigate further.

For this topic we identified the following objective:

Obj. 3-1 To improve and expand the way users with dexterity problems can interact with smartphone (and tablet) applications.

5.2.1 Literature Synopsis

According to Loke (2006) context-awareness in computing is a combination of hardware and software, which is able to sense the environment and automatically adapt and respond. Context-awareness and smartphone sensors are integrated in the sense that the sensors can provide context data that an application can use, i.e. it enables context-awareness.

One example of commonly used sensors is within cars. Sensors are installed to detect many different parameters, like rainfall and pressure in the tires (Hansmann et al. 2000). The information gathered by these sensors is sent to different processing units in the car, where an action will be triggered if needed, such as enabling the windshield wipers.

Relevant research efforts in the area of sensors include the work done by Gellersen et al. (2002), who investigated how to use a multi-sensor setup to achieve context-awareness in mobile devices and smart artefacts. Their experiments included three projects; 1) development of an awareness module for augmentation of a mobile phone, 2) *Mediacup*, exemplifying context-enabled everyday artefacts, and 3) the *Smart-Its* platform for aware mobile devices, ad hoc sharing, and collective awareness across connected devices.

In our experiment we use normal smartphone devices. This is similar to the work presented by Bhorasker et al. (2012), who created *Wolverine*, a non-intrusive road condition estimation system that can identify braking events, which uses input from smartphones to specifically collected data from the accelerometer, GPS, and magnetometer.

Similarly, Kwapisz et al. (2011) presents another interesting research effort that uses standard smartphone sensors to register events. They created a system that used the accelerometer of smartphones to perform activity recognition. This was implemented to identify physical the type of physical activity, such as sitting, walking, and running, a user is performing.

One novel aspect of our research is that we provided a very practical approach to challenges and opportunities using the on-device sensors, which we have not seen in any previous research efforts. We also focused on using smartphones and not requiring additional equipment, which is a reoccurring theme in this thesis.

5.2.2 Scenario

In the investigation of the on-device sensors and how to utilise them on the Android platform, we conducted the work as part of an experiment called *PainDroid*. PainDroid is a proof-of-concept application for Android, which provides a multimodal virtual reality solution for pain management. The application was created to enable patients to describe the pain experience in a more realistic and interactive manner from the comfort of his/her home. It included features such as the use of sensors as a control mechanism, a 3D visualisation of the human body, and support for VR technology through an HMD (Head Mounted Display). Figure 5-7 shows a user testing the PainDroid application.

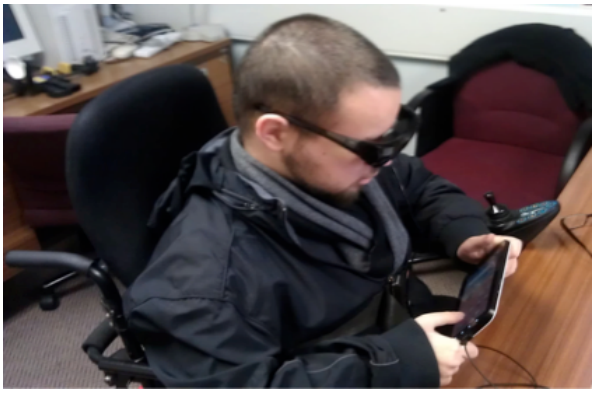


Figure 5-7, using the PainDroid application.

The application was based on user input from both touch and movement. The touch support included features such as *pinch-to-zoom, drag-to-move,* and *touch-to-select*. For the movement features we relied on sensors. When the user tilted the device up, down, left, or right, the accelerometer registered these changes and altered the interface accordingly. The movement of the device left and right, where one would turn the device, was registered using the magnetometer. Both options are presented in Figure 5-8.

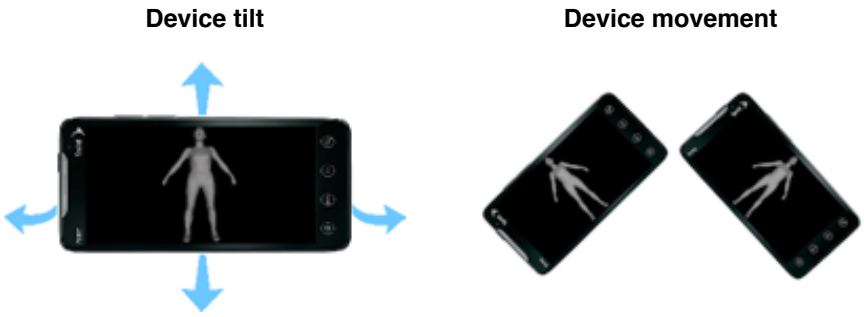


Figure 5-8, interaction with PainDroid using hand movement.

By using these simple hand movements, either tilting or moving/turning the device, we were able to rotate the mannequin shown on the screen (Figure 5-9). The use of these features in the application made it possible for users rotate the mannequin and then select the part of the body where they were experiencing pain. Additionally, we also included multiple Android devices in the experiment, which was a valuable aspect providing insight into how the sensor functionality behaves across various device models.

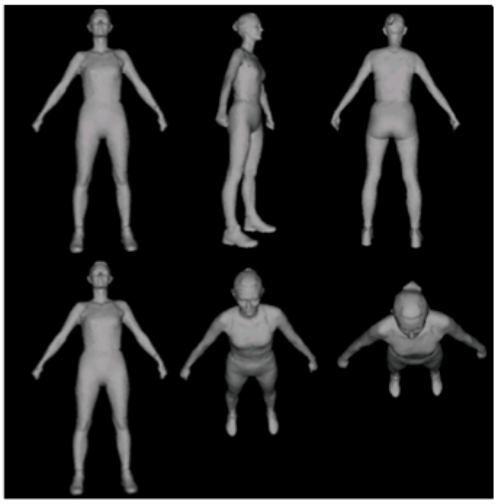


Figure 5-9, rotate mannequin.

It is important to note that in the context of this thesis, we are looking at the experiences gained during the PainDroid experiment. The focus is on the use of sensors on the Android platform and how these can provide context-awareness. We chose to use this experiment to present the topic of smartphone sensors because it used multiple sensors on the devices to provide input to the application and were tested on a variety of device

types. We continue by presenting the design and architecture of the system in the next section.

5.2.3 Design and Architecture

The system took advantage of sensor input and was implemented with a minimum requirement of Android 2.2. We included this requirement because we wanted to support tablet-layouts. The system components are presented next, before the system architecture that provide a more detailed view of the implemented proof-of-concept.

5.2.3.1 System Components

The system consisted of an application running on Android smartphones and tablets. It was created with three main components (Figure 5-10): 1) start-up utility classes (*white boxes*), 2) the views (*grey boxes*), and 3) the orientation listener (*black boxes*).

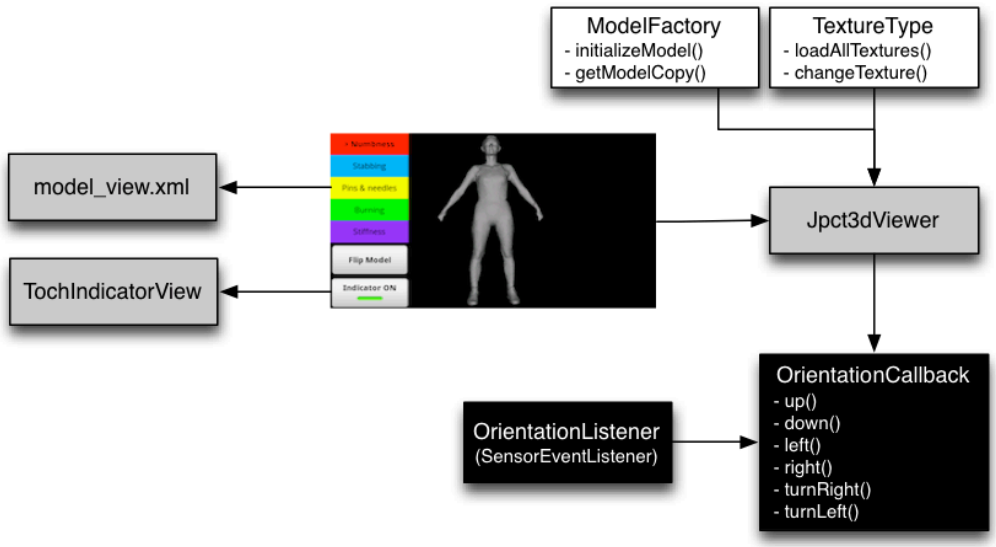


Figure 5-10, PainDroid system overview.

We used an open source library called jPCT[27] to provide the 3D features. When the application was initialised on the mobile device, the *ModelFactory* and *TextureType* classes would load the 3DS-model and textures into memory. By including these fairly heavy processes in these utility classes, we could cache the model in memory and only

[27] http://www.jpct.net/jpct-ae/

return a copy when the model was going to be presented on the screen. By doing this we chose to use a bit more memory by having a copy of the stored in memory, and thereby improving performance and reducing loading time in the application. This aspect was especially apparent when the user wanted to reload the model by restarting or resetting the main screen.

The main screen, presented in Figure 5-11, shows the 3D model in addition to the pain types presented on the left hand side of the screen. We also added the options to *flip model* and turn on/off the *indicator*. The *flip model* functionality was created as a simple and fast way to turn the model around, useful if one wanted to indicate pain in the back for example. The *indicator* was implemented to be used with the HMD glasses, this created a small circle on the screen that was used to indicate where the user had touched.

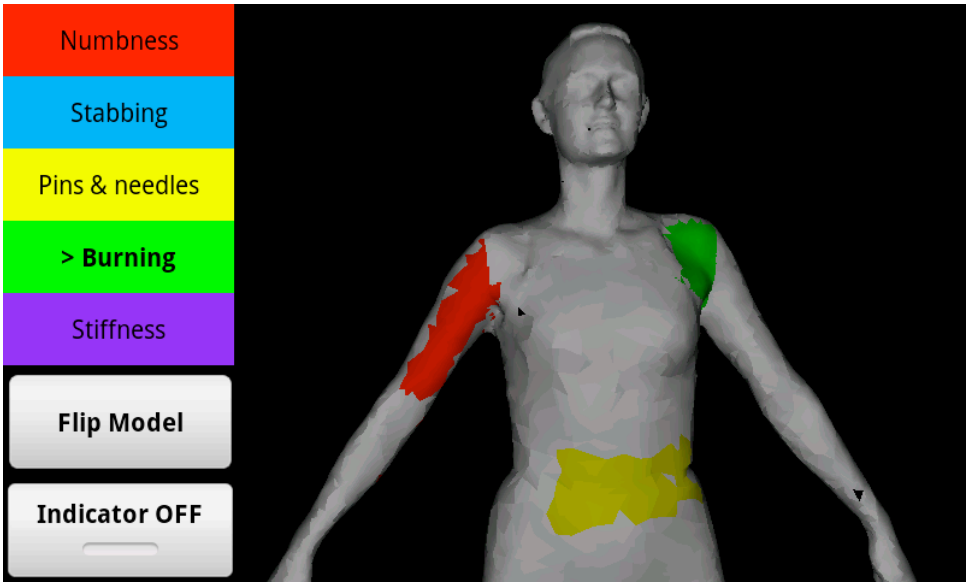


Figure 5-11, PainDroid screenshot.

The views consisted of a class called *Jpct3dViewer*, which presented the 3D model on the screen. Additionally, it was also responsible for initialising the touch detectors for all touch events in our system, which was *pinch-to-zoom, drag-to-move,* and *touch-to-select*. We implemented this feature by extending *GestureDetector*[28] available in the Android platform, making it possible to provide these touch events with only a few lines of code (See Appendix E for source code). The options menu was also created in

[28] http://developer.android.com/reference/android/view/GestureDetector.html

this view, taking care of the increase/decrease the rotation speed, stopping the rotation, recentering, resetting, and saving.

The buttons shown on the left-hand side of the screen were created by a layout-file called *model_view.xml*. It initialised the attributes of the buttons, such as the width and height. These were responsible for changing the colours shown when selecting a part on the 3D model.

Finally, the last view is *TouchIndicatorView*, which was responsible for displaying a small white circle at the location where the user touched the screen. We provided a toggle button for this feature, making it easy to turn on or off. The touch indicator was created because we offered VR functionality with the HMD glasses. When using these glasses the small white circle was provided as an indication of where the touch event took place, making navigation within the application considerably easier.

The last components are the orientation listeners, which were responsible for reading the input from the sensors and turning the values into actions, i.e. rotating the 3D model. These classes are important because they handled the sensor integration. In the next section, we go into more detail on these specific classes and their architecture.

5.2.3.2 System Architecture

As it is an important part of this experiment, we wanted to provide implementation details on how to use sensors on the Android platform. The application consisted of a set of classes and interfaces presented in Figure 5-12. The *SensorManager* class is provided by the Android platform, the other classes were created as part of our proof-of-concept project.

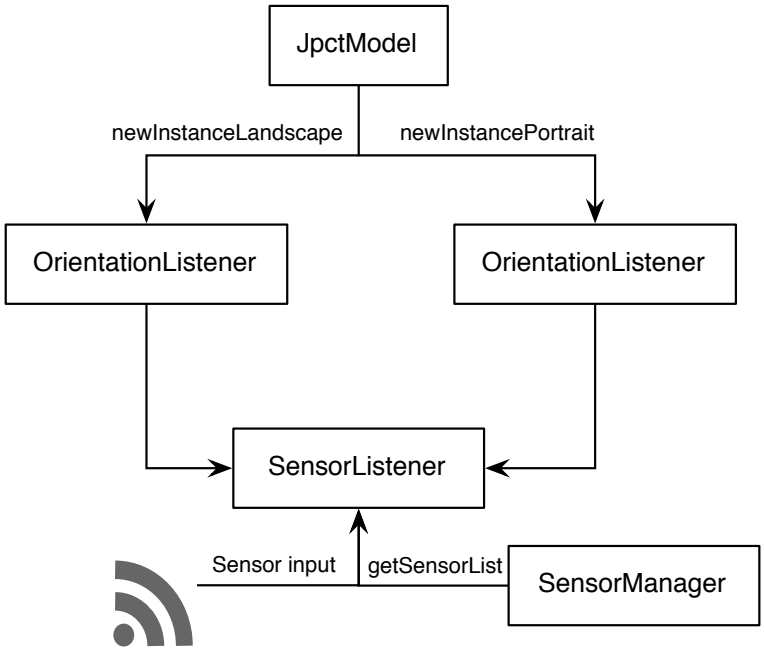


Figure 5-12, sensor components overview.

The most important classes of the system are the orientation listeners, because they were responsible for handling input from the sensors. The classes initialise the position when the application started. This means the direction the user held the device on start-up was recorded as the initial value and all movement was calculated based on this value. The code for these features was implemented in *OrientationListener* and *OrientationCallback*. *OrientationListener* was the interface, making it easy to register a custom listener. The *OrientationCallback* was responsible for handling callbacks from the sensor APIs in Android. The application used two sensors to provide input, namely the *accelerometer* and *magnetometer*.

In this section we include a few code examples to provide background information on how to utilise sensor data on the Android platform. The first issue we wanted to highlight was how Android provides access to the on-device sensors. In our experiment, we initialised the sensors in the start-method, shown in Figure 5-13. The method retrieved a list of sensors containing accelerometer and magnetometer. We continued by making sure the mobile device supported these sensors by checking that the list contained at least one item. The *SensorManager* class in Android makes it possible to access the device sensors and by registering a *SensorEventListener*, one can use the *onSensorChanged* and *onAccuracyChanged* to receive sensor data.

```
public void start() {
        sensorManager =
                (SensorManager) context.getSystemService(Context.SENSOR_SERVICE);
        List<Sensor> acclSensor =
                sensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);
        List<Sensor> magnSensor =
                sensorManager.getSensorList(Sensor.TYPE_MAGNETIC_FIELD);

        if (acclSensor.size() >= 1 && magnSensor.size() >= 1) {
            listener = new SensorListener(callback, threshold);
            sensorManager.registerListener(
                    listener, acclSensor.get(0), SensorManager.SENSOR_DELAY_NORMAL);
            sensorManager.registerListener(
                    listener, magnSensor.get(0), SensorManager.SENSOR_DELAY_NORMAL);
        } else
            throw new IllegalStateException(
                    "Unable to register ACCELEROMETER or MAGNETIC_FIELD");
}
```

Figure 5-13, initialising sensors.

One specific issue we noticed with the sensor implementation was that the data received for the direction of movement from the accelerometer, provided different results based on the device type. We were running our application in landscape mode, and for all devices, except the *Samsung Galaxy Tab 10.1*, the direction of the results did not change when we altered the standard display mode to either portrait or landscape. However, for the Samsung tablet this aspect had to be handled by our implementation. We added two different *OrientationListeners*, one for portrait and one for landscape, to fix this issue and to support sensor readings for all the devices using the same application. Figure 5-14 shows how we supported this by adding static factory-methods for the two display modes.

Having presented the most important architectural elements of the designed proof-of-concept, we move on the user evaluation conducted as part of this experiment, and the material used in this evaluation.

```
private void createOrientationCallback() {
        orientationCallback = new OrientationCallback3dModel(cameraRotation);

        if (landscape)
            listener = OrientationListener.newInstanceLandscape(
                    activity.getApplicationContext(), orientationCallback);
        else
            listener = OrientationListener.newInstancePortrait(
                    activity.getApplicationContext(), orientationCallback);

        listener.start();
    }
```

Figure 5-14, landscape and portrait OrientationListener.

5.2.4 Material

An important aspect of this experiment was to use various devices and test the sensor support with one common application. We started by using a smartphone, the *HTC Evo 4G*. This is the smallest device we used in the experiment, with a screen size of 4.3 inches and used Android. The next devices, *HTC Flyer* and *Samsung Galaxy Tab 7*, are small tablets equipped with a 7-inch screen. The largest device we tested with was the *Samsung Galaxy Tab 10.1*, which provides a 10.1-inch screen. See Appendix A for more technical details on the devices.

The selected devices all supported Android and included the on-device sensors. Another important feature was that they represented various device types, from the normal smartphone (HTC Evo) to a full-size tablet (Samsung Galaxy 10.1). Moreover, the Samsung Galaxy Tab 7 was included because it was well integrated with the HMD glasses, the *Vuzix Wrap 920*, used to provide VR capabilities.

5.2.5 Participants

The user evaluation for the PainDroid experiment was performed with seven participants who were also wheelchair users and two clinicians, namely a General Practitioner (GP) and a Rheumatologist (RH).

We targeted wheelchair users for the evaluation because of their dynamic pain patterns and severe mobility limitation. The users were all 18 years and older, and had experienced some pain during the period of the study. Another important factor was that two of the users had manual dexterity problems, and the extra options available through a multimodal interface, as implemented with the on-device sensors, could potentially provide an especially useful feature in their case.

5.2.6 Procedure

We conducted the evaluation based on usability and acceptability. Whilst the former was examined via a questionnaire, the latter was explored through interviews with the two clinicians involved. During the evaluation, the users were asked to complete a set of tasks, which are shown in Table 5-7.

	User tasks
1	Start the app.
2	Rotate the model left and right.
3	Rotate the model up and down.
4	Recenter the model.
5	Zoom in and out on the model.
6	Drag the model to the sides and/or up and down.
7	Reset.
8	Use the pain types on the left-hand side of the screen to select pain type.
9	Select a pain location on model
10	Save and exit.

Table 5-7, user instructions.

On completion of these tasks, the users were asked to provide answers to a questionnaire consisting of 13 statements (Table 5-8). In this experiment we used an adopted SUS (System Usability Scale), which was originally created by Brooke (1996). The 13 statements asked users to indicated agreement/disagreement on a five-point Likert scale, ranging from *1- Strongly Disagree* to *5-Strongly Agree*. We included a roughly equal split between positive and negative statements in the questionnaire.

Number	Statement
1	I think that I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.
5	I found the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.
11	I liked using the interface of this system.
12	The information (e.g. menu) provided by the system was clear and helpful.
13	I felt it difficult to recover after making a mistake.

Table 5-8, PainDroid questionnaire.

This evaluation was conducted as our last experiment, and we selected a different Likert scale than with the earlier evaluations. Previously we wanted to have an even-numbered scale to avoid a users answering neutral to all statements. However, for this experiment we used a five-point scale to investigate if this did in fact cause any problems. As our results show, we did not experience any issues related to neutral answers.

In addition to the statements, we provided two open-ended questions asking the evaluators to indicate what they considered to be the best aspects of PainDroid, and to point out what they thought needed the most improvement. The seven user evaluations lasted approximately between 10-18 minutes each.

A separate evaluation was performed with the two involved clinicians, the GP and RH. In this part we demonstrated PainDroid by walking them through the various functionalities of the application. Following the demonstration, we interviewed the clinicians with regards to their perceptions about PainDroid's acceptance and practicality in their everyday practice. Each interview lasted approximately 20 minutes in duration and was recorded.

In regards to the usability challenges for mobile devices (Zhang & Adipat 2005), we focused on *multimodality* in this experiment.

5.2.7 Results

The overall result of the user evaluation is presented in Table 5-9. We calculated the average score for each question based on positive statements to make the numbers easier to compare directly We grouped the statements into usability categories, and these are presented next.

.

Number	Avg. score (out of 5)	Standard deviation
1	3.86	1.46
2	4.00	1.15
3	4.00	1
4	4.86	0.38
5	4.29	0.49
6	4.71	0.76
7	4.14	1.07
8	3.43	1.72
9	4.57	0.53
10	4.86	0.38
11	3.86	1.07
12	4.00	1
13	4.14	1.21

Table 5-9, user evaluation results.

Learnability

The first category investigates the ease-of-use of the system. Statement 2 (*I found the system unnecessarily complex*) and 7 (*I would imagine that most people would learn to use this system very quickly*) asked the evaluators about how easy it was to learn and use the system. With an average response of 4.00 and 4.14 out of 5, most evaluators positioned themselves positive towards the system and found it relatively easy to learn.

Furthermore, the next statements, 4 (*I think that I would need the support of a technical person to be able to use this system*), 7 (*I would imagine that most people would learn*

to use this system very quickly), and 10 (*I needed to learn a lot of things before I could get going with this system*) focuses on how much the users needed to learn before using the system. All statements indicated that the majority of users found the system fairly easy to learn and that they would not need the support of a technical person to use the system. The average scores received were 4.86, 4.14, and 4.86.

Effectiveness

For the effectiveness group we wanted to focus on the completeness and accuracy with which users achieve certain goals.

Statement 8 (*I found the system very cumbersome to use*) and 13 (*I felt it difficult to recover after making a mistake*) both wanted the users to position themselves on how effective they were able to use the system. While statement 13 deals with the ability to recover from a mistake, statement 8 asks if the evaluators thought the system was cumbersome to use in general.

The average result for statement 8 was 3.43. In the responses we had two users strongly agreeing, while the other five respondents either positioning themselves within disagree (3) or strongly disagree (2). This indicates that some users did indeed find the system cumbersome to use, and the received feedback during the evaluation showed that there were certain users that found it difficult to rotate the mannequin using the sensor-based controls. For statement 13 we recoded an average score of 4.14, which indicates that the users found it easy to recover after a mistake.

Comprehensibility

The next focus is on comprehensibility, which deals with how easily users can understand content presented in the application.

Statement 5 (*I found the various functions in this system were well integrated*) indicated that the users found that the different functionalities of PainDroid were well integrated, with an average score of 4.29. The next statement we focused on, number 6 (*I thought there was too much inconsistency in this system*), targeted the perceived inconsistencies in the system. The responses from the users, with an average of 4.71, indicated that they did not experience any considerable inconsistencies. Finally, statement 12 (*The*

information provided by the system was clear and helpful) also showed that the participants thought that the menu information provided was clear and helpful, providing an average score of 4.00. The overall results were very positive in regards to the comprehensibility of the system.

User Satisfaction

The final group of statements are in regards to user satisfaction, which evaluates the attitude of users toward using the application.

Statement 3 (*I thought the system was easy to use*) and 9 (*I felt very confident using the system*) provided results indicating that the users found the application easy to use and they were confident when using the system. Average scores of 4.00 and 4.57 were the results from the evaluation.

However, responses to statement 11 (*I liked using the interface of this system*) provided more varied results. The average score was 3.86, and we received different comments during the evaluation process. Some users really liked the interface, with the pain-types shown on the left and the 3D model displayed in the middle. However, other evaluators wanted a different 3D model and also a more polished menu system.

Finally, in statement 1 (*I think that I would like to use this system frequently*) we asked the users if they would like to use this system frequently. The results indicated, with an average of 3.86, that several users found the system very interesting and useful, while others were more critical about the idea of using sensors as an additional input control. Also, the different opinions on the user interface may have affected the score on this statement about frequent use of the system.

We believe one issue that were a concern for a few of the evaluators was the novel implementation and use of on-device sensors as input controls. This proved to be challenging for certain users to learn in the short amount of time the evaluation lasted. Further testing with more participants is needed to provide more conclusive results on this issue.

An interesting aspect we noticed during the evaluation was that there is not necessarily a direct link between problems with manual dexterity and the ability to use the sensor-based controls. We had a case where one of the users had movement difficulties with the right arm. Due to this disability, a specific issue with handwriting was mentioned. However, when using the system the user found it quite easy to navigate and also stated: *"...easy to use for people who have hard time writing by hand"*. Nevertheless, other users had problems getting used to the fact that tilting/moving the device was utilised as an input mechanism, and this was experienced even though these users had full upper body mobility.

In addition to the wheelchair users, we involved two clinicians in the evaluation. Their responses were generally positive towards the potential for use of the system in a clinical setting. Specifically, we received the following statements:
"...very useful in the consultation room for mapping pain...and compare over time to see the improvement..." (GP)

"...advantageous if you can keep in memory previous recordings and patients don't know that - then you can keep on seeing how pain has changed...that would be quite neat!" (RH)

"...useful, since the idea of sending pain information to the doctor, directly from the app, results in patients getting more out of the doctor's appointment due to saving time." (RH)

In regards to the sensor input for an additional control mechanism, the rheumatologist appreciated the multimodal interface of PainDroid. This was because patients with neurological illness and with limited hand mobility could use the sensor-based interface as an extra modality to interact with the application.

Another interesting factor was that both clinicians stated that the prototypical nature of PainDroid means that acceptance by the community could potentially encounter hurdles. Specifically, they stated that: *"…it is a very different way of looking at things…"* (RH), and *"...a lot of doctors are not used to adopting tools like this…"* (GP).

We acknowledge these statements, but we are also encouraged by the generally positive attitudes toward the system and its potential use.

5.2.8 Limitations

The proof-of-concept created for this experiment was created as a prototype, and was evaluated with a relatively small number of users. Nevertheless, we believe that our findings reveal encouraging signs of stakeholder acceptance and satisfaction with the developed system. Larger scale evaluations and assessment of clinical efficacy both form part of our future efforts.

For the investigation of the use of sensors, the application created for this experiment was limited to the use of two sensors, specifically magnetometer and accelerometer. This is certainly enough to get an understanding of the support for sensors on Android. However, we acknowledge that a larger study with more devices and sensor types can provide additional information. Also, the addition of several mobile platforms, such as iOS, would be useful.

The next section continues to the topic of overall topic of this chapter, namely context-awareness. We start by presenting the discussion, and then finally the conclusion based on the two experiments.

5.3 Discussion

The overall research area presented in this chapter is context-awareness. We provided two specific directions within this topic, focusing on the *Context-aware Meeting Room* and *PainDroid* experiments.

The first experiment we presented was the Context-aware Meeting Room, where we focused on the *mobile context* and *restrictive data entry methods*. Zhang and Adipat (2005) identified these usability challenges as important and unique to mobile devices. By automatically registering the meeting participants and pushing meeting notes to the mobile devices, we provided context-awareness. The context was the proximity to the

meeting room and the calendar integration. Both of these aspects also provide invisibility, which is one of the main challenges of pervasive systems Saha and Mukherjee (2003).

A novel aspect of the system, which provided the invisibility, was a feature we called *Bluetooth ping*. This feature took advantage of the capabilities of Bluetooth to automatically search for devices in close proximity. By searching for a specific Bluetooth service (the hands-free service), we were able to support hidden devices and therefore avoid the requirement for the devices to be constantly in visible mode. Both Android and iPhone have limitations where they cannot be configured to be permanently in visible mode, however, with this feature we were able to work around this issue.

The context-aware features of the system, both the calendar integration and proximity registration, automatically adjusted, pushed, and persisted information for the users. Therefore we were able to minimise the usual manual tasks of entering data on mobile devices (*restrictive data entry methods*).

The final aspect we would like to highlight with the Context-aware Meeting Room was the heterogeneous environment. As previously mentioned, the variety of mobile platforms (Gartner 2012) and the trend toward heterogeneous environments (Ryan & Gonsalves 2005), highlight the importance of this issue. To create a heterogeneous system we implemented client applications supporting four mobile platforms, namely Java ME, Windows Mobile, Android, and iPhone. Because the system consisted of four separate client applications, we stressed the fact that the cloud-based server was responsible for the common services such as combining context received from proximity information with the calendar appointments. This was done to provide up-to-date information on the current meeting participants and also send messages to registered devices that were in the proximity of the meeting room.

The second experiment within the research area of context-awareness was PainDroid, where we focused on the use of smartphone on-device sensors. Specifically, we took a closer look at the sensors on the Android platform and how they can be utilised. The

sensors on the smartphones and tablets were used to provide additional input to the system, which created a control mechanism for the application.

In regards to the usability challenges for mobile devices (Zhang & Adipat 2005), we focused on *multimodality* in this experiment. The application included multiple input mechanisms, with support for touch-to-select, drag-to-move, and pinch-to-zoom as the touch based controls. Additionally, we also used the sensors to provide an option for moving and rotating the 3D model using movement and tilting of the device. An interesting result from the evaluation was that there were quite varied opinions on the input mechanisms of the application. We believe this feature, with the sensor integration, proved difficult to use for testers with little experience in using mobile devices, but a more comprehensive evaluation is needed to provide any conclusive evidence of this.

Another important part of the PainDroid experiment was to test the application on a variety of different device. During the development and evaluation we tested our application on four device types, and it is worth noting that there might be cases where precautions have to be taken due to differences between device types, such as the landscape/portrait issue we encountered and explained previously in this chapter. Even though we encountered this specific issue, we found that the Android sensor APIs provide useful features that are easy to implement. According to Henning (2009), poor APIs lead directly to increased development cost. He also states that an API should be minimal and not impose un-due inconvenience on the caller. We believe that the sensor support in Android is well defined and easy to use for developers, thus meeting the requirements stated by Henning (2009).

5.4 Conclusion

In this chapter we have identified three main objectives for the research area of context-awareness. The first two were provided by the *Context-aware Meeting Room* experiment.

The first objective, 2-1 (*To handle time consuming meeting tasks automatically, such as meeting registration, information sharing, and meeting history*), was to provide context-awareness in the implemented system. Specifically, we wanted to automatically complete time-consuming tasks in a meeting room scenario. This included the ability to register meeting participants in proximity of the meeting room and to push meeting notes to the mobile devices.

There is a significant difference between our work and previous research, such as Ferris et al. (2010) and Mantoro and Johnson (2003), where we focused on using proximity as context instead of location. The system consisted of mobile client integrated with a cloud-based server application, which was implemented with a technology called Protocol Buffer. We created a new open source project, called protobuf-javame[29], to offer support for this technology on the Java ME platform.

The automatic registration of meeting participants was a specific context-awareness feature supported by the system. With the *Bluetooth ping* feature (see Appendix D for source code), we were able to search for registered participants in close proximity. Moreover, we also used calendar data as input to the system to ensure that only registered meeting participants had access. These features were created to be invisible to the users and provide context-awareness to the system. We found the Bluetooth ping feature to be an important contribution, and we can see the potential benefit for other context-aware systems to utilise this feature for providing proximity information. The results from the evaluation also show that the users found this feature useful.

In regards to the cloud computing integration, a surprising result from the user evaluation was that the majority of users in our evaluation were comfortable sharing personal information, such as name and e-mail, on a cloud-based server. There are well-known privacy concerns regarding cloud computing, for example the lack of transparency that is highlighted by (Vigfusson & Chockler 2010). We would like to further examine the user acceptance of cloud computing in experiment 4 (*Customised Android Home Screen*).

[29] http://code.google.com/p/protobuf-javame/

The next objective for this experiment, 2-2 (*To provide context-aware and useful information without the need for additional hardware in the meeting room*), focused on creating a heterogeneous system, supporting four mobile platforms. In contrast to previous research efforts that created smart spaces, such as Jaimes and Miyazaki (2005), and Parviainen et al. (2006)**,** we utilised standard laptops and smartphones, not requiring the installation of external sensors, cameras, or microphones. The system only required smartphone devices and a Bluetooth-enabled laptop inside the meeting room, which was responsible for registering the participants.

The general responses in the evaluation on the *Context-aware Meeting Room* experiment showed that the users found the system and features to be useful. The novel features of the system, such as the automatic proximity registration and pushing presentation notes, were also well received. One suggested improvement was to include support for more content, such as files, images, and videos, to be pushed to the mobile devices.

For the next experiment, we focused on smartphone sensors for the Android platform. The experiment conducted was PainDroid, which consisted of utilising on-device sensors to provide an additional input mechanism. The objective created, 3-1 (*To improve and expand the way users with dexterity problems can interact with smartphone (and tablet) applications*), focuses a smartphone application, and specifically the task of making it easier for users with dexterity problems to interact with the system. The implemented proof-of-concept took advantage of the accelerometer and magnetometer to control a 3D model. This model was used for selecting the areas of the body where pain was occurring.

In research closely related to our work, Bhorasker et al. (2012) and Kwapisz et al. (2011) utilised smartphone sensors. However, we focused on using the sensors as a control mechanism, whereas they used it as a passive input source. Another novel aspect of our work is that we provide a very practical approach to the challenges and opportunities we encountered using the on-device sensors for Android, which we have not seen in any previous research.

One part of our experiment was to investigate the use of sensors on the Android platform. We have presented our general findings in this chapter, and shown that this feature is well integrated in the Android operating system. For future research this is a benefit that should be taken advantage of to introduce *context-awareness* into the applications. We also believe that work focusing on smartphone sensors in a context-aware setting is an interesting research opportunity within mobile environments.

From the user evaluation of the PainDroid application we received generally positive feedback, indicating that our work was perceived as useful. It is worth mentioning that a few users found the ability to control the application by using movement (sensor-based) a bit difficult, while others appreciated the additional features and flexibility it provided.

The users that were part of the usability evaluation of the system generally found the application and concept very interesting. Additionally, the two clinicians involved in the evaluation of the system responded positive. The rheumatologist appreciated the multimodal interface of PainDroid, because patients with neurological illness and with limited hand mobility could use the sensor-based interface as an extra modality to interact with the application. These findings may be of considerable interest to other researchers, and particularly healthcare providers.

Chapter 6 – Remote Information Access

This chapter introduces the topic of remote information access and in our experiment, *Customised Android Home Screen*, we look closer at using cloud computing in regards to context management for smartphones. This system closely integrates cloud computing with mobile devices, and provides a system that utilises context-aware information from various sources. In the next section, we start by introducing experiment 4.

6.1 Experiment 4 - Customised Android Home Screen

With our *Customised Android Home Screen* experiment we took advantage of context-awareness to provide additional features for the end-users. Specifically, we gathered context information from several context sources to build a rich foundation, on which the context-aware computations are based. Providing context-awareness in a mobile environment is an important factor because users bring their mobile phone or tablet just about everywhere, highlighting the need for adapting the content to the users current situation.

The features created for our proof-of-concept are implemented in a system using normal smartphones closely integrated with a cloud-computing environment. The main purpose of the experiment is to combine the characteristics of the smartphones with a cloud-based service to create a new user experience and a novel way to invoke control over the mobile device. The main objective created for this experiment was:

Obj. 4-1 To find the overall best suited push messaging technology for integrating smartphone applications with cloud computing.

Obj. 4-2 To provide information that is relevant based on the current situation (filter content, cloud integration).

6.1.1 Literature Synopsis

To provide remote information access we used cloud computing. Cloud computing refers to the applications delivered as services over the Internet and the hardware and systems software in the datacenters providing these services. The idea is built around an economy of scale, where the ultimate goal is to provide more resources, better scalability and flexibility for less money (Binnig et al. 2009). Research within the area of cloud computing has previously focused on issues such as security (Mowbray & Pearson 2009) (Dillon et al. 2010) (Vigfusson & Chockler 2010) and performance (Binnig et al. 2009) (Alhamad et al. 2010).

Within cloud computing, Mei et al. (2008) point out four main research areas they find particularly interesting, namely *pluggable computing entities*, *data access transparency*, *adaptive behaviour of cloud applications*, and *automatic discovery of application quality*. Our work mainly relates to the topic of data access transparency, where clients transparently receive information from the cloud.

One of the main technical goals with the system created for this experiment was to make the interaction between the cloud and mobile devices as seamless as possible for the user. Moreover, in this chapter we are focusing on context management in a cloud environment. Specifically, we wanted to provide information that was relevant to the current situation, and this meant combining several different sources of context.

We propose a novel approach, where we combine context-aware information from several dimensions to provide input to the system. In research closely related to our work, Huebscher et al. (2006), combines context data of the same context type to provide higher quality results. We are, on the other hand, using multiple sources that provide context information, such as calendar data and user configuration. In addition, we introduce the use a cloud-based server, which is responsible for the context management.

To integrate the smartphone application with the cloud computing application, we wanted to use push messaging technology. There are several alternative methods to implement push messaging for the Android platform, and that is why we wanted to do

an in-depth investigation to find the best-suited technology for our main experiment. This is presented next, before we in section 6.1.3 describe the scenario for the *Customised Android Home Screen* experiment.

6.1.2 Cloud to Device Messaging

Cloud computing and *mobile applications and media tablets* are both on Gartners' (2011) top 10-list of strategic technologies for 2012. This shows the importance of these technologies. To provide the integration between smartphones and cloud computing services, there is a need for network communication.

There are three main categories of data-delivery mechanisms (Kamal 2008): 1) *Pull-based* (on-demand), 2) *Push-based* (publish-subscribe), and 3) *Hybrid*. The pull-based messaging model is where the user device or computing system pulls data from the service provider's system. According to Kamal (2008), the advantage of this model is that no unsolicited or irrelevant data is received at the device. This is because it uses an on-demand model, where the data is received only when the device asks for it. The disadvantages of this model include the difficulty of setting the correct frequency of requests. Setting the frequency too fast drains the battery and uses a considerable amount of bandwidth. However, configuring with a slower frequency can cause the data to become old. Another disadvantage is the amount of server resources used. When many mobile devices are constantly polling for updates, it can cause a considerable load on the system.

Due to the mentioned disadvantages with a poll-based solution, we wanted to use the push-based data-delivery mechanism. Therefore, we created the first objective (*To find the overall best suited push messaging technology for integrating smartphone applications with cloud computing*) as part of this experiment to investigate this issue closer.

In our experiment we focused on the integration of the cloud-based server, which published information to the registered mobile clients. The final goal, as stated in the objective, was to find the overall best suited technology to use in our *Customised Android Home Screen* experiment.

6.1.2.1 Push Messaging Research

The push-based messaging model is where the server pushes data from the computing system to a client. It is also referred to as publish-subscribe, where the data is pushed to clients that have subscribed to a service and is an asynchronous messaging model. Only data from subscribed sources will be received, which makes this paradigm loosely coupled (Pohja 2009). An advantage of the push-based model is that there is no need to set the polling-frequency. The server will publish new data when it arrives. Kamal (2008) states that one disadvantage with this model is the possibility of receiving unsolicited, irrelevant, or out-of-context data. While we acknowledge this possibility, the case remains that one will still have to subscribe to a service to receive information from it. In this section we will present a few relevant research efforts on the topic of push messaging.

There are several research efforts on the topic of push messaging and the integration of cloud computing with mobile applications. One example is the research effort by Paniagua et al. (2011), who created *Bakabs*, an application created for Android and iOS, which relies on the cloud to bring its functionality on the provisioning of load management services in cloud-based web applications. Another paper utilising push messaging has been written by Flores et al. (2011), who focus on the interoperability issues of cloud computing. They created a generic middleware framework called *Mobile Cloud Middleware* (MCM), which simplifies the use of process-intensive services from mobile phones. MCM uses an asynchronous notification feature to de-couple the client and server. This is used to address cases where the mobile client is expected to perform a time consuming task.

In our investigation, we provided a benchmarking test of push messaging technologies on the Android platform. Cloud computing benchmarking tests are available in previous efforts, with research such as Alhamad et al. (2010) and Calheiros et al. (2011), but in our case we tested the integration between cloud computing and smartphones. We also provide a different perspective in that we are specifically looking at the benefits and

drawbacks of push messaging on the Android platform, which we have not seen in any previous research efforts.

6.1.2.2 Benchmarking Test

We conducted a benchmarking test to compare the relevant push messaging technologies for Android. We considered the six main push messaging libraries, whereas SMS was not included in the test. This was due to the fact that we wanted to include devices that did not support SMS, such as the Samsung tablet, and we focused on the mobile and cloud integration.

The libraries we found particularly interesting were: *C2DM* (Cloud to Device Messaging), *Urban Airship*, *Xtify*, *XMPP* (Extensible Messaging and Presence Protocol), *MQTT* (Message Queue Telemetry Transport) and *Deacon*. We believe these technologies represent the most promising and useful push messaging libraries for Android, however, we cannot completely rule out the possibility of other interesting options we were not able to find. Of these alternatives, we chose to not include two specific libraries in our test, namely MQTT and Deacon. MQTT was not included because we wanted to investigate push-messaging technologies that can be easily integrated into the cloud, and specifically on the Google App Engine. MQTT is useful for connections that require a small code footprint and where network bandwidth is limited[30]. It does require a message broker hosted on a separate server, and we did not find an easy way to integrate this service with the App Engine.

The second technology not included was The Deacon Project[31]. It is an open source project providing push notifications to Java and Android applications. This project was the least mature technology of the options we considered, as it is currently in the beta stage. Additionally, the project was created for users wanting to run push notifications on their own server and support Android versions lower than 2.2[32], whereas C2DM requires at least Android 2.2. These requirements did not match what we wanted to investigate in this experiment, which included a close integration with a cloud-based

[30] http://mqtt.org/
[31] http://deacon.daverea.com/about/
[32] http://deacon.daverea.com/2010/04/welcome-to-the-deacon-project/

server application and devices running on at least the 2.2 version of Android. Therefore, we included the following four technologies in our benchmarking test:

XMPP

XMPP is created for real-time communication[33] and for streaming XML (Pohja 2009). The technology behind XMPP was created in 1998 and then refined in the Jabber open source community in 1999 and 2000, before it was formalised by IETF (The Internet Engineering Task Force) in 2002 and 2003. It is commonly used in Instant Messaging (IM) and is used by Google Talk, Jabber and other IM networks.

C2DM

The next technology selected was C2DM. The overall goal of this library was to make it easier for mobile applications to sync data with servers[34]. The message limit is set to 1024 bytes and developers are encouraged to send short messages, essentially notifying the mobile application that updated information can be retrieved from the server.

Urban Airship

Urban Airship is a commercial option with support for Android, Blackberry, and the iOS platform[35]. In addition to offering C2DM support, it also provides a proprietary alternative. Urban Airship includes a push messaging alternative called *Helium*, which supports Android 1.6 and onwards.

Xtify

Similar to Urban Airship, Xtify is also a commercial option supporting multiple platforms (Android, Blackberry, and iOS). It also supports C2DM, but in addition it provides a proprietary protocol built on top of XMPP, using their internal infrastructure[36].

It is important to note that even though two technologies in our benchmarking test are based on XMPP, we use XMPP directly when integrating with the Google App Engine.

[33] http://xmpp.org/about-xmpp/
[34] http://code.google.com/android/c2dm/
[35] http://urbanairship.com/products/push-notifications/
[36] http://developer.xtify.com/display/sdk/Xtify+Android+XMPP+Rich+Notification+Guide

When testing Xtify we use their API and infrastructure, which provides a distinct difference between the two alternatives.

Procedure

For running the benchmarking test, we created a client application running on the mobile device. This client application was responsible for calling all the different push messaging technologies in sequence and then record the time used. On the server side we implemented a Google App Engine application that sends messages when requested to do so from the client, it is also responsible for storing all the result data received from the mobile application.

Thus, the performance test followed these main steps:

1. The Android client registers with the server. This is done differently for each technology, for example C2DM will send a registration id to the device.
2. A timer is started on the client, followed by a message being sent to the server requesting a new push message.
3. The server application receives the message and immediately sends out a push message consisting of 450 bytes to the mobile device. This will happen for each technology type.
4. When the message is received, the Android client stops the timer and registers the result. This result is then sent back to a result-servlet that is part of the server application, which permanently stores the information.
5. Finally, the process waits 5 minutes before continuing with the next technology.

As part of the benchmarking test, we used three Android devices, namely *HTC Nexus One*, *HTC Evo 4G*, and *Samsung Galaxy Tab 10.1*.

Results

The goal of the benchmarking test was to compare push messaging technologies on the Android platform, and find the overall best-suited alternative for our experiment. We define performance in this context as:

- **Response times**, what are the response times for the different push messaging technologies?

- **Stability**, are the response times providing stable results over the time we run the test?
- **Energy consumption**, which technology provides the best energy efficiency?

Table 6-1 presents the overall results from the test. Using the results from the Samsung Galaxy tablet, the average response time for XMPP was the shortest, with C2DM on second and finally Urban Airship. There is a difference of 276.12ms between the fastest (XMPP) and slowest (Urban Airship) average response times.

Device	Tech	Number of messages	Average response time (ms)	Standard deviation
Samsung Galaxy Tablet 10.1 - *Android 3.1* - *WIFI*	C2DM	281	466.82	203.76
	Urban Airship	279	619.43	708.72
	XMPP	280	343.31	172.91
HTC Evo - *Android 2.3* - *WIFI*	C2DM	174	401.89	95.40
	Urban Airship	172	473.88	321.97
	XMPP	168	316.90	67.84
HTC Nexus One - *Android 2.3* - *3G*	C2DM	17	502.47	59.68
	Urban Airship	37	814.27	943.24
	XMPP	30	436.60	286.10
HTC Evo - *Android 2.3* - *WIFI*	Xtify	213	432.92	250.09

Table 6-1, benchmarking test results.

In Figure 6-1 we present the results gathered for the *Samsung Galaxy Tab 10.1*, and in Figure 6-2 are the results from the *HTC Evo*. Both devices ran on the same WIFI network and we were able to provide a fairly equal number of messages for each technology. As can be seen from the graph, the difference is mostly due to spikes in the response from Urban Airship. These spikes were more frequent on the results gathered from the Samsung Galaxy, however, they were also present during the other tests (for example with the HTC Evo device), which can also be confirmed when looking at the standard deviation. The max time used for Urban Airship was 5337ms (*Samsung Galaxy*) and 3601ms (*HTC Evo*), whereas both XMPP and C2DM provided considerably more stable results. XMPP had the most stable results in our test, with a standard deviation of 172.91 (*Samsung Galaxy Tab 10.1*) and 67.84 (*HTC Evo*). Urban

Airship did appear to have more stability issues than the rest and these issues surfaced several times during the test. When comparing the results from different WIFI networks and 3G, the same pattern emerges. The 3G response times are higher, but this is to be expected since they will have less bandwidth than the WIFI connection.

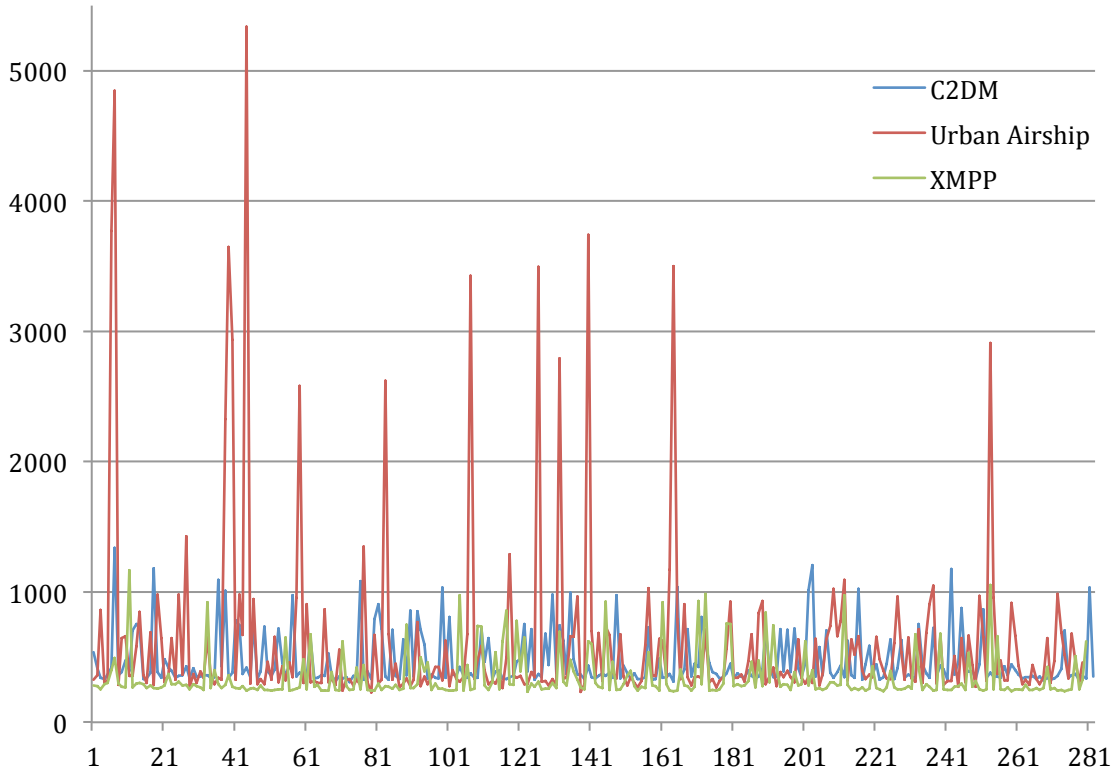


Figure 6-1, results for Samsung Galaxy Tab 10.1 (WIFI).

Overall the C2DM results were stable and the performance results recorded showed an average response time of 466.82ms on the Samsung Galaxy tablet over 281 messages. With the HTC Evo we recorded an average response time of 401.89ms over 174 messages.

The final technology we tested was Xtify, and it comes very close to the overall performance of C2DM with an average response time of 432.92ms on the HTC Evo (again, running on a WIFI network). Additionally, it also provides more stable results than Urban Airship. We were unable to record more messages with Xtify, because of limitations with our developer account, which is also stated in the *limitations* section.

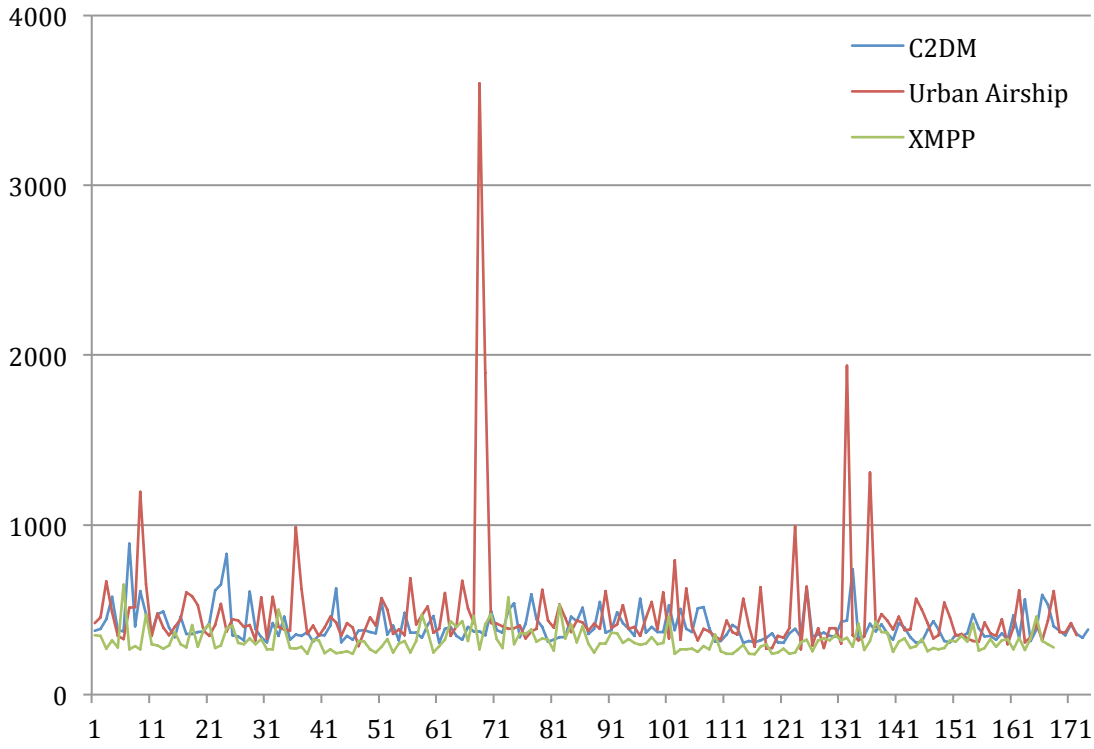


Figure 6-2, results for HTC Evo (WIFI).

For the final test, we wanted to investigate the energy consumption of the different push-messaging technologies. In this performance test, we ran the same application as before, but we increased the time between messages to 10 minutes. Another difference was that instead of running each technology in sequence, we only recorded one technology at a time. This was done because we wanted to provide results based on the battery level for each technology, and also to expand the test over longer periods of time. The client still ran the test by sending requests to the cloud-based server, but we added a feature that triggered a new message for each change in battery level. By doing this, we were able to record the messages and also the corresponding battery level for the device.

The test ran on the *Samsung Galaxy Tab 10.1* and this was done to get a more comprehensive test. This is because the Samsung Galaxy Tab 10.1 has a significantly larger battery (7000mAh) compared to for instance the HTC Evo (1500mAh). In this test we included C2DM, Urban Airship, and XMPP. Xtify was not included due to limitations with the development account we had created specifically for this experiment.

With both C2DM and Urban Airship we were able to provide fairly equal number of messages, with 862 and 858 messages sent respectively. However, with XMPP we were unable to send more than 295 messages because of quotas and limits in the Google App Engine[37]. The results are presented in Figure 6-3, and we have added trend lines for each result to make it easier to see the differences between the technologies.

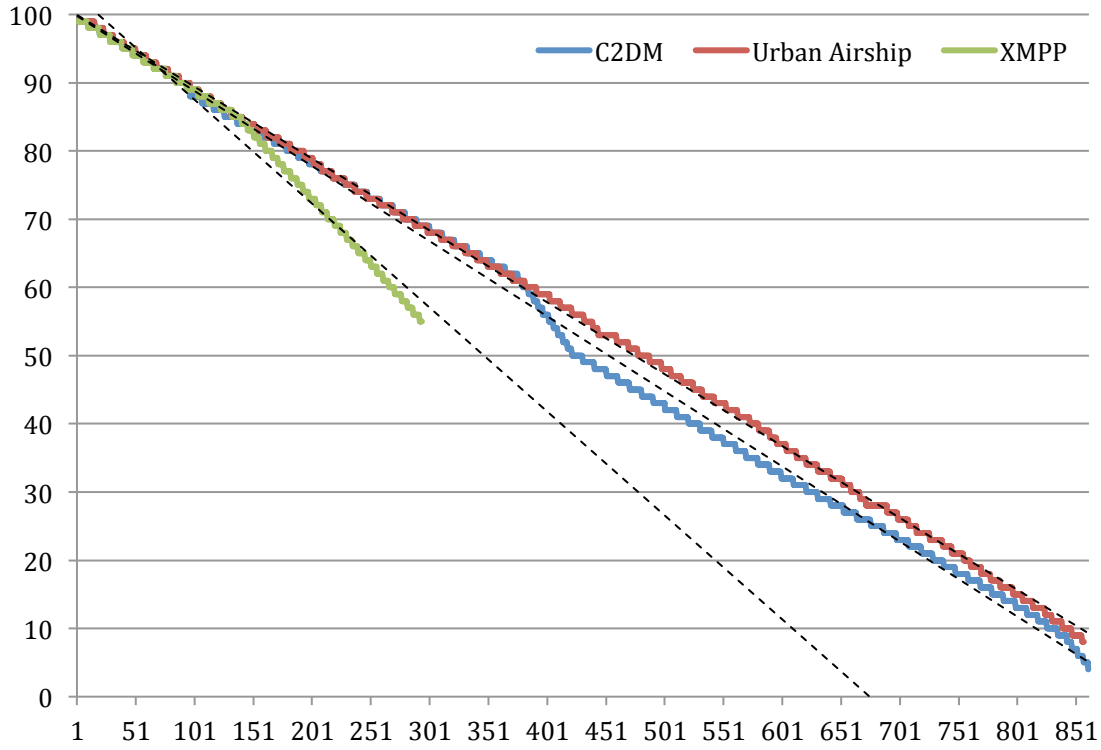


Figure 6-3, energy consumption test results.

Both C2DM and Urban Airship provided the best results, using less energy than XMPP. This was an expected result, and as stated by Xtify[38], the usage of XMPP is recommended in cases where one needs to communicate frequently over a short period of time. Both Urban Airship and C2DM provided fairly similar results, although from the last 50% and towards 0%, Urban Airship did provide slightly better results.

Limitations

Our benchmarking test had some limitations, and this was specifically apparent on some of the mobile devices where we had problems running all of the push messaging technologies over an extended period of time. This would result in certain messages not

[37] https://developers.google.com/appengine/docs/java/xmpp/overview#Quotas_and_Limits
[38] http://developer.xtify.com/display/sdk/Xtify+Android+XMPP+Rich+Notification+Guide

being received. The test would run reliably for period of time, before the messages were no longer received on the mobile client. A restart of the Android client would solve the problem, but this scenario happened multiple times. This is why some of the devices have very few test results for certain technologies, for instance the HTC Nexus had issues with C2DM. This only happened on the Android 2.3 mobile devices, whereas the Android 3.1 tablet was very stable over the entire test period. It would be interesting and useful for future work to focus on these stability issues, by including Android version 4 devices in the tests to see if the problems are fixed or at least improved in this newer version of the operating system. However, at the time of writing, an official version of Android 4 is not released for the devices used in the benchmark test.

Additionally, both Urban Airship and Xtify were tested on development servers. In the case of Xtify, we had to run this separately because there is a limit of 300 messages sent when using a basic account. The test consisted of a total of 213 messages, because we had to setup and test the system before running the actual experiment.

6.1.2.3 Summary

The results show that C2DM performance was good compared to the alternatives, and it also performed well in the energy consumption test we completed, where it proved that it is an energy efficient technology when compared to other similar technologies, and especially compared to XMPP. Because C2DM provided the best overall results, we wanted to use this technology in our experiment.

We also experienced a few issues with the API of C2DM, which we wanted to improve. These were specifically issues related to the lack of flexibility and certain useful features. A more in-depth presentation of this is available in Appendix F. We move on with the description of the Customised Android Home Screen application, and present the scenario.

6.1.3 Customised Android Home Screen Scenario

The proof-of-concept created for the main experiment provided users with the ability to configure their Android device through the use of a cloud-based service. The mobile device collected context data from several sources and cooperated with the cloud server to provide useful features, such as a context-aware calendar and contacts list. The features included the ability to filter contacts based on the upcoming meeting.

The process of using the system started with the user logging into a webpage, which was part of the cloud application. The webpage contained all the configuration options available to the users, as shown in Figure 6-4.

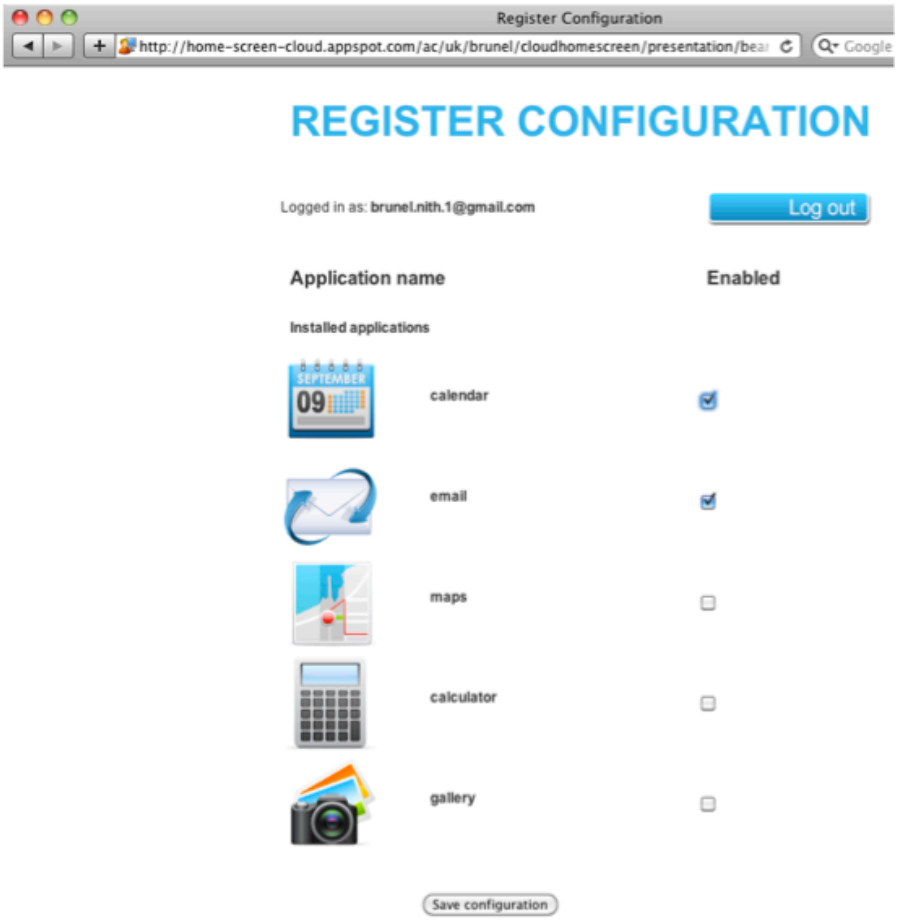


Figure 6-4, configuration webpage.

This configuration controlled the home screen on the smartphone, which means each icon *enabled* on the website was also automatically displayed on the home screen of the

device (Figure 6-5). When a user wanted to store the configuration, he/she would simply press the *Save configuration*-button. Within seconds the device is updated to reflect the configuration selected on the webpage. A push message was sent to the device in the background, and no user action was needed after the save button was pressed.

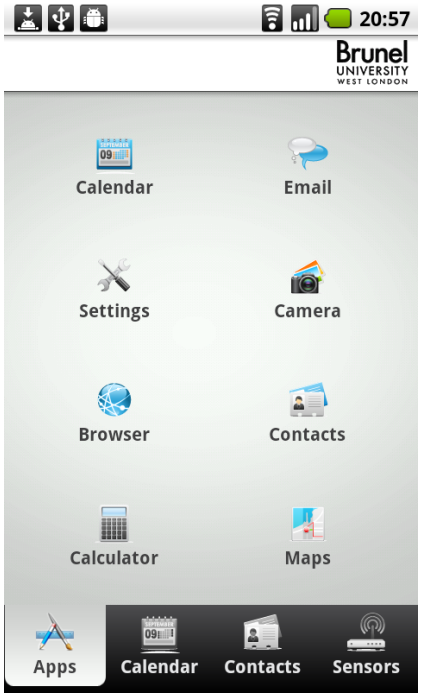


Figure 6-5, custom home screen.

We see two main benefits with this solution. Firstly, the user configuration was based on the Google account and not bound to the mobile device, since the *"master configuration"* was persisted on the cloud-based datastore. When the user wanted to upgrade to a new smartphone, the configuration was automatically installed at initial login without the need to re-configure all the previously saved options. Secondly, the configuration was done through a webpage and pushed to the mobile device. This gives us the option to add more advanced configuration options. For most users it is easier to configure the options using a normal computer, because one has the benefit of using a normal keyboard and mouse in addition to a much larger screen.

6.1.4 Design and Architecture

We provided use cases for the main requirements, which are shown below in Table 6-2, Table 6-3, and Table 6-4. It is important to note that these use cases are not all the requirements we created for the system, but the most important ones we selected in the context of this chapter.

Use case	Present contact information based on the current context received from the calendar integration.
Primary actor	Mobile device user
Preconditions	The user has an upcoming meeting, where he/she has tagged the event as *work* or *leisure*.
Basic flow	1. The user selects the *Calendar* tab on the mobile clients, and is presented with a list of upcoming meetings. The next meeting is tagged as *work*. 2. In the next step, the user selects the *Contacts* tab and is presented with only work-related contacts.
Post conditions	The user is presented with a list of contacts that are filtered based on the type of event, either *work* or *leisure*.

Table 6-2, Use Case 1 – present contact information based on context.

Use case	Show a simplified view when the user is moving.
Primary actor	Mobile device user
Preconditions	The client application active.
Basic flow	1. The user is carrying the mobile device when moving, for instance exercising. 2. The device registers the movement from the accelerometer, a *"shake"*-event. 3. When active, the screen on the device switches to a simplified view. This view presents all icons in a list, providing bigger buttons that make it easier to use when on the move.
Post conditions	The simplified view is shown.

Table 6-3, Use Case 2 – show the simplified view.

Use case	Send a push message to the mobile device containing the updated home screen configuration.
Primary actor	Cloud-based server application
Preconditions	The user has initialised the system, meaning that a registration id has been stored on the server (this happens automatically on application start-up).
Basic flow	1. The user is logged in to the webpage and changes the configuration. The *Save configuration*-button is pushed. 2. The server reads the new configuration values and persists them on the cloud-based datastore. 3. Next, the server locates the registration id based on the e-mail of the logged-in user and constructs a message in the C2DM format. 4. The message is sent to the located registration id.
Post conditions	The message is received by the mobile application, and the home screen is automatically updated based on the new configuration.

Table 6-4, Use Case 3 – send a push message.

6.1.4.1 System Components

The system consisted of three main components: 1) *Android client*, 2) *external Google services,* and 3) *a cloud-based server*.

The Android client was the main component. It communicated with both the cloud-based server, installed on the Google App Engine, and the external Google services. These services provided data for calendar and contacts information, where the client directly communicated with them through an API provided by Google. We decided to provide the responsibility of communicating with these services to the Android client instead of going the route through the server. The main reason behind this decision is that we did not believe the extra cost of the additional network call was justified in this case. The interaction is very simple, the application just asks for data for a specific user and there is no significant business logic involved in this step. The client would simply query the calendar and contact API and then use xml parsers to extract the content.

To provide this context information on the contacts and calendar appointments, we created special meta-tags. By adding a type tag, such as *${type=work}* or

${type=leisure}*, we were able to identify the type of the appointment, in this case either a business meeting or a leisure activity. The system was then able to filter the contacts based on this information, and provide the context-awareness feature based on the next upcoming appointment. If the tag *${type=work} was added, the system knew that the user is in a work setting and would automatically adapt the contacts based on this input. In a leisure context only, no work contacts would be shown, as presented in Figure 6-6. To add and edit these tags we used the web-interface of Google contacts and calendar. Both calendar appointments and contacts were tagged with this information.

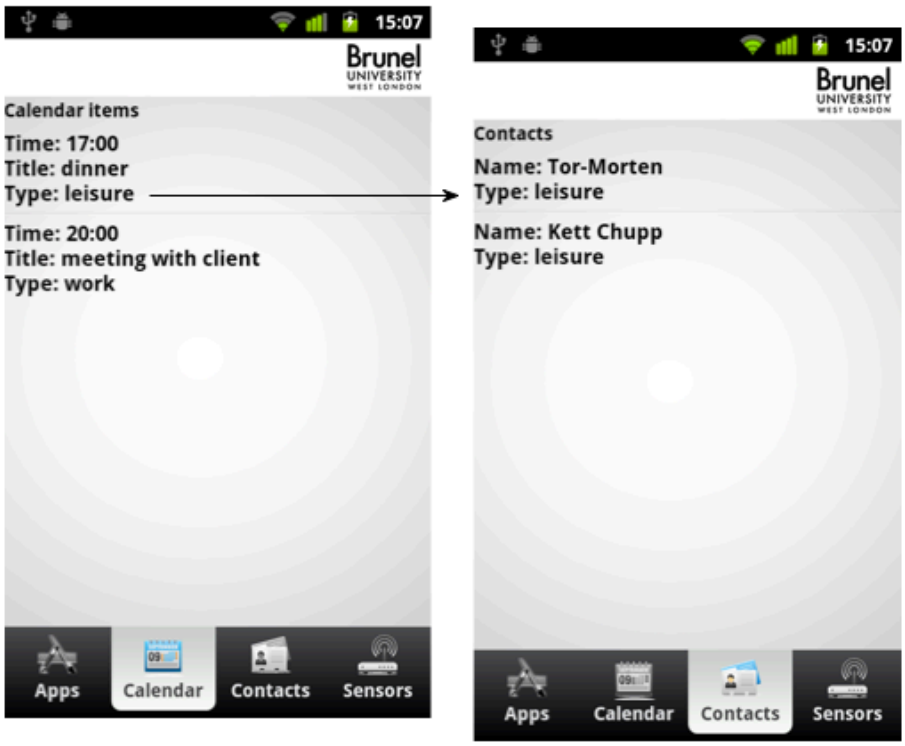


Figure 6-6, calendar and contacts information.

Another main feature of the application, as presented by use case in Table 6-3, was to change the GUI (Graphical User Interface) to a simplified mode if we registered that the user was moving. One of the potential scenarios for this feature was when the user was for example exercising and wanted to use the mobile device while moving. In the previous chapter we presented how it is possible to utilise on-device sensors on the Android platform, and we took advantage of this feature to register *"shaking"* events that provided an indication of movement. When these events were registered we change to a simplified mode, as shown in Figure 6-7. In this mode, the icons were presented as a list, which provided considerably larger buttons that were easier to select.

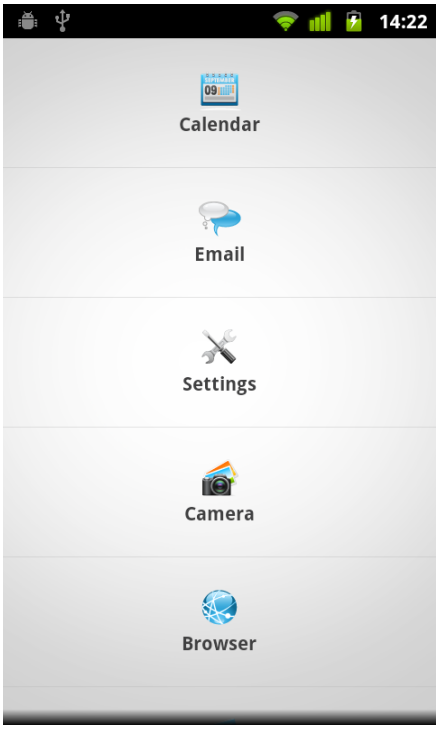


Figure 6-7, the simplified interface.

The final component was the cloud-based server. It was responsible for handling the user configuration, where each user could configure the home screen icons. The server was built using push messages to send updated configuration values to the mobile devices from the cloud server. For this purpose we used C2DM (Cloud to Device Messaging), which provided the best overall performance as shown previously. Moreover, this technology has some very appealing benefits, namely messages can be received by the device even if the application is not running, it saves battery life by avoiding a custom polling mechanism, and it takes advantage of the Google authentication process to provide security.

6.1.4.2 System Architecture

The system provided services for the application running on the mobile devices and Figure 6-8 presents an overview of the implemented system. The blue boxes represent the parts of the system we implemented for the proof-of-concept, while the white boxes are external systems, such as Google calendar and contacts.

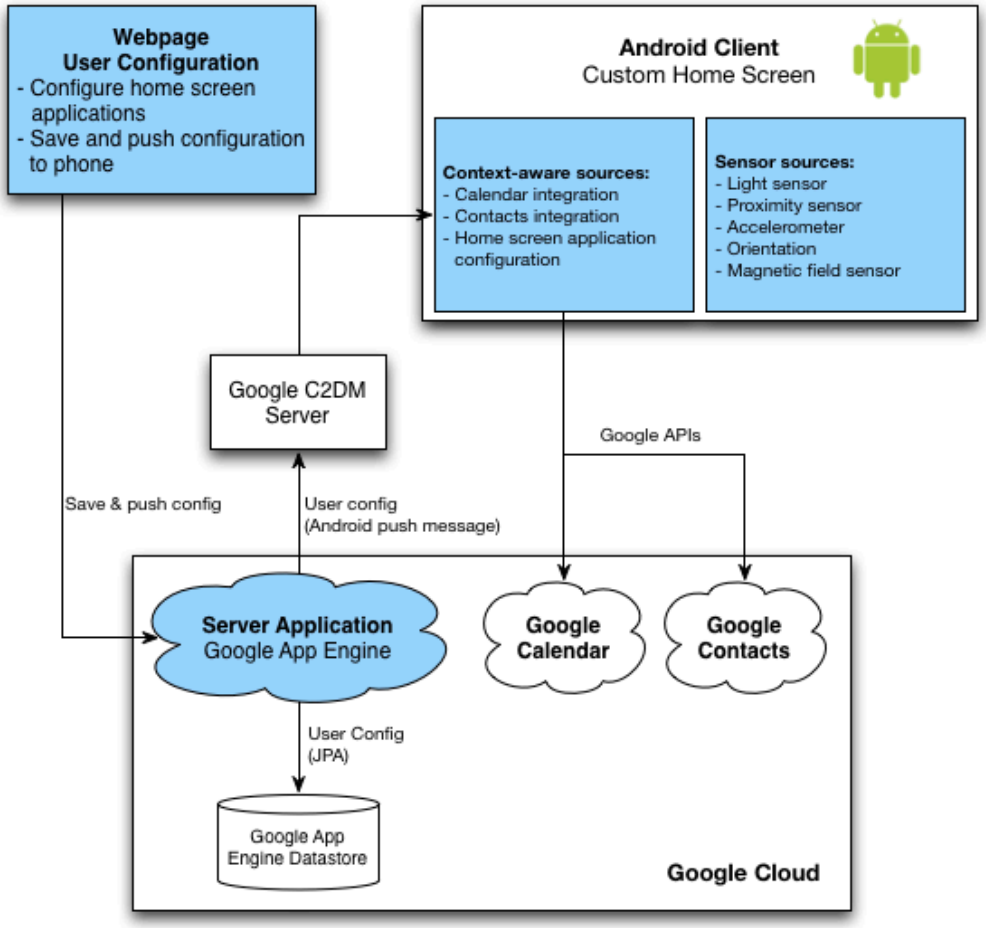


Figure 6-8, Customised Android Home Screen system overview.

One important advantage of providing the close integration with the Google App Engine in our system was the fact that we were able to utilise the Open Authentication service. This was implemented on both the cloud-based server application and the client installed on the mobile devices. With little effort from our side, it provided us with the possibility to offer a login process where the users were asked to enter their Google username and password. By leveraging the possibilities of Open Authentication, we facilitated users sharing their private calendar appointments and contacts stored in their Google cloud account without having to locally persist their credentials. Specifically, this service provided us with a well-tested solution, and we did not have to create our own security mechanism.

For the integration between the mobile device and the cloud, we used the C2DM push-messaging technology that supports Android 2.2 devices and onwards.

In the following sections we move on to the evaluation of the system, starting with the material used before the participants, procedure, and results are presented.

6.1.5 Material

This section presents the most important devices and platforms used as part of the experiment, starting with the cloud computing platform and then the mobile devices.

6.1.5.1 Cloud Computing Platform

In our proof-of-concept we used the Google App Engine as our cloud-computing environment. Specifically, we integrated with Google calendar and contacts, and we also utilised the authentication service provided. The data, such as the registration id for each mobile device, was persisted in the Google App Engine datastore and accessed with JPA (Java Persistence API).

The choice of using the Google App Engine was made because we, similar to previous experiments, used other Google services. These services, such as Google calendar and contacts, are well-integrated with the Google App Engine.

6.1.5.2 Hardware Devices

The experiment focused on the Android platform. As already mentioned, we wanted to integrate with other Google services and Android provides a good foundation to accomplish this. Additionally, because we wanted to provide services customising the mobile device, i.e. to a certain extent replace and/or extend the standard home screen, we saw the benefit of using Android because of the openness of the platform. The devices that were used as part of the experiment were the *HTC Nexus One* and the *HTC Evo 4G* (more details on the hardware is presented in Appendix A). These were selected because they are both Android devices and support high-speed network connections.

6.1.6 Participants

We started with an initial evaluation consisting of 12 users, which conducted quality assurance and a preliminary test of the system. The results from this session where fed back into the development loop and produced a new iteration, where we fixed the bugs identified and worked on improving the features.

User Tasks	
1	Start Android application, you might need to enter username/password and give the application access to the Google account the first time.
2	Select calendar tab, and pay attention to the type of the first appointment
3	Select contacts tab. Make sure the contacts are filtered based on the type tag on the next meeting. If work is the type of the next meeting, only work related contacts should be shown.
4	Move to the sensor tab on the Android device. Select the different sensor options and look at how the values changes when you move/shake/expose to light etc.
5	On your computer start a browser and log in to webpage: http://home-screen-cloud.appspot.com - Select the application you want displayed on the home-screen on the Android device. - Press the *"Save configuration"* button. The configuration is sent as a push message directly to your phone when this button is pressed.
6	Make sure the home-screen on the Android device (Apps tab) is updated and the icons you selected on the webpage are shown.
7	Press one of the icons to launch the application and get back into the application afterwards
8	On the apps-screen of the Android client, try to expose the phone to more/less light.
9	Make sure the background changes colour based on the different light levels.
10	Still on the apps-screen, pick up the phone and shake it.
11	The UI should now change to a simple layout. This is meant for users on the move (for example running/walking) where they might want to use a simpler and more concise layout. After 15 seconds of keeping the device still, the usual layout is displayed
12	Log in to Google calendar in your browser: http://calendar.google.com
13	Experiment with the calendar and contacts integration by moving existing appointments, and make sure the contacts shown are updated.
14	Please answer the questionnaire: http://www.surveymonkey.com/s/androidHomeScreen

Table 6-5, user instructions.

The second round of testing involved a larger number of users, with a total of 38 participants. The evaluation users were of mixed age (between 20 and 55), gender, and computer expertise. Additionally, the evaluators were selected from both an academic and corporate background.

6.1.7 Procedure

In the main evaluation the users were given a mobile device running the application, and were asked to complete a set of tasks. These tasks are presented in Table 6-5. After the tasks were completed, we provided each user with a questionnaire consisting of 17 statements. The users were asked to provide answers to a roughly equal number of positive and negative statements. These statements had responses on a four-point Likert scale, ranging from *1-Strongly Disagree* to *4-Strongly Agree*. As with experiment 2, we selected a four-point scale because we wanted to avoid situations where users only answer neutral to all statements.

The statements in the questionnaire were grouped into the following categories: *User Interface*, *Sensor Integration*, *Web Application*, *Context-awareness*, and *Cloud Computing*.

We focused specifically on *mobile context*, *connectivity*, and *restrictive data entry methods*, which are challenges for usability in mobile devices (Zhang & Adipat 2005).

The statements are presented next (Table 6-6), before we go into more detail on the results.

Number	Statement
	User Interface
1	It is easy to see the available functions.
2	The features of the application are hard to use.
3	The adaptability of the application is a feature I approve.
	Sensor Integration
4	The background colour in the application changes when the lighting in the room changes.
5	When moving around, a simplified user interface is not presented.
6	I find the sensor integration annoying and would disable it on my device.
	Web Application
7	I was able to register my device application configuration in the web application.
8	I was not able to store and push my configuration to my device from the web page.
9	I would like to configure my phone from a cloud service on a daily basis (webpage, user configuration, and Google services such as mail/calendar/contacts).
	Context-awareness
10	The close integration with Google services is an inconvenience, and I am not able to use the system without changing my existing or creating a new e-mail account at Google.
11	Calendar appointments displayed matched my current user context.
12	The contacts displayed did not match my current user context.
13	I would like to see integration with other online services such as online editing tools (for example Google Docs) and user messaging applications (like Twitter).
	Cloud Computing
14	I do not mind Cloud server downtime
15	I do not like sharing my personal information (like my name and e-mail address) to a service that stores the information in the cloud.
16	Storing data in the Google cloud and combining this with personal information on the device is a useful feature.
17	I find the cloud-to-device application feature useful.

Table 6-6, Remote Information Access questionnaire.

6.1.8 Results

The results for each statement are presented in Table 6-7. It is important to remember that the average scores are based on positive questions, making the numbers easy to compare.

Number	Avg. score (out of 4)	Standard deviation
1	3.50	0.51
2	3.11	0.73
3	3.45	0.55
4	3.55	0.65
5	2.89	1.06
6	3.16	0.72
7	3.61	0.59
8	3.53	0.80
9	3.18	0.69
10	3.24	0.88
11	3.58	0.55
12	3.71	0.52
13	3.29	0.73
14	2.92	0.78
15	2.84	0.79
16	3.26	0.60
17	3.52	0.51

Table 6-7, user evaluation results.

The results are again categorised into groups based on important aspects of usability in software applications.

Effectiveness

The first presented statements are grouped into effectiveness, which deals with the completeness and accuracy with which users achieve certain goals.

The system supported a feature where the users could register their home screen settings on a webpage and then push the changes directly to the mobile device. Statement 7 (*I was able to register my device application configuration in the web application*) asked if the functionality worked correctly. Similarly, Statement 8 (*I was not able to store and*

push my configuration to my device from the web page) asked about the ability to transfer the configuration of the home screen to the mobile device. These statements received an average score of 3.61 and 3.53, indicating that the features worked correctly during the evaluation.

In addition to the configuration of the home screen, we provided sensor-integration in the system. Specifically, we supported changing the background colour of the device when the light in the room changes, statement 4 (*The background colour in the application changes when the lighting in the room changes*). Moreover, we also included the ability to switch to a simplified user interface when a "shake-event" was registered, statement 5 (*When moving around, a simplified user interface is not presented*). With an average score of 3.55 and 2.89, the general feedback from the users on the sensor integration was varied. This is further verified by statement 6 (*I find the sensor integration annoying and would disable it on my device*), which also provided different opinions among the evaluators. The results show an average score of 3.16.

We specifically noticed that the ability to register *"shake"*-events on the Android application did not always work consistently. Particularly, it was difficult for the application to distinguish shaking resulting from putting the phone down on the table or handing it over to another person, from walking or running with the phone. This is a feature we would like to improve in future versions of the application, to make it more reliable. Additionally, it would be a good idea to give the user the option to easily turn the feature off.

The system also provided context-awareness features, with calendar and contacts integration, which were highlighted in statement 11 (*Calendar appointments displayed matched my current user context*) and 12 (*The contacts displayed did not match my current user context*). The multiple sources of context and the ability to automatically filter content based on the current situation was an important aspect of the experiment. Both statements provided positive user evaluation results in regards to the functionality, with an average score of 3.58 and 3.71 out of 4.

Comprehensibility

The next group of statements are in regards to comprehensibility, which deals with how easily users can understand content presented in the application.

Statement 1 (*It is easy to see the available functions*) and 2 (*The features of the application are hard to use*) both asks about the features of the applications. The features, which included the home screen configuration and the context-awareness support, were generally well received with average scores of 3.50 and 3.11. However, for statement 2 we did receive various comments, where some evaluators explained that they thought the features were a bit hard to use. We believe this can be a result of the different experience the evaluators had with computers and mobile devices. We are of the impression that it is easier for an experienced smartphone user to take advantage of the features in the application. However, further testing is needed to verify this claim.

User Satisfaction

User satisfaction is the final usability attribute we grouped the statements into. This deals with the attitude of users toward using the application.

In experiment 2 (*Context-aware Meeting Room*) we investigated the acceptability of cloud computing during the evaluation. We wanted to continue this investigation further, and therefore we created several statements focusing on this issue. Statement 10 (*The close integration with Google services is an inconvenience, and I am not able to use the system without changing my existing or creating a new e-mail account at Google*), 14 (*I do not mind Cloud server downtime*), 15 (*I do not like sharing my personal information to a service that stores the information in the cloud*), and 16 (*Storing data in the Google cloud and combining this with personal information on the device is a useful feature*) asked the evaluators about cloud computing. We wanted to know if they found it to be an inconvenience, if they were comfortable sharing personal information, and if they found the close integration between the cloud-based server and the mobile client useful. The average results for these statements were 2.92 (14), 2.84 (15), and 3.26 (16).

Interestingly, the results differ in the two experiments. Whereas most users did not mind sharing information during the previous evaluation, we now recorded different opinions

in the responses. 11 users answering negatively towards sharing their personal information on a cloud service while 27 did not have a problem with it. There was considerably more disagreement in the answers in this evaluation, where more users did not like sharing personal information in a cloud-based solution. A more in-depth investigation of this issue, with focus on cloud computing acceptability, would be an interesting future research direction.

For statement 13 (*I would like to see integration with other online services such as online editing tools and user messaging applications*), we asked the users if they would like to extend the integration to other services, such as user messaging applications. It is of no surprise that the users wanted additional integration possibilities, and the average score of 3.29 verifies this.

In two statements, 3 (*The adaptability of the application is a feature I approve*) and 17 (*I find the cloud-to-device application feature useful*), we wanted to get the users opinion on the usefulness of the general adaptability of the application and the cloud-to-device (push messaging) feature. The average score were 3.45 and 3.52, indicating that the users were positive towards the mentioned features.

Finally, in statement 9 (*I would like to configure my phone from a cloud service on a daily basis*) we asked if they users would like to use the configuration feature of the application on a daily basis. The results, with an average score of 3.18, indicate that many users found the feature useful. Moreover, we find this result positive in regards to further development of the system.

6.1.8.1 Limitations

Similar to the previous experiments, we acknowledge the fact that the results might change with a considerably higher number of participants during the evaluation. We were able to gather a total of 38 users for the evaluation in this experiment, which we believe provides a valuable insight into the issues of mobile devices and context-awareness integrated with cloud computing.

6.2 Discussion

The research area covered in this chapter is remote information access. The experiment we presented was *Customised Android Home Screen*.

The application we created adapted the user interface, such as calendar and contacts information, based on context input. This was possible due to tagging calendar appointments and contact information with special *type-tags,* such as *${type=work}*. These were then read by the application, which automatically adapted the content for the users. One comment we received in regards to this feature was that the users would like to have the option to turn the filtering of contacts on and off. It was also suggested that it would be a good idea to include a filter-options button, where one could easily provide fine-grained filtering of the contacts, thereby giving the users more control. Loke (2006) highlights this issue of user control within context-awareness, and states that users will ultimately want to remain in control.

In this experiment we wanted to focus on three usability challenges, namely *mobile context*, *connectivity*, and *restrictive data entry methods* (Zhang & Adipat 2005). The created system offered mobile context with the integration of multiple sources of context. We integrated calendar data and contacts information, which made it possible to create situation-dependent information displayed to the users. The connectivity was included with the close integration of the mobile client and the cloud-based server. As part of the experiment we also included a comparison of the push-messaging technologies for Android to identify the overall best technology to use. Finally, the restrictive data entry methods were improved by using the context-awareness features to automatically adapt the content for the users. Additionally, we also offered a configuration option where the settings were edited using a normal computer and then directly transferred to the mobile device.

In regards to the cloud computing integration, it was interesting to see that a few users during the evaluation for this experiment were not comfortable sharing information in the cloud. There were also users who had reservations on their personal information being stored on some unknown place in the cloud provider data centre. This is a

different result than what we registered in a previous experiment, where most of the users did not have any issues with storing their personal information in a cloud-based system. Cloud computing acceptance would certainly be an interesting area for future user evaluations.

The last issue we want to highlight in the topic of cloud computing is vendor lock-in and the standardisation process. According to Clemons and Chen (2011), standards for cloud computing may reduce many of the risks of opportunistic behaviour of the cloud vendors. While we acknowledge the fact that standards are needed and will provide a significant benefit, we feel that Clemons and Chen (2011) overemphasise the importance of it. We created, with our Simple-C2DM project (presented in Appendix F), utilities for server applications that were provided as a platform neutral component, whereas the only requirement was support for Java. This was a conscious design decision, because we wanted to implement a different alternative than what was previously available, such as the Google Plugins support[39], which is directly tied to Google App Engine and GWT (Google Web Toolkit). In certain cases the proprietary APIs and tools cannot be completely avoided, but we believe that with a clear focus on design elements and close control of dependencies within the cloud projects, one can reduce the risk of being locked into a specific vendor significantly.

We also find that Clemons and Chen (2011) are not correct in their critical comments on the *Go Programming language*[40], where they suspect that the language is not created to address any deficiency in the full set of languages available elsewhere, and more to limit the customers' future mobility. We find this comment directly misleading, because the Google App Engine supports multiple languages, *Go*, *Python*, and *Java*, meaning they are not locked in because of the support for Go. Additionally, Go is an open source initiative, and is not proprietary to the Google cloud in any way.

6.3 Conclusion

In this chapter we presented remote information access. Our experiment focused on context management in a cloud computing environment, and how a mobile client can

[39] http://code.google.com/eclipse/docs/appengine_connected_android.html
[40] http://golang.org/

utilise this feature. The system consisted of a cloud computing service, which could store the registered context information that is used on the mobile device. Additionally, users were also able to manage their own context information using the cloud-based solution.

The system we created offered a close integration between a mobile client and a cloud-based server. To provide the network communication we used push-messaging technology, where the server could push messages directly to the mobile device. There are several push-messaging alternatives available for the Android platform, and as part of our experiment we wanted to investigate the best-suited technology to use in our proof-of-concept. The first objective focuses on this issue (*To find the overall best suited push messaging technology for integrating smartphone applications with cloud computing*). The comparison we conducted measured the performance of four push messaging technologies in regards to *response time, stability,* and *energy consumption*. The technology we found to provide the best overall results was C2DM, and therefore this was the technology used in our proof-of-concept system. We have not seen any previous research conducting a similar comparison of push-messaging technologies for Android.

We also identified a few issues with the C2DM API when using it to implement our system. Due to these issues we wanted to provide our own solution that used C2DM and fixed the major problems, such as the lack of flexibility and features. To improve the push-messaging library we created a new open source project called Simple-C2DM. This is presented in more detail in Appendix F.

The second objective (*To provide information that is relevant based on the current situation*) was to provide useful information to the end-user based on the current situation. We provided multiple sources of context in the system, including the integration of calendar and contacts data. By combining this information with the meta-tags, the mobile application could customise the content based on the situation of the user.

There are other research efforts focusing on using context-awareness to adapt the content, such as Ferris et al. (2010) which used location and (Huebscher et al. 2006)

who combined context data from the same context type. The difference in our work is that we provided various sources of context information in our proof-of-concept. The information included calendar and contacts data, and user configuration. Additionally, we also used a cloud-based server that was responsible for the context management, which is identified by Saha and Mukherjee (2003) as one of the main research challenges within pervasive computing. By using this service we were able to take advantage of the flexibility and scalability of cloud computing.

The general feedback of the system was positive, and the users found most features to be useful. In regards to the sensor integration, such as the ability to automatically change to a simplified user interface, received mixed comments from the users. This feature did not work consistently because we had issues differentiating events such as running from putting the phone on the table. In future releases of the system this feature needs more work, and also it should be possible to turn it off.

Chapter 7 – Conclusion

Usability studies in software projects have attracted considerable research efforts. A few relevant examples include Nielsen and Levy (1994), Kwakh and Han (2002), Zhang and Adipat (2005), and Andreasen et al. (2007). However, there are relatively few studies that focus on mobile usability (Coursaris & Kim 2011). Another important factor is that usability studies of mobile devices face unique challenges such as limited bandwidth, the unreliability of wireless networks, and environmental factors (Kenteris et al. 2009). Because of the lack of existing research and the unique challenges introduced by usability testing of mobile devices, we wanted to explore this research area in our work.

We specifically focused on smartphones, and this decision was made for three main reasons: 1) *the widespread use of smartphones*, 2) *smartphones provide considerable hardware and software capabilities*, and 3) *the smartphones can utilise high-speed wireless networks*.

To further narrow the scope of this thesis and to categorise the experiments, we specified three main research areas. These are *Wireless Personal Area Networks*, *Context-awareness*, and *Remote Information Access*.

We believe the three research areas provide important aspects of mobile and pervasive computing. They highlight important issues related to smartphones, ranging from local input from on-device sensors and actions/input based on the surroundings (*Context-awareness*), short-range communication possibilities (*WPAN*), and integration with flexible and highly scalable remote services (*Remote Information Access*). We have also shown, based on a thorough review of previous research efforts as shown in Chapter 2, that these specific research areas have not been examined in the same manner before.

Due to these findings we created the following aim:
To investigate aspects of usability in smartphone applications related to three areas: *Wireless Personal Area Networks*, *Context-awareness*, and *Remote Information Access*.

7.1 Wireless Personal Area Networks

We explored the WPAN area in our first experiment, where we focused on providing a tool for presentations during meetings. The created system used Bluetooth in a heterogeneous environment, specifically focusing on a multi-platform Bluetooth remote control system.

Because of the popularity of multiple mobile device types, verified by Gartner (2012), and the trend toward heterogeneous environments (Ryan & Gonsalves 2005), we wanted to create a system that supported multiple platforms for this experiment. We believe it provided a realistic scenario, where we supported multiple platforms and all the tests were conducted using only mobile devices. It is an important factor to use mobile devices, instead of emulators, whenever possible. The reason for this is because the user experience is quite different when using a physical device, compared to an emulator running on a computer device. The scenario of the usability test is more realistic if devices are used.

We identified three objectives, *Obj. 1-1, 1-2,* and *1-3*, for the *Multi-Platform Bluetooth Remote Control* experiment.

Obj. 1-1. **To provide an improved experience for presenters, specifically to improve the task of presenting.** One of the important features of the system was to add useful features for meeting presenters. We created a very flexible remote control solution that supported a very high degree of customisation. Additionally, each mobile application (Java ME and Windows Mobile) offered the same functionality, and was seamlessly integrated with the server. Bluetooth was the common technology used to provide the integration.

Obj. 1-2. **To create the possibility for each user to have their own independent solution (user specific configuration).** Our goal was to offer a system where each user could store their configuration settings separately. This was included to make it possible for multiple users to share the same system, but with their own configuration. This feature was implemented

by the client applications, which were responsible for storing the user configuration. The persisted values consisted of Bluetooth addresses and the key mappings.

Obj. 1-3. **To remove the need for additional hardware, such as cameras, microphones, and remote control devices.** The system was created to only use the mobile devices that were integrated with a Bluetooth-enabled laptop. The remote control was implemented to run on normal mobile phones and did not require any infrastructure related to network communication because we used Bluetooth technology, which provides short-range communication possibilities.

The focus on creating a heterogeneous system using Bluetooth provided a novel aspect we have not seen in any existing research. Another relevant example of previous research in this area is the work done by Fernández et al. (2006), where they focused on the server implementation and automatically adjusted the content based on the mobile device. We used the mobile devices as the main component in the system, with responsibilities including persisting the user configuration and providing a novel custom map key feature. Additionally, it also differs from the universal remote control system implemented by Feldbusch et al. (2003), which was created to work over the Internet. Our solution is targeted towards a local service and was created for a meeting scenario, specifically to be used with devices within close proximity.

7.2 Context-awareness

From the work on short-range personal network, we moved on to the next research area, which was *context-awareness*. A context-aware application tries to understand the situation in which a device operates, resulting in better and more efficient computing strategies (Kamal 2008).

In our research, we wanted to investigate context-awareness from two angles, namely 1) *Context-aware applications* and 2) *Smartphone sensors*.

For the area of context-aware applications, we created a heterogeneous system that took advantage of Bluetooth to provide proximity information. The experiment within this research area is called *Context-aware Meeting Room*, and included *objective 2-1* and *2-2*.

Obj. 2-1 **To handle time consuming meeting tasks automatically, such as meeting registration, information sharing, and meeting history.** The system used proximity information combined with calendar data to provide context-awareness to the mobile applications. One of the main features was the automatic sending of meeting notes, which contained additional information about the current presentation. All previous meeting notes were persisted on the mobile devices, which added the history functionality. Only the registered meeting participants in the proximity of the meeting room received the meeting notes due to a feature we implemented called *Bluetooth ping*.

Obj. 2-2 **To provide context-aware and useful information without the need for additional hardware in the meeting room.** Similar to objective 1-3 in that we only required standard smartphones and a laptop computer to operate the system within the meeting room. Additionally, we expanded the number of mobile platforms supported, by including clients for *Android, iPhone, Java ME,* and *Windows Mobile*. All meeting participant applications implemented the same features, but they were written specifically for each platform.

A significant difference between our work in this experiment and previous research, such as Mantoro and Johnson (2003) and Ferris et al. (2010), is that we did not focus on using location as input to the system. Instead, we took advantage of the short-range features of Bluetooth to provide proximity as a context input source. Additionally, in contrast to previous research efforts that created smart spaces, such as Jaimes and Miyazaki (2005) and Parviainen et al. (2006), we utilised standard laptops and smartphones, and did not require installation of external sensors, cameras, or microphones. The use of standard hardware is a feature we explored further in all experiments conducted as part of this thesis. This was because we saw the potential

smartphones have to offer a wide variety of services, and we also recognised the benefit it was using only standard components, avoiding the need for additional hardware installations, such as external sensors, cameras, and microphones, to operate the system.

For the second part of context-awareness, we investigated the area of smartphone sensors. We created a system called *PainDroid*, which provided a multimodal virtual reality solution for pain management. The application was created to enable patients to describe the pain experience in a more realistic and interactive manner from the comfort of his/her home. In this experiment we looked closely at the sensors on the Android platform and how they can be utilised. For this experiment we defined *objective 3-1*.

Obj. 3-1 **To improve and expand the way users with dexterity problems can interact with smartphone (and tablet) applications.** We utilised the on-device sensors to provide wheelchair users suffering from mobility difficulties, such as the inability to use all fingers, with an additional option for controlling the application. We used both the accelerometer and magnetometer to control the 3D model, which was used for selecting the areas of the body where pain was occurring.

The on-device sensors were used to provide input to the system, which added an additional control mechanism. Similar to our work, the research effort by Bhorasker et al. (2012) also utilised smartphone sensors, however, in our work we focus on using the sensors as a control mechanism, whereas they used it as a passive input source for registering road conditions. Another novel aspect of our work is that we provided a very practical approach to challenges and possibilities using the on-device sensors, which we have not seen in any previous research.

7.3 Remote Information Access

The last research area we investigated was *Remote Information Access*, where we integrated the mobile clients with server applications deployed on a remote cloud-computing platform.

For this experiment, called *Customised Android Home Screen*, we implemented a system that stored context information on a cloud-based server application. The system included *Mobile and Cloud Integration*, where it provided users with the ability to configure their Android device through the use of the cloud-based service. The mobile device collected context data from several sources and cooperated with the cloud server to provide useful features, such as a context-aware calendar and contacts list.

For the *Customised Android Home Screen* we defined *objective 4-1* and *4*-2.

Obj. 4-1 **To find the overall best suited push messaging technology for integrating smartphone applications with cloud computing.** We conducted a comparison of four push messaging technologies for Android that we were able to easily integrate with a cloud-computing platform, namely the Google App Engine. The push-messaging technologies were compared them based on *response time*, *stability*, and *energy consumption*. Based on the results, we selected to use the C2DM technology that provided the best results overall. Additionally, we also implemented our own library, built on C2DM, to improve the API and to make it easier to develop new projects using push messaging and also offer new functionality. Hence, we provided additional features to improve the integration between the mobile device and cloud-based servers.

Obj. 4-2 **To provide information that is relevant based on the current situation (filter content, cloud integration).** The cloud-based server was used to store the user configuration, which included the icons that was shown on the mobile device. The system used standard Google accounts to provide a secure login process and to make it possible for each user to store and access their own configuration. The tag information used in the contacts and calendar features were also stored in the cloud. With these tags, the users were able to filter content, for example if they had an upcoming work-related meeting the contacts would be filtered based on this information.

A previous and relevant research effort in this area, conducted by Huebscher et al. (2006), combines context data of the same type to provide higher quality results. We are, on the other hand, using various sources to provide context information to the system, such as calendar data and user configuration. In addition, we use a cloud-based server application to be responsible for the context management. By using this service we were able to take advantage of the flexibility and scalability, which are important and useful features offered by cloud computing.

In the next section, we continue by providing the main contributions from our work. We will also state the limitations and then finally present the direction for future work.

7.4 Research Contributions

The main contributions from our research are provided through the aim and objectives. In this section we will go into detail on the contribution by looking at each objective and describing the contributions made in the context of these.

As part of the first experiment, *Multi-platform Bluetooth Remote Control*, we provided three objectives. For objective 1-1 (*To provide an improved experience for presenters, specifically to improve the task of presenting*) we created a system, running on multiple mobile platforms, which allowed the meeting presenters to use their mobile phone to control the presentation. The remote control application provided a highly flexible solution, where each button on the phone could be configured to any combination of keyboard actions on the laptop computer. Based on the usability evaluation conducted as part of this experiment, the users found the application easy to understand and use. Additionally, the features of the system were perceived as useful and most of the evaluators would use it for presentations in the future. The main issue raised by the evaluators was the documentation. The user documentation was not detailed enough in certain aspects, such as explaining installation and the Bluetooth pairing process.

The next objective, 1-2 (*To create the possibility for each user to have their own independent solution*), was selected because we wanted to make it possible for each user to have separate configuration. This provided the benefit of allowing multiple users

connect to the same server, but using their own configuration. Additionally, each user could connect to any server and use the client application, which was previously configured, right away. There was no setup and storage needed on the laptop where the server application was running. To implement this feature, both the configuration of keyboard actions and Bluetooth addresses were stored locally on each device. This feature was well received by the users in the evaluation.

For our final objective in the first experiment, 1-3 (*To remove the need for additional hardware, such as cameras, microphones, and remote control devices*) we focused on the use of normal mobile phones that were connected to a Bluetooth-enabled laptop. In contrast to previous research, we did not require any additional hardware in the meeting room. Moreover, we provided support for multiple platforms (Java ME and Windows Mobile), which is an important aspect due to the number of mobile platforms on the market. Also, we have not seen the multiple-platform support being highlighted in similar research efforts. This provided us with the benefit of doing the evaluation on the mobile devices of the users, because we supported the two main platforms that were identified prior to the test.

Moving on to experiment 2, *Context-aware Meeting Room*, we had two main objectives. For the first objective, 2-1 (*To handle time consuming meeting tasks automatically, such as meeting registration, information sharing, and meeting history*), we created a system that would automatically register meeting participants when they entered the meeting room. This was implemented with a feature called *Bluetooth ping* (see Appendix D for source code), which automatically registered devices in close proximity. This feature also provided invisibility, because it automatically registered the meeting participants in our Context-aware Meeting Room without the need for any manual tasks. Saha and Mukherjee (2003) highlight invisibility as one of the most important challenges to overcome within pervasive systems.

We found the Bluetooth ping feature to be a particularly important contribution, and we can see the potential benefit for other context-aware systems to utilise this feature for providing proximity information. One main advantage is that it supports devices not being able to set a permanently visible Bluetooth mode, which was the case with the Android and iPhone devices used in our experiments. A general description of how it

works is that we send a request for a specific service on the mobile device (hands-free service in our experiment) and wait for a response. If a response is received, we know the device is within close proximity. For Bluetooth, close proximity is usually within 10m. We imagine that similar solutions, using other technology than Bluetooth, can also be created and could provide an interesting area for future research. We have not seen any previous research efforts providing this functionality. During the evaluation this feature worked as intended, and the user feedback was positive.

The information sharing in the system was implemented by automatically pushing meeting notes to the registered meeting participants. We also had history functionality, where previous meetings were persisted on the smartphone clients. With the proof-of-concept created for this experiment, we used a technology called Protocol Buffer to enable the communication between the server applications and the mobile clients. Java ME did not initially support this technology when we started the implementation, and therefore we created a new open source project[41] to provide this feature. Specifically, we created a Java ME library, consisting of a code generator and runtime library, which was compatible with the standard Google Protocol Buffer library.

Based on the usability test conducted for the *Context-aware Meeting Room* experiment, we found that the evaluators generally found the system to be useful. Additionally, they appreciated and saw the potential of the meeting notes and history functionality. The performance of the system, in regards to receiving meeting notes in sync with the presentation, was also satisfactory. One improvement the users wanted in the system was the support for more content, such as files, images, and videos, in addition to the meeting notes functionality.

A surprising result from the user evaluation was that almost all users that were part of the evaluation were comfortable with personal information, such as name and e-mail, being stored in a cloud-based server. This answer was unexpected because of the well-known privacy concerns in cloud computing, such as the lack of transparency (Vigfusson & Chockler 2010). Because the result was somewhat surprising, we wanted to investigate this further in a later experiment, namely experiment 4 (*Customised Android Home Screen*).

[41] http://code.google.com/p/protobuf-javame/

The next objective in the second experiment, 2-2 (*To provide context-aware and useful information without the need for additional hardware in the meeting room*), is similar to objective 1-3, but we added support for four platforms. Moreover, we only required standard smartphones and a laptop to install and use the system. This is different than other relevant research we found, such as Jaimes and Miyazaki (2005) and Parviainen et al. (2006). All the participant clients supported the same functionality, and we also included a presenter client from the Android platform.

In the third experiment, *PainDroid*, we included one objective, namely 3-1 (*To improve and expand the way users with dexterity problems can interact with smartphone (and tablet) applications*). The application provided a possibility for disabled users to interact with the application using movement in addition to the touch-based interface.

The result from the usability test was that generally the users found the application very interesting. A few users did have difficulties getting used to the sensor-based controls. Additionally, an interesting aspect we found during the evaluation was that there was no direct link between upper-body mobility and the ability to use the sensor input in our evaluation. We conducted the experiment with some users that had full mobility in the upper body and had a hard time using the controls, however, on the other hand we had users that had limitations in their hands/fingers and were able to use the controls without any issues.

The users also found the system easy to learn and they did not report of any major inconsistencies in the system. The functionalities of PainDroid were perceived as well integrated and most users did not see the need for technical support using the system. This is further explained by the confidence the users had when testing the PainDroid application.

In addition to the wheelchair users we involved two clinicians in the evaluation, namely a General Practitioner (GP) and a Rheumatologist (RH). Their responses were generally positive towards the potential use of the system in a clinical setting, which is highlighted by the following statements:

"*...very useful in the consultation room for mapping pain...and compare over time to see the improvement..."* (GP)

"*...advantageous if you can keep in memory previous recordings and patients don't know that - then you can keep on seeing how pain has changed...that would be quite neat!"* (RH)

In regards to the sensor input for an additional control mechanism, the rheumatologist appreciated the multimodal interface of PainDroid. This was because patients with neurological illness and with limited hand mobility could use the sensor-based interface as an extra modality to interact with the application. We believe that these findings may be of considerable interest to other researchers, and particularly healthcare providers. Moreover, we were generally very encouraged by the positive attitudes of both the clinicians and the wheelchair users toward the system and its potential use.

Furthermore, in this experiment we also provided an in-depth look at how the sensors can be used on the Android platform, and issues we encountered with the implemented proof-of-concept system, such as the difference between device models and the accelerometer input data. We have not seen these details on the sensor integration presented in previous research.

For the final experiment, *Customised Android Home Screen*, we had two objectives. The first objective is 4-1 (*To find the overall best suited push messaging technology for integrating smartphone applications with cloud computing*). When we were developing the proof-of-concept for this experiment we wanted to utilise push messaging to integrate the mobile device with the cloud-based server. The need for push messaging was due to the features where the cloud-based server could *push* updates of the home screen configuration directly to the mobile devices. One example of a benefit of using push-based technology is the avoidance of unnecessary requests compared to poll-based alternatives. The results from the test provided us with a technology that provided the best results in regards to *response times*, *stability*, and *energy consumption*. We therefore used this technology, which was C2DM, in the main experiment.

Secondly, the other contribution in regards to push messaging was to extend the features and improve the capabilities of C2DM by introducing our own open source library called Simple- C2DM[42] (presented in detail in Appendix F). We created this library to target specific areas of C2DM that we felt were either cumbersome to use, such as the inheritance limitations, or simply missing, for instance registration id handling. By including features such as automatic registration id handling and annotation support for increased flexibility, our library provided a contribution in the area of push messaging on Android.

The second objective for this experiment was 4-2 (*To provide information that is relevant based on the current situation*). By including multiple sources for context, such as calendar and contacts data, we were able to tailor the information to the users based on the current situation. Moreover, by providing a simple way for the users to login to a webpage and configure the home screen on their phone, we added context management and a close integration with a cloud-based server. We have not seen this integration of multiple sources of context used in the same manner before, and additionally we created our own meta-tags that made this integration possible. The close integration with the cloud-based server used the push-messaging technology, which was identified in objective 4-1.

As previously mentioned, we wanted to further investigate the user acceptance of the close integration with cloud computing. In this context we required the users to store their name and e-mail address on a cloud-based server. The results, compared to the previous questionnaire where we looked at this issue (experiment 2), were quite interesting. In the previous responses we received positive feedback toward using the cloud, and the users did not have any concerns storing personal information on our cloud-based server. However, in this evaluation we recorded a mixed response, where some users did state their concern. This area of cloud computing acceptability needs more research, in addition to our initial investigation of it.

Finally, we would like to express a few general comments based on our experience of working with these experiments.

[42] http://code.google.com/p/simple-c2dm/

We developed heterogeneous systems in several of the experiments and therefore gained experience with the challenges that these systems often face. According to Saha and Mukherjee (2003), the task of developing applications that run across all platforms will be exceedingly difficult. Nevertheless, we were able to create these systems efficiently because we followed two general ideas:

1. **Share the system design and architecture**. An important aspect of software projects is the overall design and architecture of the system. We found that in most cases the majority of this design work was platform independent and could simply be reused on all the platforms. In certain corner cases this aspect does not apply, such as differences in Bluetooth support where we needed to add platform specific features.
2. **Where possible and practical, use external resources such as cloud computing, to be responsible for common services/features**. By including common operations on the server side and integrating them with client application features, one can offer appealing features while keeping the client as simple as possible. This is directly related to SOA, which provides several benefits: *Improved agility*, *reduction of cost*, and *reduction of risks* (Dan et al. 2008). Additionally, the ability to use cloud-based platform features, for example security and authentication, is a significant advantage. We did not have to implement a custom solution of these features ourselves, but could focus on our main business logic.

In our proof-of-concepts created as part of our research, we have shown that it is a viable option to implement native clients for the different mobile platforms. We believe that our experiences, as presented above, can benefit other similar projects that want to implement a mobile solution, and can be particularly interesting to systems wanting to take advantage of the cloud computing integration. Additionally, a major benefit of our work is that all systems were implemented and evaluated. We had working systems tested on mobile devices (not using emulators). We believe this is an advantage because it provides a more realistic scenario.

7.4.1 Limitations

In our usability study we selected three research areas (*WPANs, Context-awareness,* and *Remote Information Access*). The area of usability for smartphones is considerably larger than this, but it was a deliberate choice because we wanted to include a suitable scope for our work. However, a more general approach to usability for smartphones is certainly an interesting area that can be improved with future research.

One important aspect of the contributions is to be aware of the limitations that apply to the findings. Generally, our user evaluations were evaluated by a limited number of users (*exp-1*: 16 users, 2: *exp-2*: 40 users, *exp-3*: 7 users (w/disabilities), and *exp-4*: 38 users). While we think our results do provide valuable insight into the areas investigated, we also acknowledge the fact that the results might be different if the number of evaluators were significantly increased.

In experiment 3 (*PainDroid*) the application created was limited to the use of two sensors, specifically magnetometer and accelerometer. We believe this is enough to get an understanding of the support for sensors on Android. However, we acknowledge that a larger study with more devices and sensor types can provide additional and useful information. Additionally, adding a new mobile platform to the investigation, such as iOS, could also provide useful insight into the use of smartphone sensors.

As previously mentioned, we focused on cloud computing in several experiments, namely 2 and 4. Specifically, we used the Google App Engine, which is a PaaS platform developed and maintained by Google. In cloud computing there is a danger of being locked into a specific cloud vendor. According to Clemons and Chen (2011), standards for cloud computing may reduce many of the risks of opportunistic behaviour of the cloud vendors. While the Google App Engine supports widely used platforms such as Java and Python[43], it is also worth mentioning that it has limitations such as the aspect that a Java application cannot spawn background threads that outlive the request that created them[44]. We see that the connection to the Google App Engine is a limitation, namely that we are bound to a platform we do not directly control ourselves. However,

[43] https://developers.google.com/appengine/docs/whatisgoogleappengine
[44] https://developers.google.com/appengine/docs/java/runtime#The_Sandbox

we believe that the risk of vendor lock-in can be significantly reduced by taking precautions, such as a clear focus on design elements and close control of dependencies within the software projects. Nevertheless, cloud computing would certainly benefit from industry wide standards, namely the addition of support to easily move between cloud vendors without any programmatic changes.

In the last experiment we conducted a benchmarking test of push messaging technologies (experiment 4). In this experiment we experienced issues running all of the technologies over a period of time. This resulted in certain messages not being received. The test would run reliably for period of time, before the messages were no longer received on the mobile client. A restart of the Android client would solve the problem, but this scenario happened multiple times. However, we still managed to get a decent amount of data (with nearly 300 messages) with the use of the Samsung Galaxy Tab 10.1. With both Urban Airship and Xtify were tested on development servers. In the case of Xtify, we had to run this separately because there is a limit of 300 messages sent when using a basic account.

Additionally, in the energy consumption test, which was also part of experiment 4, we did have a few limitations. Mainly, the energy consumption tests ran over several days and this means that we could not control every variable, such as the use of the network and varying latency from the services. Nevertheless, we took several precautions when running this test, such as running them from the same network and started the tests on the same time of day. Additionally, we also turned off the screen and disabled the auto-sync feature to prevent applications from using network communication in the background. Even with this limitation in mind, we still believe there is value in the results we gathered as part of this experiment.

7.5 Future Work

In this section we will provide the opportunities for future work, which will expand on our research. We start by the presenting the issue of heterogeneity. According to Charland and LeRouch (2011), it can be very expensive to write separate applications in the native language of the platforms. Therefore, when developing multi-platform

systems there is also the possibility to use HTML 5 technology to create a shared application. While we acknowledge this fact, we also see that in today's environment there are scenarios where HTML 5 does not provide the wanted performance requirements or lacks specific features. In the topic of heterogeneity we see the potential for future research providing more detail on this idea of a common platform, such as a closer investigation of HTML 5 compared to native applications.

In experiment 3, we presented the platform support for sensors, and showed that this is a well-integrated feature of Android. For future research this is a benefit that should be taken advantage of to introduce *context-awareness* more directly into the applications. We believe that further work focusing on smartphone sensors in a context-aware setting is an interesting research opportunity within mobile environments.

Furthermore, we used cloud computing in both experiment 2 and 4, and from our user evaluations we received different results based on the sharing of personal information on a cloud-based server. This is also highlighted in relevant research, conducted by Dillon et al. (2010) and Vigfusson and Chockler (2010). Specifically, large cloud-computing providers have access to an enormous amount of data, and the lack of transparent knowledge on how this information is used has provoked concerns. This would certainly be an interesting topic for future user evaluations on the topic of cloud computing acceptance in regards to personal information sharing.

In experiment 4 we created an open source library, improving on the features of C2DM that is presented in Appendix F. This library provided a contribution in the area of push messaging on Android, with features such as automatic registration id handling. We would like to continue the development of this project in future work. Moreover, in the same experiment we conducted a benchmarking test, where we used Android devices with version 2 and 3. At the time of writing, an official version of Android 4 is not released for the devices used in the benchmark test. It would be interesting and useful for future work to focus on the issues we experienced during the test, by including Android version 4 devices, and investigating if the problems are fixed or at least improved in this newer version of the operating system. Finally, a more comprehensive benchmarking test that includes more cloud computing platforms, such as Amazon EC2, would also be a useful direction for future research.

References

Ahmed, S., Sharmin, M. & Ahamed, S.I., 2005. A smart meeting room with pervasive computing technologies. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, 2005 and First ACIS International Workshop on Self-Assembling Wireless Networks. SNPD/SAWN 2005. Sixth International Conference on*. pp. 366 – 371.

Alavi, M., Carlson, P. & Brooke, G., 1989. The ecology of MIS research: a twenty year status review. In *Proceedings of the tenth international conference on Information Systems*. ICIS '89. New York, NY, USA: ACM, pp. 363–375.

Alhamad, M. et al., 2010. Response time for cloud computing providers. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications and Services*. iiWAS '10. New York, NY, USA: ACM, pp. 603–606.

Andreasen, M.S. et al., 2007. What happened to remote usability testing?: an empirical study of three methods. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. CHI '07. New York, NY, USA: ACM, pp. 1405–1414.

Bagchi, S., 2010. On reliable distributed IPC/RPC design for interactive mobile applications. In *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. MEDES '10. New York, NY, USA: ACM, pp. 33–38.

Bak, J.O. et al., 2008. Obstacles to usability evaluation in practice: a survey of software development organizations. In *Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges*. NordiCHI '08. New York, NY, USA: ACM, pp. 23–32.

Bertino, E., Jajodia, S. & Samarati, P., 1999. A flexible authorization mechanism for relational data management systems. *ACM Trans. Inf. Syst.*, 17(2), pp.101–140.

Bhagwat, P., Perkins, C. & Tripathi, S., 1996. Network layer mobility: an architecture and survey. *Personal Communications, IEEE*, 3(3), pp.54 –64.

Bhoraskar, R. et al., 2012. Wolverine: Traffic and road condition estimation using smartphone sensors. In *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*. pp. 1 –6.

Binnig, C. et al., 2009. How is the weather tomorrow?: towards a benchmark for the cloud. In *Proceedings of the Second International Workshop on Testing Database Systems*. DBTest '09. New York, NY, USA: ACM, pp. 9:1–9:6.

Birrell, A.D. & Nelson, B.J., 1984. Implementing remote procedure calls. *ACM Trans. Comput. Syst.*, 2(1), pp.39–59.

Bisdikian, C., 2001. An overview of the Bluetooth wireless technology. *Communications Magazine, IEEE*, 39(12), pp.86 –94.

Boger, M., 2001. *Java in Distributed Systems: Concurrency, Distribution and Persistence* 1st ed., Wiley.

Borg, A. et al., 1989. Fault tolerance under UNIX. *ACM Trans. Comput. Syst.*, 7(1), pp.1–24.

Brooke, 1996. SUS: A quick and dirty usability scale. In *Usability evaluation in industry*. Taylor and Francis.

Burrell, G. & Morgan, G., 1979. *Sociological Paradigms and Organisational Analysis: Elements of the Sociology of Corporate Life*, Ashgate Publishing.

Buyya, R. et al., 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.*, 25(6), pp.599–616.

Calheiros, R.N. et al., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience, Software: Practice and Experience*, 41, 41(1, 1), pp.23, 23–50, 50.

Chacon, S., 2009. *Pro Git* 1st ed., Apress.

Chandrasekaran, P. & Joshi, A., 2002. MobileIQ: a framework for mobile information access. In *Mobile Data Management, 2002. Proceedings. Third International Conference on*. pp. 43 – 50.

Charland, A. & Leroux, B., 2011. Mobile application development. *Communications of the ACM*, 54(5), p.49.

Chary, R. et al., 2006. Sensor-Based Power Management for Mobile Devices. In *Computers and Communications, 2006. ISCC '06. Proceedings. 11th IEEE Symposium on*. pp. 263 – 269.

Chen, H. et al., 2004. Intelligent Agents Meet Semantic Web in a Smart Meeting Room. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '04. Washington, DC, USA: IEEE Computer Society, pp. 854–861.

Chen & Hirschheim, 2004. A paradigmatic and methodological examination of information systems research from 1991 to 2001. *Information Systems Journal, Information Systems Journal*, 14, 14(3, 3), pp.197, 197–235, 235.

Chen & Kotz, 2000. *A Survey of Context-Aware Mobile Computing Research*, Hanover, NH, USA: Dartmouth College.

Cheng, J. et al., 2007. SmartSiren: virus detection and alert for smartphones. In *Proceedings of the 5th international conference on Mobile systems, applications and services*. MobiSys '07. New York, NY, USA: ACM, pp. 258–271.

Cheverst, K. et al., 2001. Using Context as a Crystal Ball: Rewards and Pitfalls. *Personal Ubiquitous Comput.*, 5(1), pp.8–11.

Cidon, A. et al., 2011. MARS: adaptive remote execution for multi-threaded mobile devices. In *Proceedings of the 3rd ACM SOSP Workshop on Networking, Systems, and Applications on Mobile Handhelds*. MobiHeld '11. New York, NY, USA: ACM, p. 1:1–1:6.

Clemons & Chen, 2011. Making the Decision to Contract for Cloud Services: Managing the Risk of an Extreme Form of IT Outsourcing. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*. pp. 1 –10.

Cook, D.J. & Das, S.K., 2007. How smart are our environments? An updated look at the state of the art. *Pervasive Mob. Comput.*, 3(2), pp.53–73.

Coursaris, C.K. & Kim, D.J., 2011. A Meta-Analytical Review of Empirical Mobile Usability Studies. *J. Usability Studies*, 6(3), pp.11:117–11:171.

Dan, A., Johnson, R.D. & Carrato, T., 2008. SOA service reuse by design. In *Proceedings of the 2nd international workshop on Systems development in SOA environments*. SDSOA '08. New York, NY, USA: ACM, pp. 25–28.

Dey, A.K., 2001. Understanding and Using Context. *Personal Ubiquitous Comput.*, 5(1), pp.4–7.

Dillon, T., Wu, C. & Chang, E., 2010. Cloud Computing: Issues and Challenges. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*. pp. 27 –33.

DiMarzio, J.F., 2008. *Android a Programmers Guide* 1st ed., McGraw-Hill Osborne Media.

Do, T.M.T., Blom, J. & Gatica-Perez, D., 2011. Smartphone usage in the wild: a large-scale analysis of applications and context. In *Proceedings of the 13th international conference on multimodal interfaces*. ICMI '11. New York, NY, USA: ACM, pp. 353–360.

Doukas, C., Pliakas, T. & Maglogiannis, I., 2010. Mobile healthcare information management utilizing Cloud Computing and Android OS. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. pp. 1037 –1040.

Eagle, N. & Pentland, A., 2006. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4), pp.255–268.

Elsenpeter, R.C., Velte, T. & Velte, A., 2009. *Cloud Computing, A Practical Approach* 1st ed., McGraw-Hill Osborne Media.

Evans, E., 2003. *Domain-Driven Design: Tackling Complexity in the Heart of Software* 1st ed., Addison-Wesley Professional.

Fei, Y., Zhong, L. & Jha, N.K., 2008. An energy-aware framework for dynamic software management in mobile computing systems. *ACM Trans. Embed. Comput. Syst.*, 7(3), pp.27:1–27:31.

Feldbusch, F. et al., 2003. The BTRC Bluetooth remote control system. *Personal Ubiquitous Comput.*, 7(2), pp.102–112.

Fernandez, C. et al., 2006. MOVILTOOTH: a Bluetooth context-aware system with push technology. In *Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean*. pp. 761 –764.

Ferris, B., Watkins, K. & Borning, A., 2010. OneBusAway: results from providing real-time arrival information for public transit. In *Proceedings of the 28th international conference on Human factors in computing systems*. CHI '10. New York, NY, USA: ACM, pp. 1807–1816.

Fitzgerald, B. & Howcroft, D., 1998. Competing dichotomies in IS research and possible strategies for resolution. In *Proceedings of the international conference on Information systems*. ICIS '98. Atlanta, GA, USA: Association for Information Systems, pp. 155–164.

Flinn, J. & Satyanarayanan, M., 2004. Managing battery lifetime with energy-aware adaptation. *ACM Trans. Comput. Syst.*, 22(2), pp.137–179.

Flores, H., Srirama, S.N. & Paniagua, C., 2011. A generic middleware framework for handling process intensive hybrid cloud services from mobiles. In *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*. MoMM '11. New York, NY, USA: ACM, pp. 87–94.

Fowler, M., 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language* 3rd ed., Addison-Wesley Professional.

Frank, L., 1999. Atomicity Implementation in Mobile Computing. In *Proceedings of the 10th International Workshop on Database & Expert Systems Applications*. DEXA '99. Washington, DC, USA: IEEE Computer Society, p. 105–.

Galliers, R.D., 1991. Choosing appropriate information systems research approaches: a revised taxonomy. In pp. 155–173.

Gao, T., Guo, N. & Zhu, Z., 2009. A Distributed Security MIPv6 Model Based on Dynamic Policies. In *Hybrid Intelligent Systems, 2009. HIS '09. Ninth International Conference on*. pp. 423 –426.

Gartner, 2011. Gartner Identifies the Top 10 Strategic Technologies for 2012. Available at: http://www.gartner.com/it/page.jsp?id=1826214 [Accessed April 13, 2012].

Gartner, 2012. Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth. Available at: http://www.gartner.com/it/page.jsp?id=1924314 [Accessed April 13, 2012].

Gellersen, H.W., Schmidt, A. & Beigl, M., 2002. Multi-sensor context-awareness in mobile devices and smart artifacts. *Mob. Netw. Appl.*, 7(5), pp.341–351.

Georgiev, I.K. & Georgiev, I.I., 2001. A security model for distributed computing. *J. Comput. Sci. Coll.*, 17(1), pp.178–186.

Glesner, M. et al., 2004. Reconfigurable platforms for ubiquitous computing. In *Proceedings of the 1st conference on Computing frontiers*. CF '04. New York, NY, USA: ACM, pp. 377–389.

González-Castaño, F.J. et al., 2005. Bluetooth-assisted context-awareness in educational data networks. *Comput. Educ.*, 45(1), pp.105–121.

Greenhalgh, C. et al., 2007. Addressing mobile phone diversity in ubicomp experience development. In *Proceedings of the 9th international conference on Ubiquitous computing*. UbiComp '07. Berlin, Heidelberg: Springer-Verlag, pp. 447–464.

Gregg, D.G., Kulkarni, U.R. & Vinzé, A.S., 2001. Understanding the Philosophical Underpinnings of Software Engineering Research in Information Systems. *Information Systems Frontiers*, 3(2), pp.169–183.

Gross, T., Steenkiste, P. & Subhlok, J., 1999. Adaptive Distributed Applications on Heterogeneous Networks. In *Proceedings of the Eighth Heterogeneous Computing Workshop*. HCW '99. Washington, DC, USA: IEEE Computer Society, p. 209–.

Hager, C.T. & Midkiff, S.F., 2003. An analysis of Bluetooth security vulnerabilities. In *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*. pp. 1825 – 1831 vol.3.

Hall, S.P. & Anderson, E., 2009. Operating systems for mobile computing. *J. Comput. Sci. Coll.*, 25(2), pp.64–71.

Hansmann, U. et al., 2000. *Pervasive Computing: The Mobile World* 2nd ed., Springer.

Harn, L. & Huang, D., 1994. A Protocol for Establishing Secure Communication Channels in a Large Network. *IEEE Trans. on Knowl. and Data Eng.*, 6(1), pp.188–191.

Henning, M., 2009. API design matters. *Communications of the ACM*, 52(5), p.46.

Hevner, A. & Chatterjee, S., 2010. *Design Research in Information Systems: Theory and Practice* 1st ed., Springer.

Hohpe, G. & Woolf, B., 2003. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions* 1st ed., Addison-Wesley Professional.

Huebscher, M. et al., 2006. Issues in Developing Ubicomp Applications on Symbian Phones. In *Proceedings of the International Workshop on System Support for Future Mobile Computing Applications*. FUMCA '06. Washington, DC, USA: IEEE Computer Society, pp. 51–56.

International Organization for Standardization, 1998. ISO 9241- 11: Guidance on usability.

Jaimes, A. & Miyazaki, J., 2005. Building a Smart Meeting Room: From Infrastructure to the Video Gap (Research and Open Issues). In *Data Engineering Workshops, 2005. 21st International Conference on*. p. 1173.

Johns, R., 2010. Likert items and scales of measurement? *Methods*, 1(March), pp.1–11.

Jones, G.J.F., 2005. Challenges and opportunities of context-aware information access. In *Ubiquitous Data Management, 2005. UDM 2005. International Workshop on*. pp. 53 – 60.

Kächele, S., Domaschka, J. & Hauck, F.J., 2011. COSCA: an easy-to-use component-based PaaS cloud system for common applications. In *Proceedings of the First International Workshop on Cloud Computing Platforms*. CloudCP '11. New York, NY, USA: ACM, pp. 4:1–4:6.

Kale, A. & Bharambe, U., 2009. Highly available fault tolerant distributed computing using reflection and replication. In *Proceedings of the International Conference on Advances in Computing, Communication and Control*. ICAC3 '09. New York, NY, USA: ACM, pp. 251–256.

Kamal, R., 2008. *Mobile Computing*, Oxford University Press, USA.

Kang, Y.M., Cho, C. & Lee, S., 2011. Analysis of factors affecting the adoption of smartphones. In *Technology Management Conference (ITMC), 2011 IEEE International*. pp. 919 –925.

Kaptein, M.C., Nass, C. & Markopoulos, P., 2010. Powerful and consistent analysis of likert-type ratingscales. In *Proceedings of the 28th international conference on Human factors in computing systems*. CHI '10. New York, NY, USA: ACM, pp. 2391–2394.

Kendall, S.C. et al., 1994. *A Note on Distributed Computing*, Mountain View, CA, USA: Sun Microsystems, Inc.

Kenteris, M., Gavalas, D. & Economou, D., 2009. An innovative mobile electronic tourist guide application. *Personal Ubiquitous Comput.*, 13(2), pp.103–118.

Khairullah, E.F. & Al-Aama, A.Y., 2009. Improving accuracy of context aware data replication in mobile computing application. In *Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*. MoMM '09. New York, NY, USA: ACM, pp. 498–503.

Khajeh-Hosseini, A. et al., 2012. The Cloud Adoption Toolkit: supporting cloud adoption decisions in the enterprise. *Software: Practice and Experience, Software: Practice and Experience*, 42, 42(4, 4), pp.447, 447–465, 465.

Kitchenham, B. & Pfleeger, S.L., 2002a. Principles of survey research part 4: questionnaire evaluation. *SIGSOFT Softw. Eng. Notes*, 27(3), pp.20–23.

Kitchenham & Pfleeger, 2002b. Principles of survey research part 3: constructing a survey instrument. *SIGSOFT Softw. Eng. Notes*, 27(2), pp.20–24.

Kjeldskov, J. & Skov, M.B., 2007. Exploring context-awareness for ubiquitous computing in the healthcare domain. *Personal Ubiquitous Comput.*, 11(7), pp.549–562.

Kjeldskov & Paay, 2006. Public Pervasive Computing: Making the Invisible Visible. *Computer*, 39(9), pp.60 –65.

Kochan, S.G., 2011. *Programming in Objective-C (4th Edition)* 4th ed., Addison-Wesley Professional.

Korhonen, H. & Koivisto, E.M.I., 2006. Playability heuristics for mobile games. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*. MobileHCI '06. New York, NY, USA: ACM, pp. 9–16.

Krug, S., 2009. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems* 1st ed., New Riders Press.

Kurose, J.F. & Ross, K.W., 2002. *Computer Networking: A Top-Down Approach Featuring the Internet* 2nd ed., Addison Wesley.

Kwahk, J. & Han, S.H., 2002. A methodology for evaluating the usability of audiovisual consumer electronic products. *Applied Ergonomics*, 33(5), pp.419–431.

Kwapisz, J.R., Weiss, G.M. & Moore, S.A., 2011. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2), pp.74–82.

Kwon, O., Choi, S. & Park, G., 2005. NAMA: a context-aware multi-agent based web service approach to proactive need identification for personalized reminder systems. *Expert Syst. Appl.*, 29(1), pp.17–32.

Li, J., Bose, A. & Zhao, Y.Q., 2005. The study of wireless local area networks and wireless personal area networks. In *Electrical and Computer Engineering, 2005. Canadian Conference on*. pp. 1415 –1418.

Loke, S., 2006. *Context-Aware Pervasive Systems: Architectures for a New Breed of Applications* 1st ed., Auerbach Publications.

Malik, A.K. & Dustdar, S., 2011. Enhanced sharing and privacy in distributed information sharing environments. In *Information Assurance and Security (IAS), 2011 7th International Conference on*. pp. 286 –291.

Maniatis, P. & Chun, B.-G., 2011. Small trusted primitives for dependable systems. *SIGOPS Oper. Syst. Rev.*, 45(1), pp.126–141.

Mantoro, T. & Johnson, C., 2003. Location history in a low-cost context awareness environment. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*. ACSW Frontiers '03. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., pp. 153–158.

Mei, L., Chan, W.K. & Tse, T.H., 2008. A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues. In *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*. pp. 464 –469.

Moors, T., Mei, M. & Salim, A., 2008. Using short-range communication to control mobile device functionality. *Personal Ubiquitous Comput.*, 12(1), pp.11–18.

Morak, J. et al., 2012. Design and Evaluation of a Telemonitoring Concept Based on NFC-Enabled Mobile Phones and Sensor Devices. *Information Technology in Biomedicine, IEEE Transactions on*, 16(1), pp.17 –23.

Mowbray, M. & Pearson, S., 2009. A client-based privacy manager for cloud computing. In *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*. COMSWARE '09. New York, NY, USA: ACM, pp. 5:1–5:8.

Mukhtar, H., Belaïd, D. & Bernard, G., 2008. A model for resource specification in mobile services. In *Proceedings of the 3rd international workshop on Services integration in pervasive environments*. SIPE '08. New York, NY, USA: ACM, pp. 37–42.

Murao, K. et al., 2011. Evaluating Gesture Recognition by Multiple-Sensor-Containing Mobile Devices. In *Proceedings of the 2011 15th Annual International Symposium on Wearable Computers*. ISWC '11. Washington, DC, USA: IEEE Computer Society, pp. 55–58.

Murphy, M.M.L., 2011. *Android Programming Tutorials*, CommonsWare, LLC.

Nah, F.F.-H., Siau, K. & Sheng, H., 2005. The value of mobile applications: a utility company study. *Commun. ACM*, 48(2), pp.85–90.

Naqvi, S. & Riguidel, M., 2004. Security architecture for heterogeneous distributed computing systems. In *Security Technology, 2004. 38th Annual 2004 International Carnahan Conference on*. pp. 34 – 41.

Nielsen & Levy, 1994. Measuring usability: preference vs. performance. *Commun. ACM*, 37(4), pp.66–75.

NIST, 2011. A NIST Definition of Cloud Computing. Available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [Accessed April 14, 2012].

Nouali, N., Doucet, A. & Drias, H., 2005. A two-phase commit protocol for mobile wireless environment. In *Proceedings of the 16th Australasian database conference - Volume 39*. ADC '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., pp. 135–143.

Oh, J.R. et al., 2010. A novel site-specific interference analysis model for Wireless Personal Area Network. In *Microwave Conference Proceedings (APMC), 2010 Asia-Pacific*. pp. 1942 –1945.

Orlikowski, W.J. & Baroudi, J.J., 1991. Studying information technology in organizations: Research approaches and assumptions. *Information Systems Research*, 2, pp.1–28.

Osmani, F. & Slabbert, A., 2009. A scalable distributed security infrastructure for industrial control and sensor networks. In *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*. IWCMC '09. New York, NY, USA: ACM, pp. 84–89.

Palit, R. et al., 2011. Selection and execution of user level test cases for energy cost evaluation of smartphones. In *Proceedings of the 6th International Workshop on Automation of Software Test*. AST '11. New York, NY, USA: ACM, pp. 84–90.

Paniagua, C., Srirama, S.N. & Flores, H., 2011. Bakabs: managing load of cloud-based web applications from mobiles. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*. iiWAS '11. New York, NY, USA: ACM, pp. 485–490.

Parkkila, J. & Porras, J., 2011. Improving battery life and performance of mobile devices with cyber foraging. In *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*. pp. 91 –95.

Parviainen, M., Pirinen, T. & Pertilä, P., 2006. A speaker localization system for lecture room environment. In *Proceedings of the Third international conference on Machine Learning for Multimodal Interaction*. MLMI'06. Berlin, Heidelberg: Springer-Verlag, pp. 225–235.

Patel, P. et al., 2011. Towards application development for the internet of things. In *Proceedings of the 8th Middleware Doctoral Symposium*. MDS '11. New York, NY, USA: ACM, pp. 5:1–5:6.

Pather, S. & Remenyi, D., 2004. Some of the philosophical issues underpinning research in information systems: from positivism to critical realism. In *Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries*. SAICSIT '04. Republic of South Africa: South African Institute for Computer Scientists and Information Technologists, pp. 141–146.

Peltzer, D., 2003. *XML: Language Mechanics and Applications*, Addison Wesley.

Perkins, C.E., 1998. Mobile networking in the Internet. *Mob. Netw. Appl.*, 3(4), pp.319–334.

Petrie, H. & Power, C., 2012. What do users really care about?: a comparison of usability problems found by users and experts on highly interactive websites. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. CHI '12. New York, NY, USA: ACM, pp. 2107–2116.

Pohja, M., 2009. Server push with instant messaging. In *Proceedings of the 2009 ACM symposium on Applied Computing*. SAC '09. New York, NY, USA: ACM, pp. 653–658.

Qiu, L. et al., 2006. Troubleshooting wireless mesh networks. *SIGCOMM Comput. Commun. Rev.*, 36(5), pp.17–28.

Raento, M. et al., 2005. ContextPhone: a prototyping platform for context-aware mobile applications. *Pervasive Computing, IEEE*, 4(2), pp.51 – 59.

Rasche, A., Puhlmann, M. & Polze, A., 2005. Heterogeneous adaptive component-based applications with Adaptive.Net. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*. pp. 418 – 425.

Ratner, D. et al., 2001. Replication requirements in mobile environments. *Mob. Netw. Appl.*, 6(6), pp.525–533.

Ryan, C. & Gonsalves, A., 2005. The effect of context and application type on mobile usability: an empirical study. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*. ACSC '05. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., pp. 115–124.

Saha, D. & Mukherjee, A., 2003. Pervasive Computing: A Paradigm for the 21st Century. *Computer*, 36(3), pp.25–31.

Satyanarayanan, M., 1996a. Accessing information on demand at any location. Mobile information access. *Personal Communications, IEEE*, 3(1), pp.26 –33.

Satyanarayanan, M., 1996b. Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*. PODC '96. New York, NY, USA: ACM, pp. 1–7.

Satyanarayanan, M., 2001. Pervasive computing: vision and challenges. *Personal Communications, IEEE*, 8(4), pp.10 –17.

Schmidt, Beigl & Gellersen, 1998. There is more to Context than Location. *Computers and Graphics*, 23, pp.893–901.

Shaked, Y. & Wool, A., 2005. Cracking the Bluetooth PIN. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. MobiSys '05. New York, NY, USA: ACM, pp. 39–50.

Sohn, T. et al., 2005. Place-Its: A Study of Location-Based Reminders. In *In Ubicomp*. Springer, pp. 232–250.

Stender, M. & Ritz, T., 2006. Modeling of B2B mobile commerce processes. *International Journal of Production Economics*, 101(1), pp.128–139.

Stonebraker, M. et al., 1996. Mariposa: a wide-area distributed database system. *The VLDB Journal*, 5(1), pp.048–063.

Tanenbaum, M. & Van Steen, A., 2002. *Distributed Systems: Principles and Paradigms*, Prentice Hall.

Vaquero, L.M. et al., 2008. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39(1), pp.50–55.

Venkatesh, V., Ramesh, V. & Massey, A.P., 2003. Understanding usability in mobile commerce. *Commun. ACM*, 46(12), pp.53–56.

Vigfusson, Y. & Chockler, G., 2010. Clouds at the crossroads: research perspectives. *Crossroads*, 16(3), pp.10–13.

Waller, V. & Johnston, R.B., 2009. Making ubiquitous computing available. *Commun. ACM*, 52(10), pp.127–130.

Walsham, G., 1995. The Emergence of Interpretivism in IS Research. *Information Systems Research*, 6(4), pp.376–394.

Waluyo, A.B. et al., 2005. On Building a Data Broadcast System in a Wireless Environment. *International Journal of Business Data Communications and Networking*, 1(4), pp.15–37.

Weiser, M., 1991. The computer for the 21st century. *Scientific American*, 3(3), pp.3–11.

West, M.T., 2011. Ubiquitous computing. In *Proceedings of the 39th ACM annual conference on SIGUCCS*. SIGUCCS '11. New York, NY, USA: ACM, pp. 175–182.

Xiao, Y. et al., 2011. CasCap: cloud-assisted context-aware power management for mobile devices. In *Proceedings of the second international workshop on Mobile cloud computing and services*. MCS '11. New York, NY, USA: ACM, pp. 13–18.

Xu, Q. et al., 2011. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. IMC '11. New York, NY, USA: ACM, pp. 329–344.

Zhang, D. & Adipat, B., 2005. Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. *International Journal of Human-Computer Interaction*, 18(3), pp.293–308.

Zhang, D., Yang, L.T. & Huang, H., 2011. Searching in Internet of Things: Vision and Challenges. In *Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on*. pp. 201 –206.

Zhou, J. et al., 2010. Pervasive Service Computing: Visions and Challenges. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*. pp. 1335 –1339.

Appendix A: Hardware Devices

Android Device model	Released	CPU	RAM	WLAN	Bluetooth	Accel.	Gyro	Prox.	Compass
HTC Magic	2009	528MHz ARM 11	288MB	802.11 b/g	2.0	✓			✓
HTC Nexus One	2010	1GHz Scorpion	512MB	802.11 a/b/g	2.1	✓		✓	✓
HTC Evo 4G	2010	1GHz Scorpion	512MB	802.11 b/g	2.1	✓		✓	✓
HTC Flyer	2011	1.5GHz Scorpion	1GB	802.11 b/g/n	3.0	✓			✓
Samsung Galaxy Tab 7	2011	Dual-core 1.2 GHz	1GB	802.11 a/b/g/n	3.0	✓	✓	✓	✓
Samsung Galaxy Tab 10.1	2011	Dual-core 1GHz Cortex-A9	1GB	802.11 a/b/g/n	3.0	✓	✓	✓	✓

iOS Device model	Released	CPU	RAM	WLAN	Bluetooth	Accel.	Gyro	Prox.	Compass
iPhone 2G 	2007	412MHz ARM 11	128MB	802.11 b/g	2.0	✔		✔	
iPhone 3G 	2008	412MHz ARM 11	128MB	802.11 b/g	2.0	✔		✔	

Java ME Device model	Released	CPU	RAM	WLAN	Bluetooth
Nokia E61 	2005	220MHz Dual ARM 9	64MB	802.11 i/e/g	1.2
Nokia 6230i 	2005	*NA*	*NA*	*NA*	1.2
Nokia 6233 	2005	*NA*	*NA*	*NA*	2.0
Nokia 6680 	2005	*NA*	*NA*	*NA*	1.2
Sony Ericsson k600i 	2005	*NA*	*NA*	*NA*	2.0
Sony Ericsson W810i 	2006	*NA*	*NA*	*NA*	2.0

Windows Mobile Device model	Released	CPU	RAM	WLAN	Bluetooth
HTC Tytn II	2007	400MHz ARM 11	128MB	802.11 b/g	2.0
HTC MTeor	2006	Samsung 2442 300MHz	64MB		2.0
HTC S710	2007	200MHz ARM926EJ-S	64MB	802.11 b/g	2.0
Qtek 8020	2004	200MHz ARM926EJ-S	32MB		1.1
Qtek 9100	2005	200MHz ARM926EJ-S	64MB	802.11 b/g	2.0

Head Mounted Display

In experiment 3, *PainDroid*, we used a head mounted display to that enabled the user to describe the pain experience with a 3D model in a more realistic and interactive manner from the comfort of his/her home. The HMD glasses used was *Vuzix Wrap 920*, which is displayed in the image below. The glasses include twin high-resolution 640 x 480 LCD displays and 60Hz progressive scan update rate[45].

[45] http://www.vuzix.com/consumer/products_wrap920.html#specifications

Appendix B: Java ME Documentation (experiment 1)

Quick start

1. **Initial pairing between the phone and the computer.**

2. **Install phone application**
   - **a.** Transfer the application to the phone.
   - **b.** Install the jar file, and choose the program folder.

3. **Install the server application**
   - **a.** Copy the files (bluecove.jar, BTShareServer.jar and run server.bat) to a folder on the computer.

4. **Run the phone application**
   - **a.** Click on the BTShareMobile file in the programs folder.

5. **Search and save the Bluetooth device that contains the server**

6. **Start the server application**
   - **a.** Double click on BTShareServer.jar or use the run server.bat file.

7. **Connect to the server application from the Remote Control menu on the phone.**

8. **Accept the incoming connection on the server application**

General information, BTShareMobile

BTShareMobile is a Bluetooth Remote Control application. It enables the user to control a Bluetooth enabled computer from their mobile phone. It is created to be as flexible as possible, this means that the user can custom map each button and each button can hold several actions.

If you do experience any problems, for example that the application stops or the loading screen will not go away, hold the back button on your phone down for a few seconds and you can end the application from a menu that pops up.

It is important that you always turn off Bluetooth when you are finished using it. Also, if you have completed the initial pairing, turn off visible mode.

Requirements

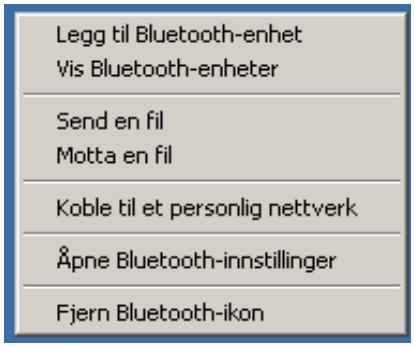
- Bluetooth enabled phone.
- 19 required buttons:
- Numeric keys.
- Left and right softbuttons.
- Directional keys.
- Minimum screen size: 208*208 pixels
- MIDP 2.0
- The JSR-082 api.

Initial pairing

Before you start the application you should do the initial pairing between the phone and computer. This is only done the first time you connect the two devices together.

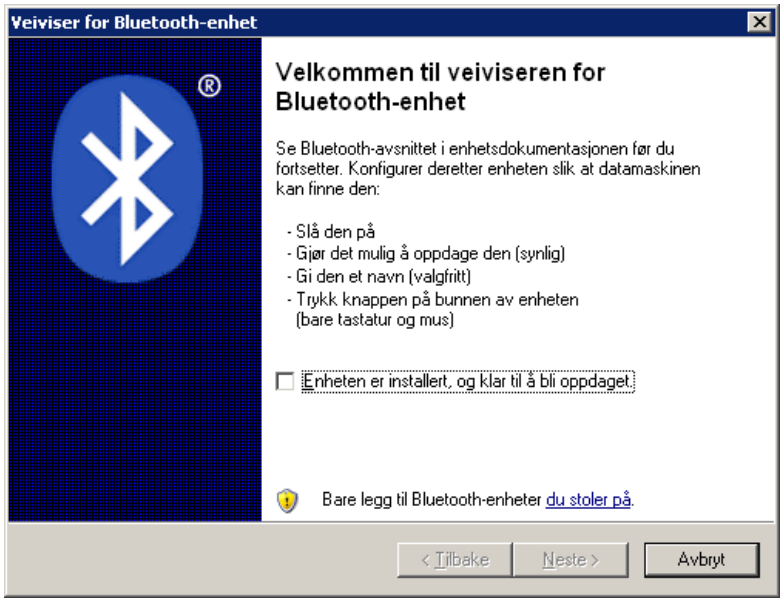
Make sure Bluetooth is enabled. On most IBM/Lenovo laptops Bluetooth is turned on by pressing the FN and F5 keys at the same time. In the dialog window choose Bluetooth “PÅ”. If you have a separate USB Bluetooth device, just plug this in and Bluetooth should be turned on automatically.

Bluetooth taskbar icon.



Bluetooth menu.

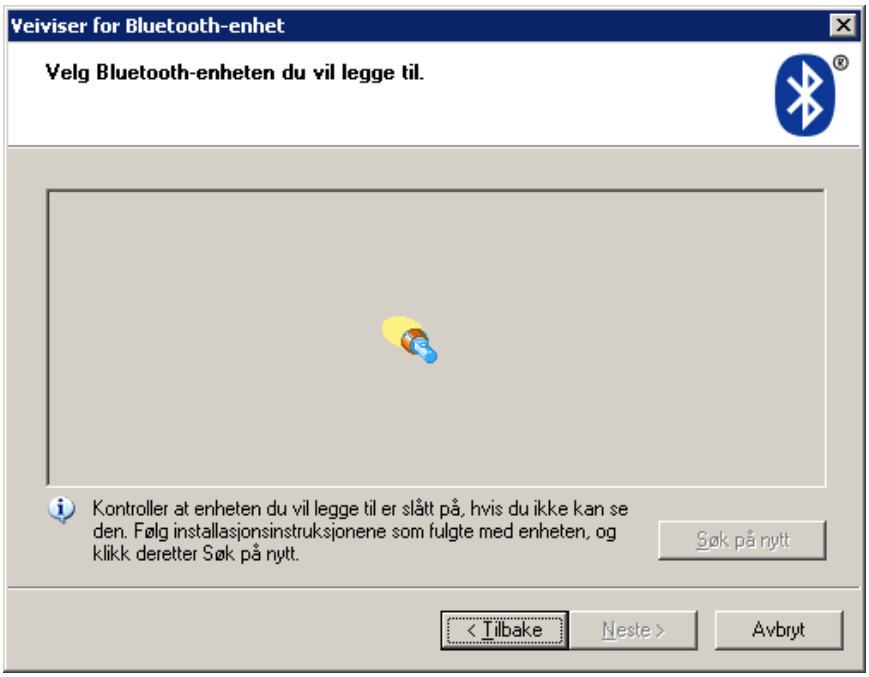
Right click on the Bluetooth symbol and select “Legg til Bluetooth-enheter”.



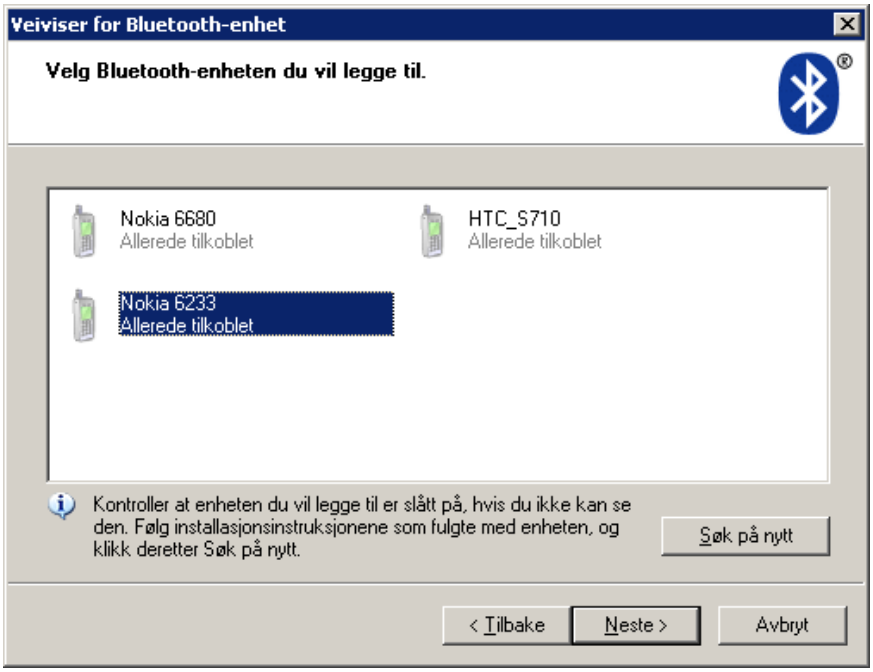
Bluetooth device discovery.

Tick the “Enheten er installert og klar til å bli oppdaget” box.

Set the phone to visible (in Bluetooth settings on the phone, set it to visible mode. In Norwegian phones this is often called “Synlighet” and then “Vis telefon”) and press “Neste”.

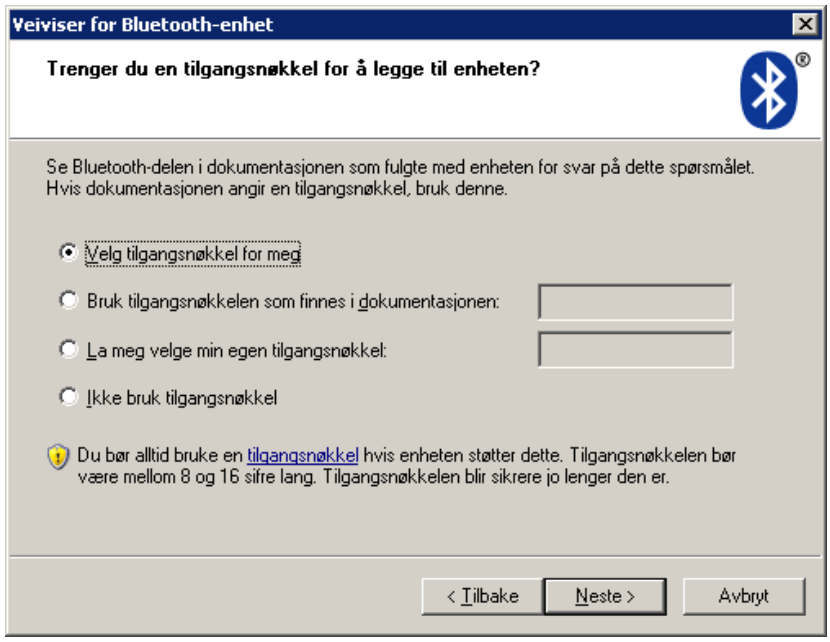


Bluetooth device search.



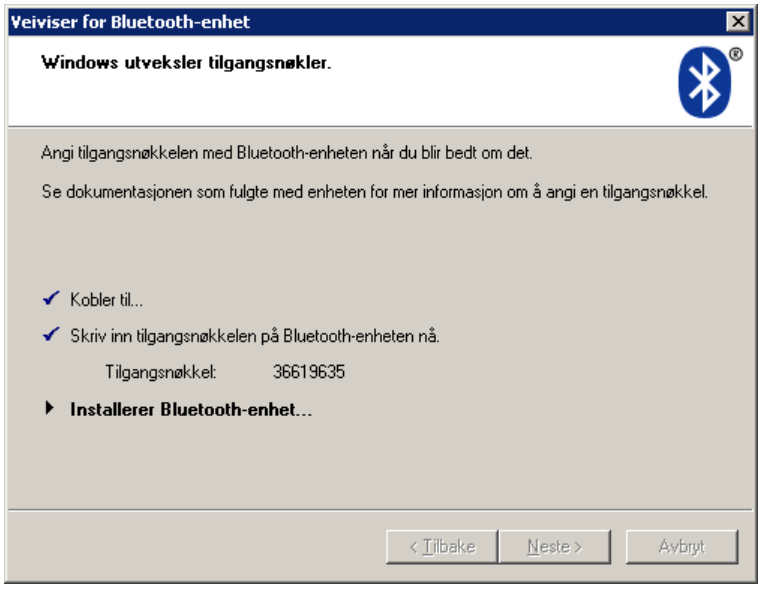
Bluetooth devices discovered.

Continue the installation process by searching for available Bluetooth devices and select the one you want to add and press “Neste”.



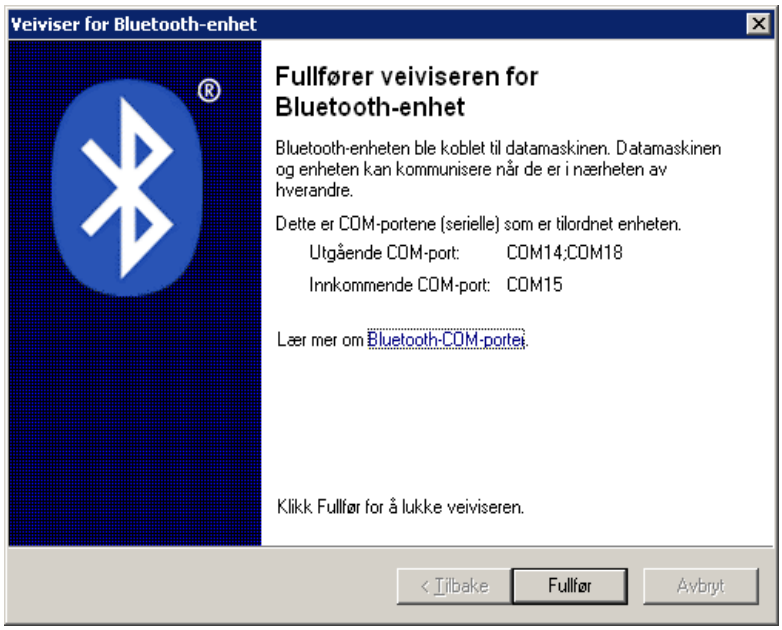
Bluetooth pin number.

It is important to use a passkey that is at least 4 digits long (it is recommended to use 8 digits). Simply select “Velg tilgangsnøkkel for meg” and press “Neste”.



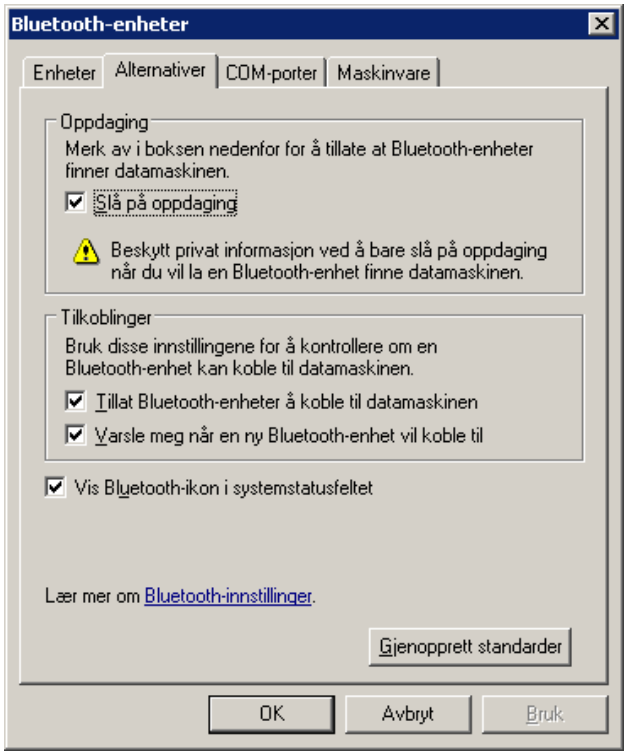
Bluetooth pin number.

You will the have to enter the same passkey on the phone (The application will not work if you do not use a passkey).



Bluetooth device discovery completed.

Press "Fullfør" to finish. When this is done the device will be installed and you are ready to use Bluetooth for communication between the devices.



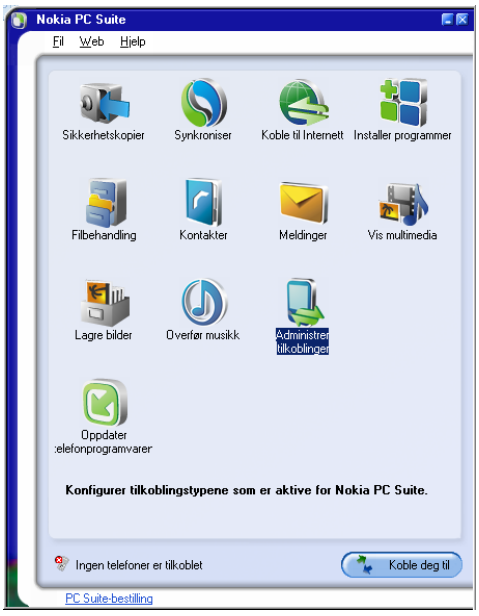
Bluetooth settings.

Right click the Bluetooth icon at the taskbar again and select ”Vis Bluetooth enheter”. Click on the “Alternativer” option and make sure “Slå på oppdaging” and “Tillat Bluetooth-enheter å koble til datamaskinen” is enabled. When the Bluetooth device is saved (in the phone application), make sure the “Slå på oppdaging” option is disabled.

Install the application

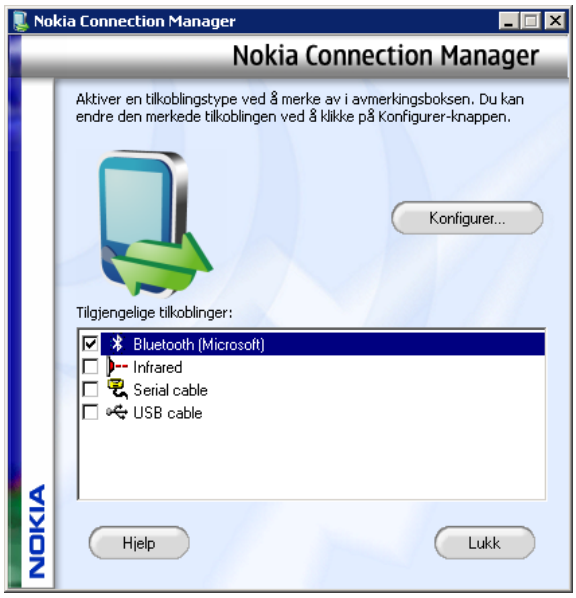
How you install the application depends on what model your phone is. There are usually two methods for installing the application. Both Sony Ericsson and Nokia have special software called PC Suite. This enables you to transfer files and install applications. The other method is simply to transfer the file to your phone and install it. To transfer the file directly will not work on all phones because not all models will recognize the jar-files.

Nokia PC Suite:



Nokia PC suite main menu.

Select “Administrer tilkoblinger”.

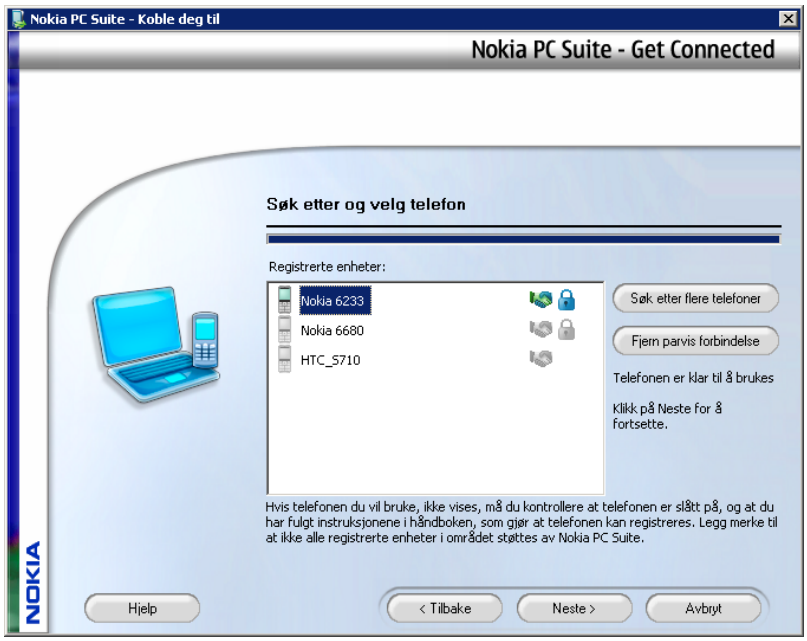


Nokia Connection Manager.

Choose the connection you have between the phone and computer, normally “Bluetooth” and then press “Konfigurer…”.

Nokia PC Suite – Get Connected.

Press “Neste”.

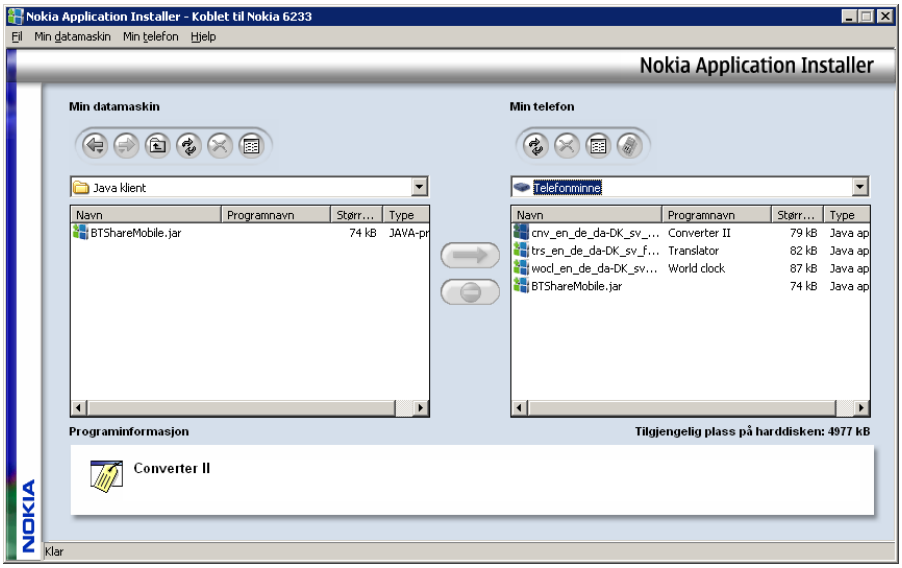


Nokia PC Suite search for Bluetooth device.

Select the phone you want to install the application on. Make sure there is a lock ( ) icon behind the phone name. This means you have a passkey for the connection.

Nokia PC Suite main menu.

Select ”Installer programmer”.



Nokia Application Installer.

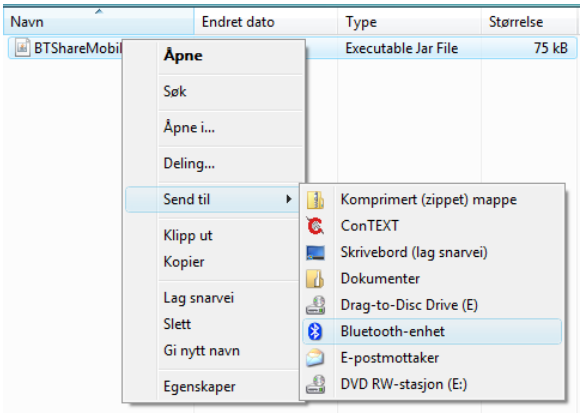
On the right hand side you can find the J2ME application (BTShareMobile.jar) that you want to install. Press the arrow in the middle to install it on the phone.

Sony Ericsson PC Suite:

Connect the phone and Sony Ericsson PC Suite and use the program's installation option. For more information use the Sony Ericsson website (www.sonyericsson.com).

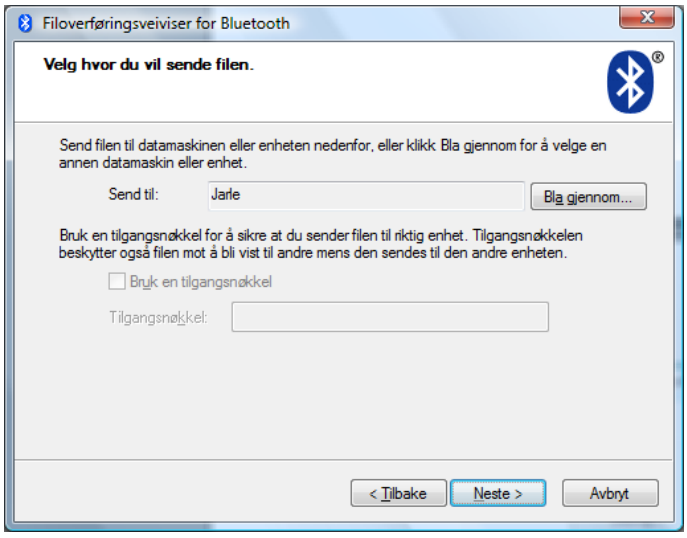
Transfer the file via Bluetooth:

Transfer the file to the phone and then install it.



File menu.

To send a file via Bluetooth you right click the file and then choose "Send til" -> Bluetooth-enhet.



File transfer.

Select the phone you want to send it to (in the "Send til" textbox) and press "Neste". Accept the file on the phone, and choose where you want to save it. Remember to turn on Bluetooth before you send the file.

If the phone does not recognize the jar-file, please use the software available for your phone.

Start the application

Always remember to enable Bluetooth before starting the application.

From the file explorer on the phone, find the folder programs (the folder you selected in the installation process). Simply open that folder and press the left softbutton to start the application.

Some phones may have a separate option for Programs under the main menu.

Nokia 6233 the application will be saved under the menu "Programmer -> samling".
Nokia 6680 BTShareMobile will be saved as an icon under the "Meny" screen, usually last.
Sony Ericsson w810i the application will be saved under "Programmer".

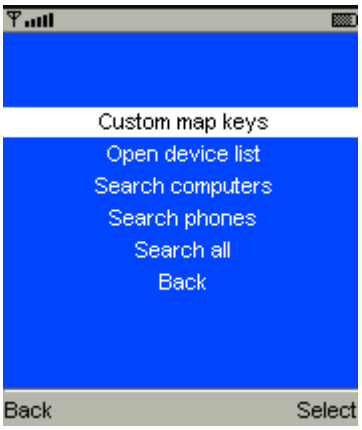
Uninstall the application

Use the file explorer to navigate to the folder where you saved the application (Usually the "Programmer" folder or similar). Select the BTShareMobile file, and press the "C" button or you can enter the file menu and select "Slett". Press "Ja" to the dialog box that asks you if you want to delete the file. If the application asks if you want to delete the user data from the application select yes.

Profile administrator

Main menu.

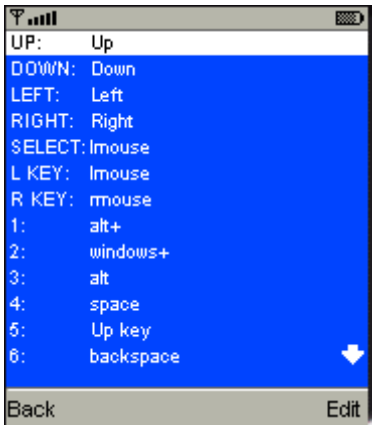
In the main menu you can select between 3 options (Remote Control, Profile admin and Exit) by using the directional keys and one of the softbuttons. To open the Profile administrator select "Profile admin" and press Open.



Profile admin.

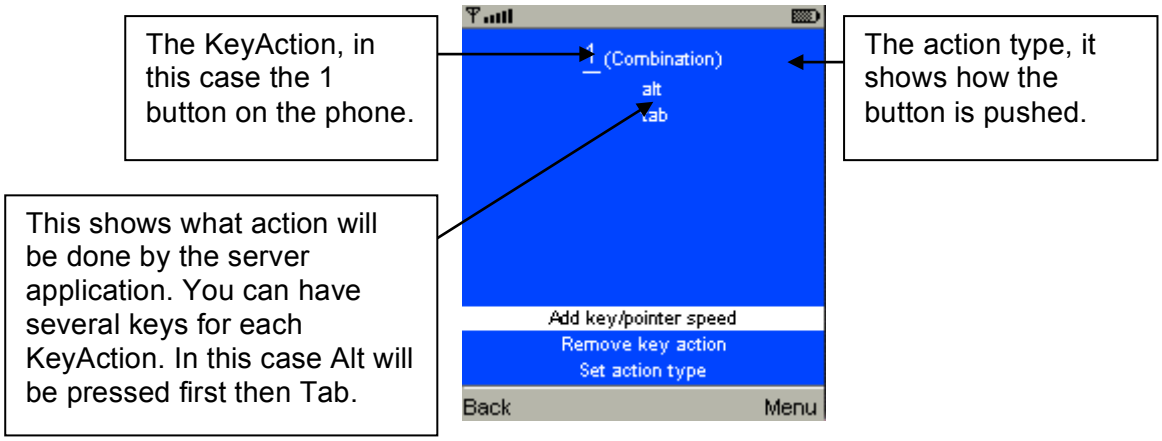
Custom map a key action

To open the Custom map keys menu simply press the “Custom map keys” option. You will then get to select the button you want to change and press “Edit” to change the key actions.



Custom map key menu.

When you have selected a button you want to change, the next screen will show what actions that are already saved on it and also what type of action.



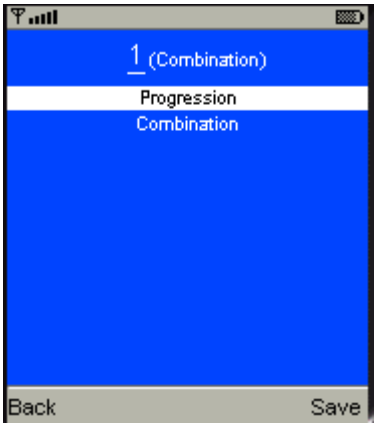
KeyAction men.

If it is a normal button (like in the picture) you can choose to add a new key. If the button is a mouse movement (Up, Down, Left or Right) you can set the pointer speed with the first option.

The second option is to remove a key action that is stored for this button.
The last option is to change the key action type. For a mouse movement button there is two options: Single and Press/Release.

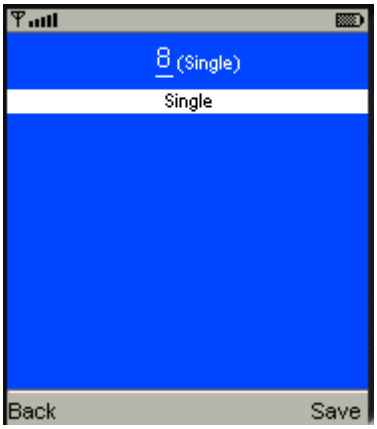
Action type menu.

For a normal button you will have the option to choose Progression or Combination. It is important to use Combination if you want to create keyboard shortcuts.



Action type menu.

If there is only one key action set to this button you will of course not have the option to set the type to progression or combination.



Action type menu.

Press save to store the key action type.

Key Action types:

- Single: The key will be pressed and released once.
- Combination: All the keys will be pressed and then released.
- Progression: The keys will be pressed and then released one by one.
- Press/Release: First key action from the phone will press the button, the next will release it. Especially useful for mouse movement and the possibility to select text.

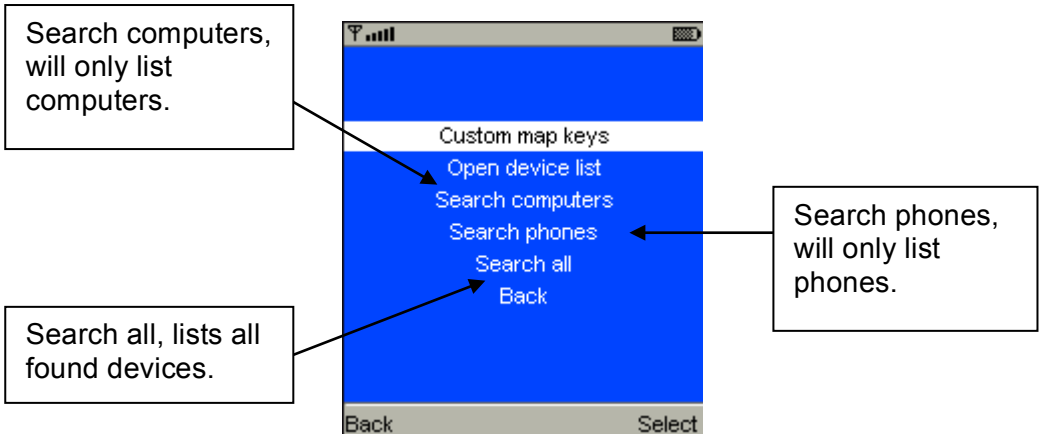
KeyAction menu.

When you are finished editing the KeyAction, press “Save” to store the changes in the database. The “Save” button will in most cases be in a submenu, in the screenshot this is called “Menu”. In Norwegian phones this is often called “Mer”.

Open Bluetooth device list

Select ”Open device list” in the Profile administrator menu. The entire list of saved devices will then be shown.

Search for Bluetooth devices



Profile admin.

If you are in a location with several devices and device types it can be a good idea to use a filter. When a search has been started a loading screen will appear until the search is completed.

If a device is new, not already saved, it will have the text (New) in front. All devices will be shown with the device name and address. If the name and address is too long for the screen, only the name will be shown. There is also a possibility where the device name isn't found, in this case only the address will be shown.

The user will have the option to save the selected device (simply click on save from the softbutton submenu, often called "Menu" or "Mer") and you can also view the details of the device. The details include device name, address, major-, minor- and service class.

Remember to set the Bluetooth settings on the other Bluetooth device to visible. After the device is saved turn off visible mode again. This is an easy security measure. Also when you do not use Bluetooth deactivate it.

Save a Bluetooth device

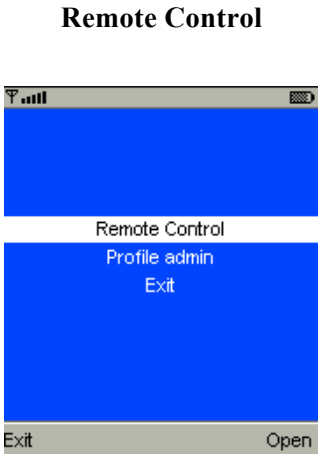
If a new device is found (must have the text (New) in front) it can be saved. From the softbutton menu select save. If the text "Device saved" is shown, the device is stored in the database. If the device already is saved, an error message will be shown.

The "Save" button might be located under a submenu (Often called "Menu" or "Mer" on Norwegian phones). This depends on the phone model.

Delete a saved Bluetooth device

From the Profile administrator select “Open device list”. Select the device you want to delete and from the softbutton menu select “Delete”. If successful the message “Device deleted” will appear. If an error occurs this will be shown to the user and the device will not be deleted.

The “Delete” button might be located under a submenu (Often called “Menu” or “Mer” on Norwegian phones). This depends on the phone model.



Main menu.

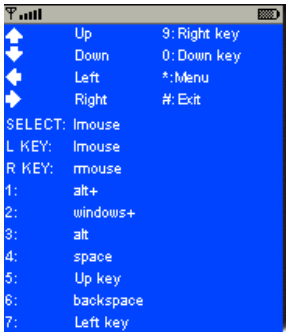
In the main menu select “Remote Control” and press open to start the Remote Control application.

Connect to the server application

In the main menu select Remote Control and press “Open” with the softbutton. A list of all the stored devices will then appear. Remember that you have to save the device before connecting to it (see section A8.3.3). Select the one you want to connect to and press “Connect”. If the application does not have the rights to establish a Bluetooth connection it will ask the user for permission. Press “Yes” to allow the Bluetooth connection.

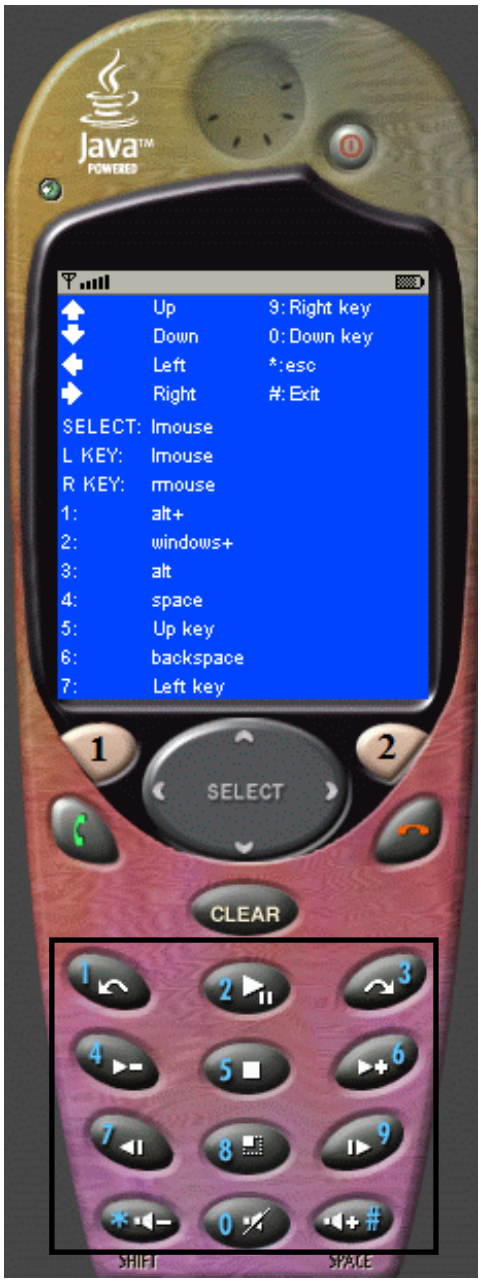
If the server application is up and running the phone will then ask to connect. Also, if this is the first time the application is connected the default key mappings will be stored in the database on the phone.

If the user accepts the connection the Remote Control GUI will appear. It contains information about all the key mappings on the phone. It shows a list of all the commands. If a command has a “+” after the name (like windows+) it means that it has more than one key mapped to it.



Remote Control user interface.

Default action keys



Softbuttons:
1. (Left softbutton): Left mousebutton.
2. (Right softbutton: Right mousebutton.

Directional buttons:
Up: Mouse up.
Down: Mouse down.
Left: Mouse left.
Right: Mouse right.

Select: Left mouse button, press/release. Makes it possible to select text.

Numeric keys:
1: Alt + Tab, switch programs.
2: Windows + D, minimize all windows.
3: Alt, useful to access toolbar options.
4: Spacebar.
5: Up arrow (Keyboard).
6: Backspace.
7: Left arrow (Keyboard).
8: Enter.
9: Right arrow (Keyboard).
***:** Esc.
0: Down arrow (Keyboard).
#: Exit.

Default key mappings.

General information, BTShare server

The server application handles all incoming connections. It is a very simple program that requires minimal input.

It is important that you always turn off Bluetooth when you are finished using it. Also, if you have completed the initial pairing, turn off visible mode.

Requirements

- Java Runtime Environment (JRE).
- Bluetooth enabled computer.
- Microsoft Bluetooth Stack.
- Windows XP Service Pack 2 and newer or Windows Vista.

Install the application

Copy the files (bluecove.jar, BTShareServer.jar and run server.bat) to a folder. It doesn't matter what folder you put them in, but they must be in the same folder.

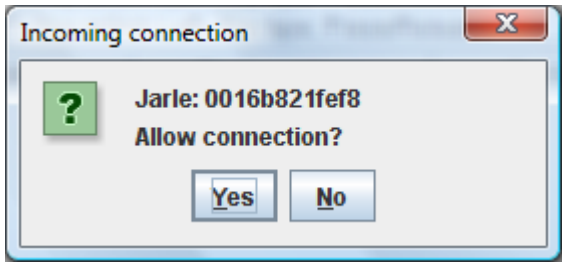
Run the application

In Windows Vista (If you have Java Runtime installed) it will be enough to click on BTShareServer. For Windows XP (also works in Windows Vista) you can start the program from the bat-file (run server.bat). If Nokia PC suite has been installed this might overwrite the jar-files, which can make the Nokia PC Suite open when you double click on the jar-file instead of the server application. If this is the case use the "run server.bat" to start the application.

If Bluetooth is enabled on the computer the server application should start. If there is a problem an error message will be shown and the application will be terminated.

Incoming connections

Each incoming connection must be accepted at the server. When a client tries to connect, the server application will receive a window with information about the client and also if you want to approve the connection.

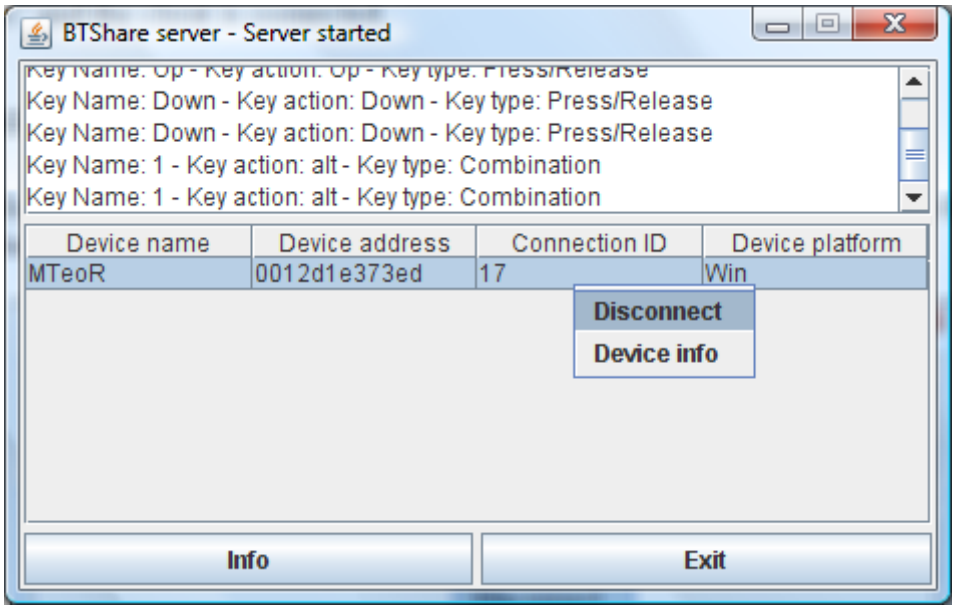


Incoming connection window.

Press "Yes" and the client is connected.

Disconnect a user

You can disconnect a user from the server application. This is done by clicking on the user and selecting disconnect.



Server application user interface.

List of keys

Here is a list of all the keys that are possible to map to a KeyAction (from the "Custom map keys" menu on the phone). Each key represents an action on the server application.

Key name on the phone	Key description, server action
0	Numbers from 0 to 9
1	
2	
3	
4	
5	
6	
7	
8	
9	
a	Letters from A to Z
b	
c	
d	
e	
f	
g	
h	
i	
j	
k	
l	
m	
n	
o	
p	
q	
r	
s	
t	
u	
v	
w	
x	
y	
z	
Up key	The up arrow.
Down key	The down arrow.
Left key	The left arrow.
Right key	The Right arrow.
Up	Mouse movement Up
Down	Mouse movement Down
Left	Mouse movement Left
Right	Mouse movement Right

lmouse	Left mouse button
rmouse	Right mouse button
ctrl	Ctrl-key
alt	Alt-key
tab	Tab-key
enter	Enter-key
space	Space-key
backspace	Backspace-key
shift	Shift-key
esc	Esc-key
windows	Windows-key

List of keys.

With combinations of these you can create several shortcuts. Remember to set the keyActionType to Combination when creating shortcuts. For example when creating the shortcut for minimizing all windows, simply add the "windows" button first, then add "d" and set the action type to combination. At the end save the KeyAction object.

Here is a short list of a few keyboard shortcuts for Windows:

Keys	Action
Alt + Tab	Switch between programs
Alt + Spacebar	Open the window's System menu
Ctrl + P	Open the print menu
Alt + Esc	Switch between open applications on the taskbar.
Windows + E	Open Windows explorer
Windows + D	Minimize all windows
Windows + R	Open the Run dialog
Windows + F	Open the search window

Windows shortcut list.

Appendix C: Windows Mobile Documentation (experiment 1)

Quick start

1. **Initial pairing between the phone and computer.**

2. **Install the phone application.**
   - **a.** Transfer the file or use the install option in ActiveSync.
   - **b.** Install the BTShareMobile.cab file.

3. **Install the server application.**
   - **a.** Copy the files (bluecove.jar, BTShareServer.jar and run server.bat) to a folder on the computer.

4. **Run the phone application.**
   - **a.** Enter the start menu and select the BTShareMobile icon.

5. **Start the server application.**
   - **a.** Double click on BTShareServer.jar or use the run server.bat file.

6. **Connect to the server application from the Remote Control menu on the phone.**

7. **Accept the incoming connection on the server application**

General information, BTShareMobile

BTShareMobile is a Bluetooth Remote Control application. It enables the user to control a Bluetooth enabled computer from their mobile phone. It is created to be as flexible as possible, this means that the user can custom map each button and each button can hold several actions.

It is important that you always turn off Bluetooth when you are finished using it. Also, if you have completed the initial pairing, turn off visible mode (Called discoverable in Windows Mobile 5.0).

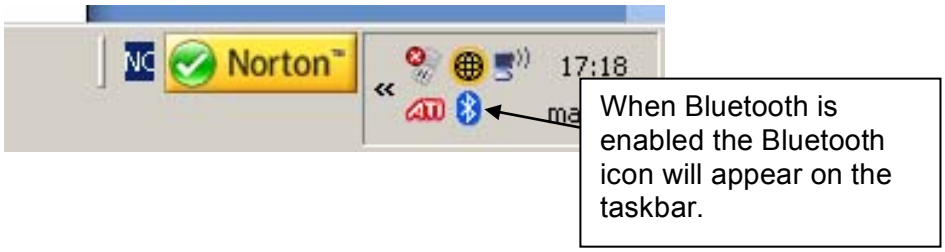
Requirements

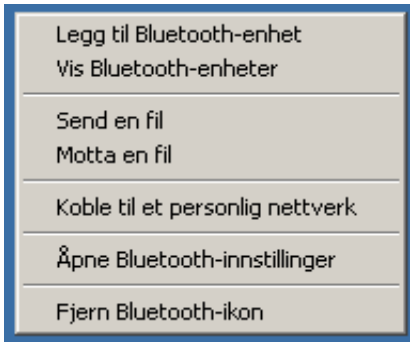
- Bluetooth enabled phone.
- 19 required buttons:
- Numeric keys.
- Left and right softbuttons.
- Directional keys.
- Windows Mobile 5.0 or newer.
- .Net Compact Framework 2.0 or newer.
- Microsoft Bluetooth stack.

Initial pairing

Before you start the application you should do the initial pairing between the phone and computer. This is only done the first time you connect the two devices together.

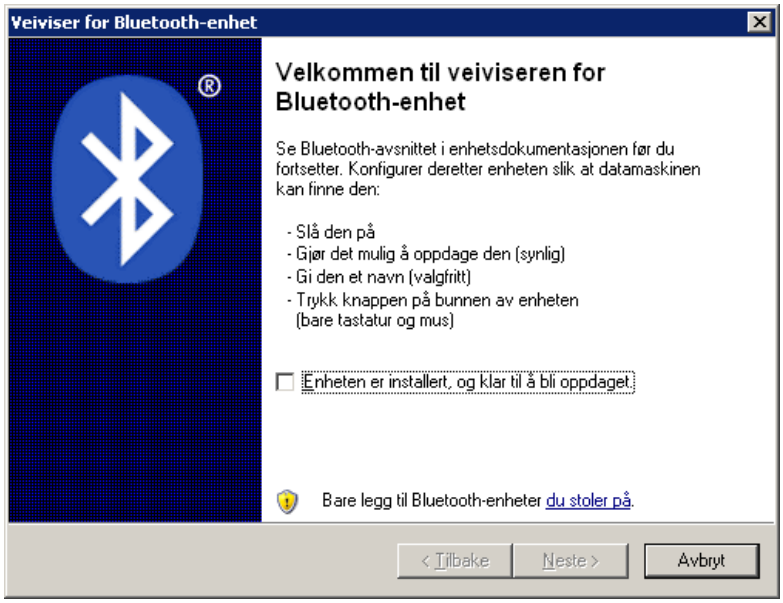
Make sure Bluetooth is enabled. On most IBM/Lenovo laptops Bluetooth is turned on by pressing the FN and F5 keys at the same time. In the dialog window choose Bluetooth “PÅ”. If you have a separate USB Bluetooth device, just plug this in and Bluetooth should be turned on automatically.





Bluetooth menu.

Right click on the Bluetooth symbol and select "Legg til Bluetooth-enheter".

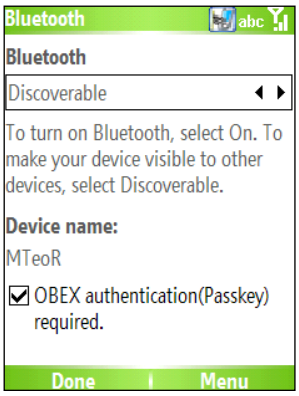


Bluetooth device discovery.

Tick the "Enheten er installert og klar til å bli oppdaget" box.

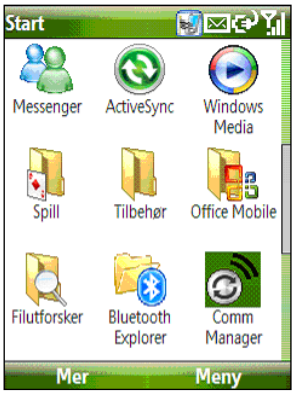
Set the phone to visible in Windows Mobile 5.0

Open the "Comm manager" from the start menu and select "Settings" and then "Bluetooth settings". In the Bluetooth settings menu, set Bluetooth to "Discoverable". Remember to either turn Bluetooth off (if you are not using it) or set it to "On" instead of "Discoverable" when you are finished with the initial pairing.



Windows Mobile 5.0 Bluetooth settings.

Set the phone to visible in Windows Mobile 6.0

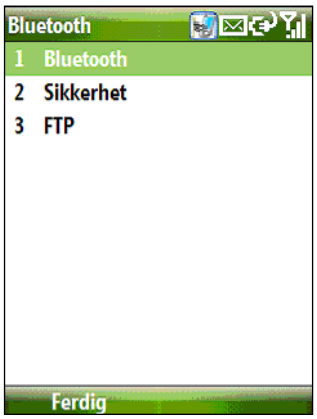


Windows Mobile 6.0 start.

In Windows Mobile 6.0 enter the “Comm Manager” from the start menu.

Windows Mobile 6.0 Bluetooth settings.

Press “Instillinger” and select “Bluetooth”.

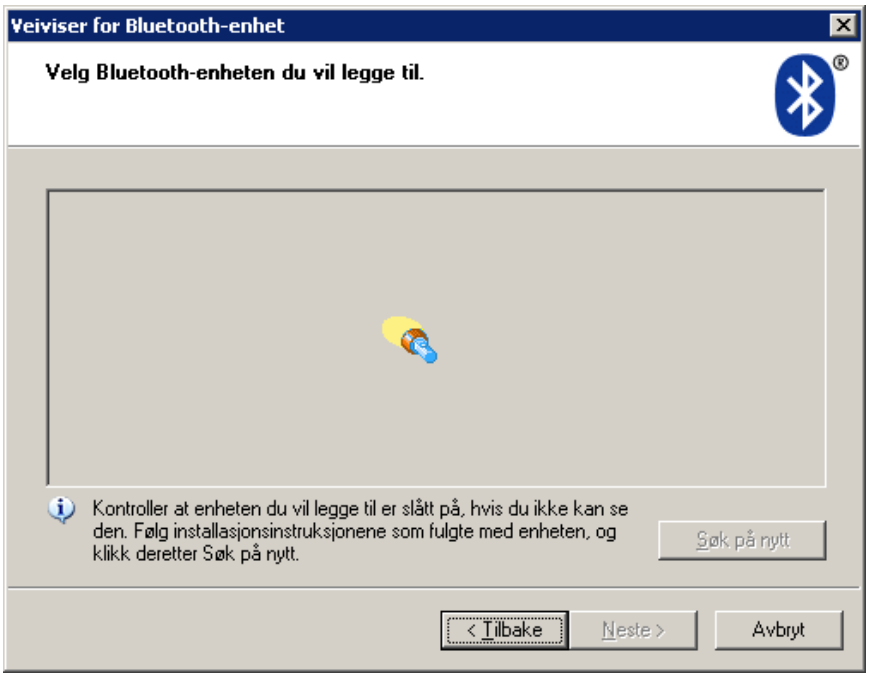


Windows Mobile 6.0 Bluetooth settings.

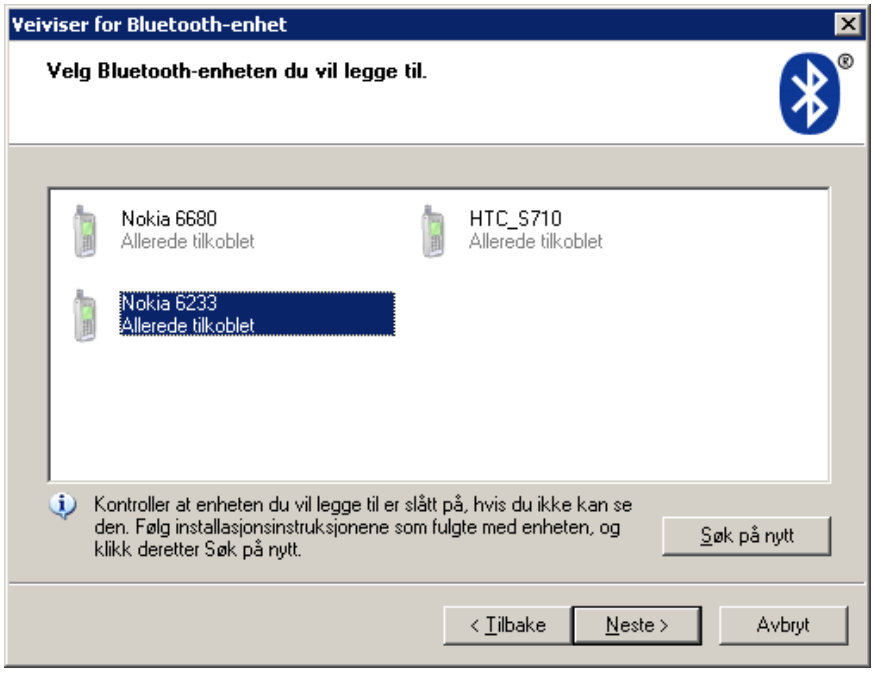
Select “Bluetooth” again.

Windows Mobile 6.0 Bluetooth settings.

Press “Meny” and then choose “Slå på synlighet”.

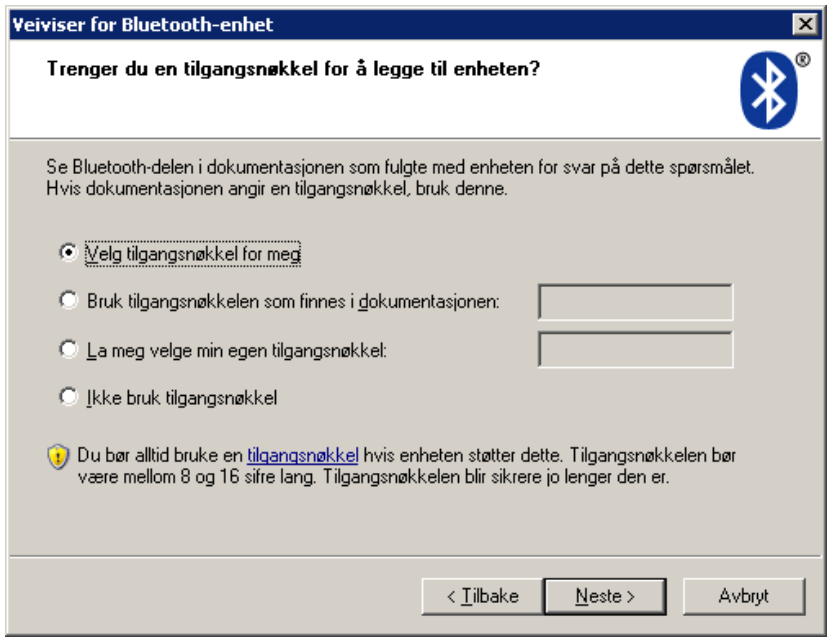


Bluetooth device search.



Bluetooth devices discovered.

Continue the installation process on the computer by searching for available Bluetooth devices and select the one you want to add and press “Neste”.

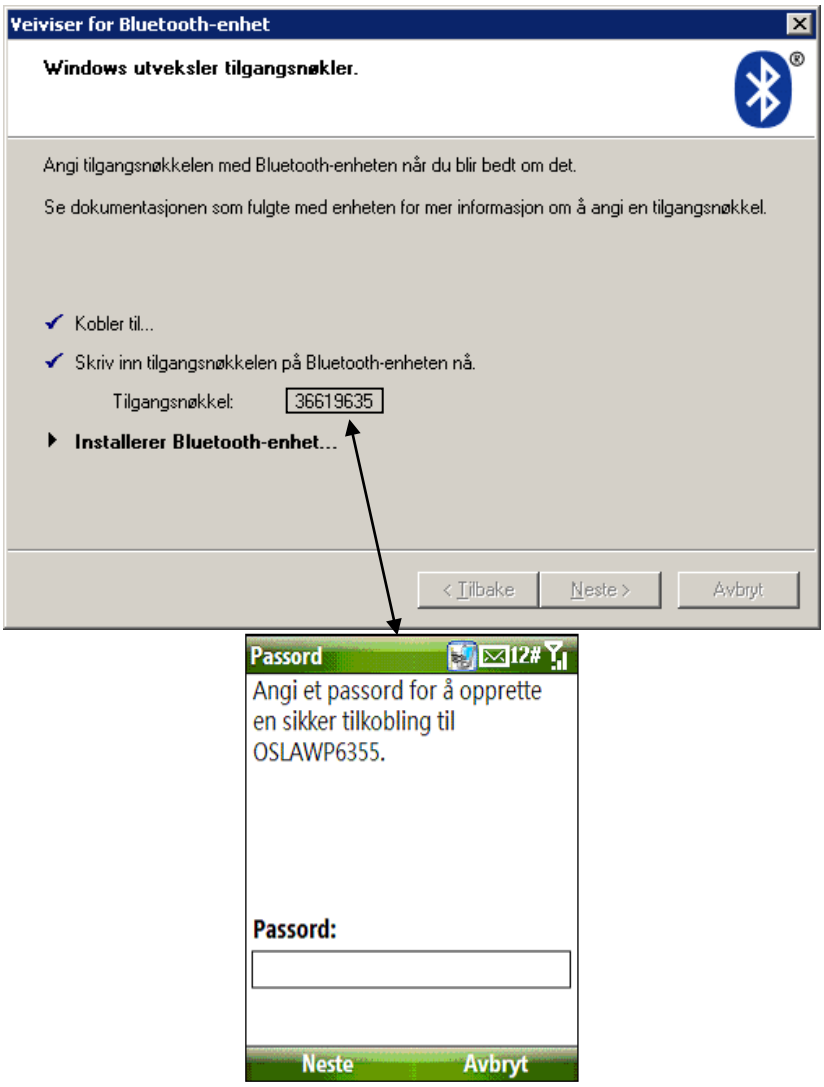


Bluetooth pin number.

It is important to use a passkey that is at least 4 digits long (it is recommended to use 8 digits). Simply select “Velg tilgangsnøkkel for meg” and press “Neste”.

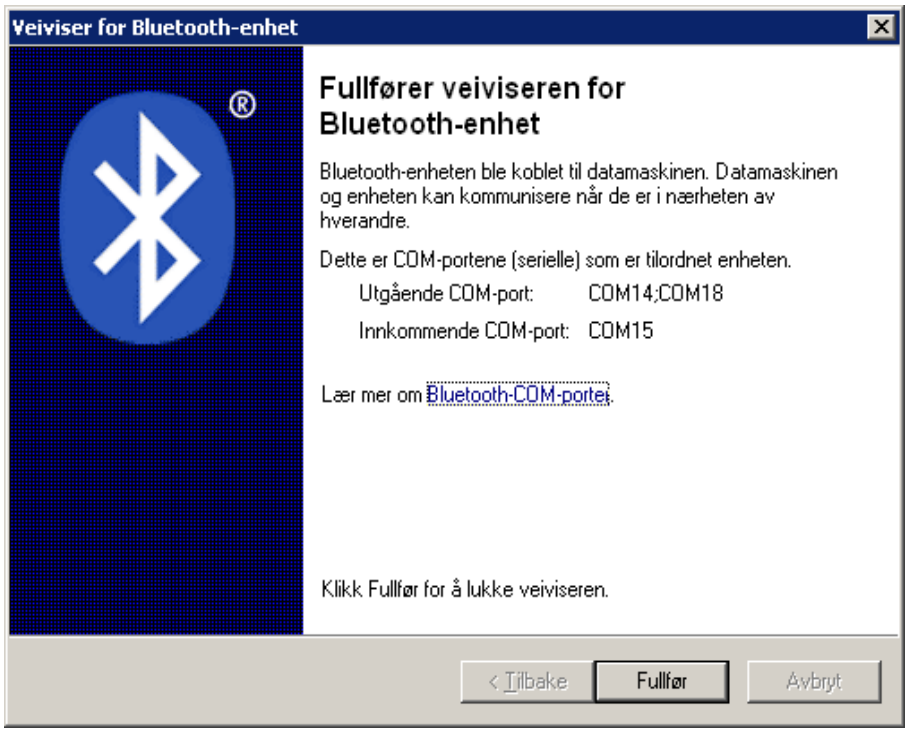
Windows Mobile Bluetooth connection.

On the phone you will then get a new window asking if you want to connect to the computer. Press “Ja”.



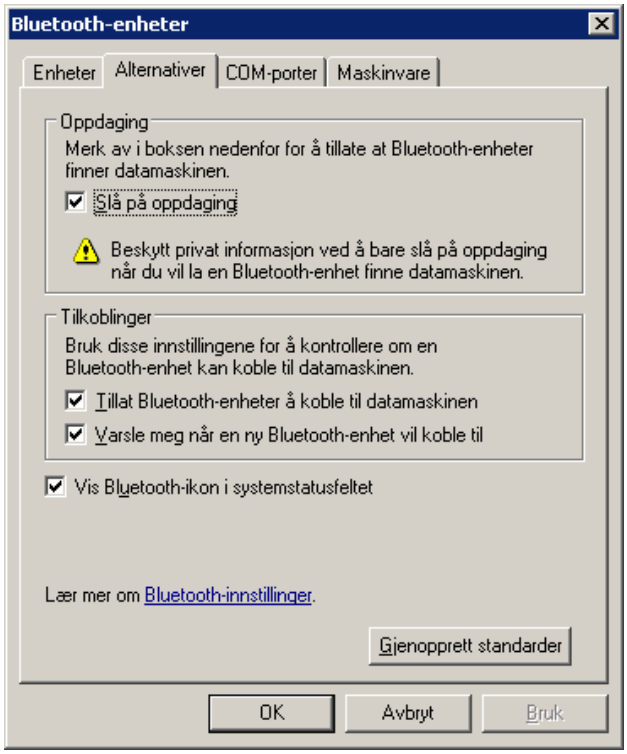
Bluetooth pin number.

You will the have to enter the passkey on the phone (The application will not work if you do not use a passkey).



Bluetooth device discovery completed.

Press “Fullfør” to finish. When this is done the device will be installed and you are ready to use Bluetooth for communication between the devices.



Bluetooth settings.

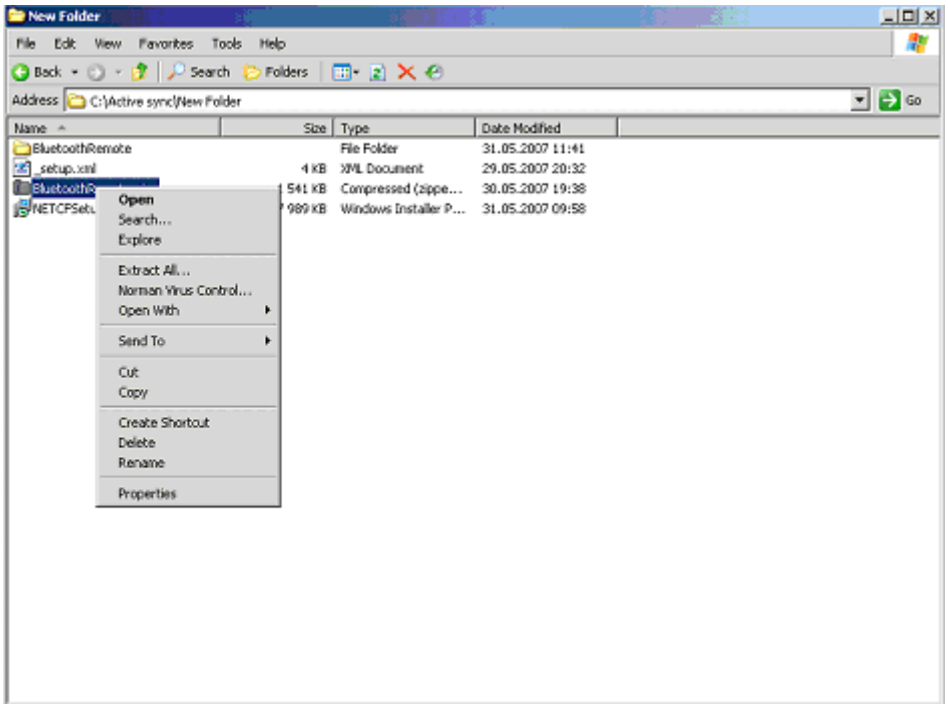
Right click the Bluetooth icon at the taskbar again and select ”Vis Bluetooth enheter”. Click on the “Alternativer” option and make sure “Slå på oppdaging” option is disabled when you are finished with the initial pairing.

Install the application

The installation only requires that you copy the BTShareMobileSetup file to your phone and run the installation file. After copying the file simply locate it on the phone and push it to start the installation. In this section there is a more detailed explanation of the installation process when using either ActiveSync or Windows Mobile Device Center.

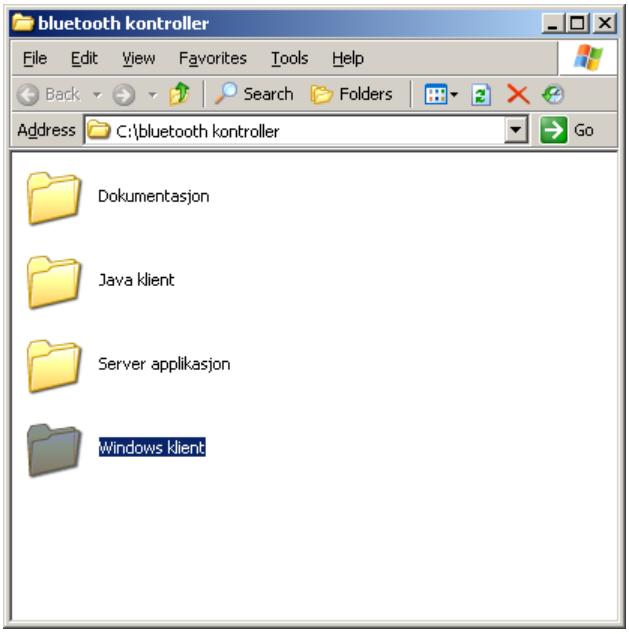
If you do get an error while installing saying that you need a newer version of the .Net framework you will have to download and install the new framework before you can use the application (this is only the case for certain Windows Mobile 5.0 devices). Visit the following links for more information:
http://msdn2.microsoft.com/en-us/netframework/aa497273.aspx
http://www.microsoft.com/downloads/details.aspx?familyid=9655156b-356b-4a2c-857c-e62f50ae9a55&displaylang=en

General installation:



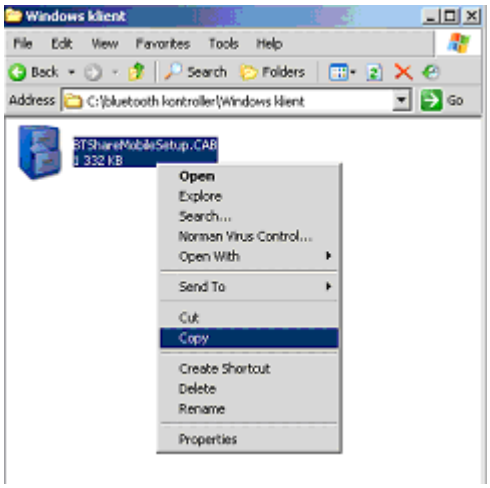
General installation.

Unzip the file to a folder. Select Extract All to unzip the file and follow the on-screen instructions.



General installation

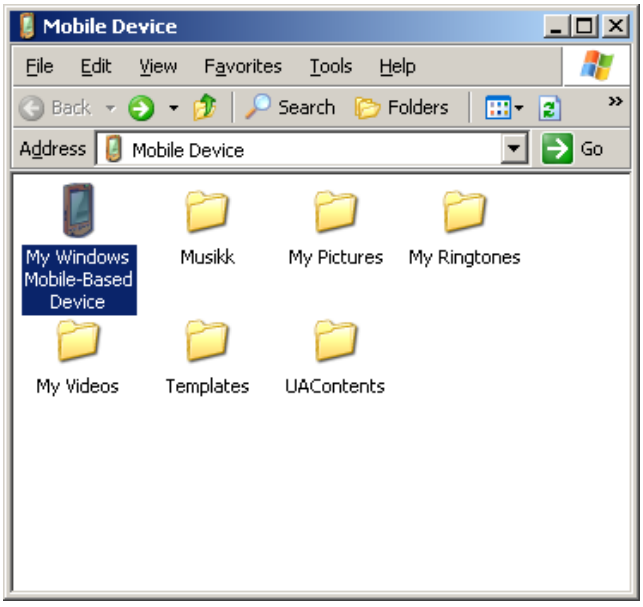
Open the unzipped folder and then open the “Windows klient” folder.



Copy Windows Mobile installation file.

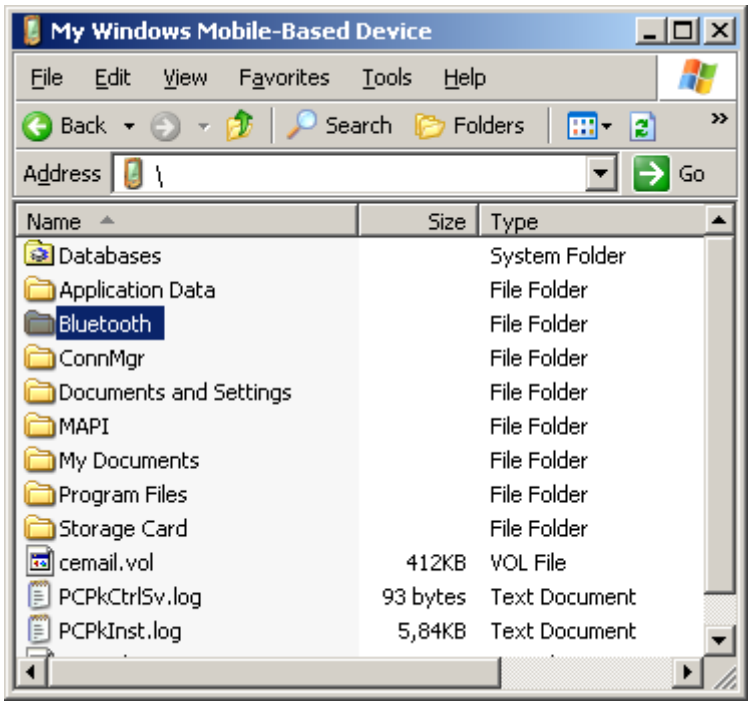
Copy the BTShareMobileSetup file to the phone. With Windows Mobile this is normally done by using either ActiveSync or Windows Mobile Device Center.

ActiveSync (Windows XP):



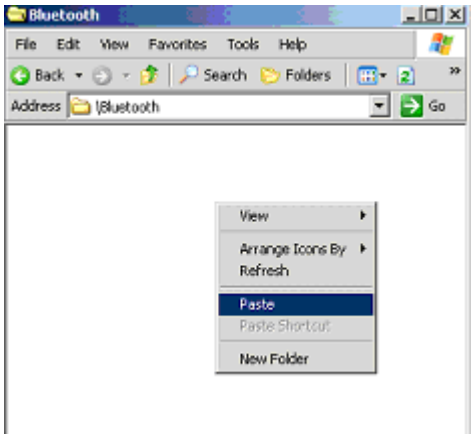
Mobile device menu.

Open the Mobile Device.



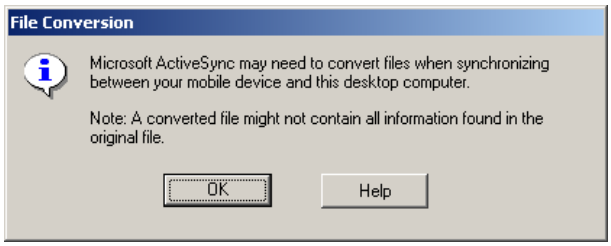
Mobile device content.

Create a new folder on the phone.



Add installation file.

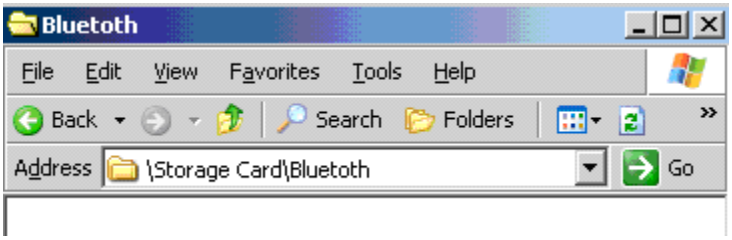
Paste the BTShareMobileSetup to the folder. The file you copied from the “Windows klient” folder on the computer.



File conversion.

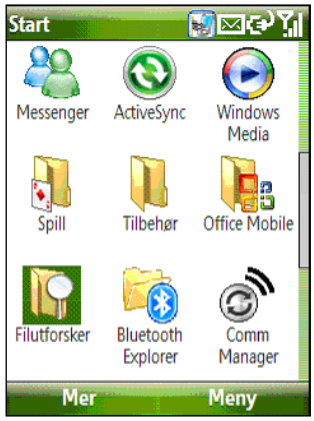
Press ”OK”.

If your phone's storage is full, copy the install file (BTShareMobileSetup) to the storage card.



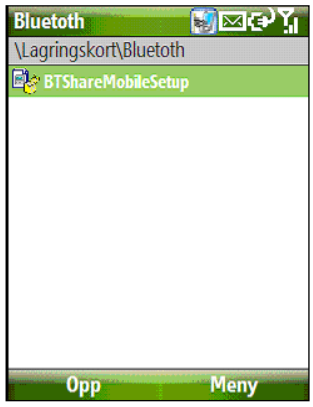
Storage card content.

You will now have to locate the file you just copied on the phone. Open the start menu and press the "Filutforsker" button.



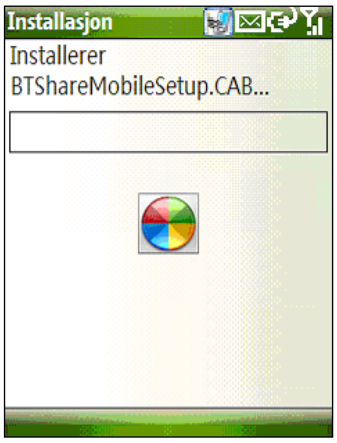
Windows Mobile start menu.

Use the file explorer to find the folder you saved the file in



Windows Mobile file explorer.

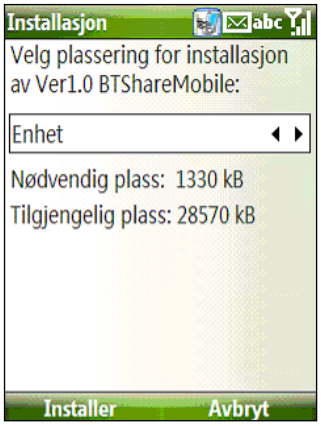
Press the select button (Press down the directional joystick) to start the installation.



Installation process.

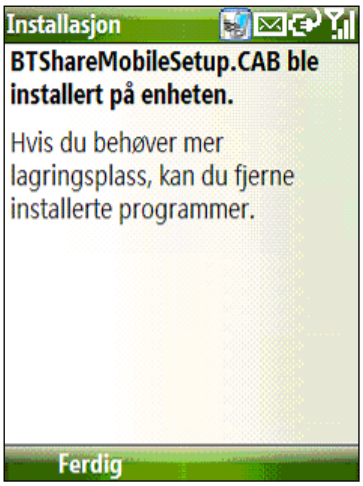
Installation process.

Press “Ja” if you get a dialog window asking if you want to continue.



Installation process.

You will then have to select where you want to store the application. Choose “Enhet” and make sure you have enough storage space.



Installation completed.

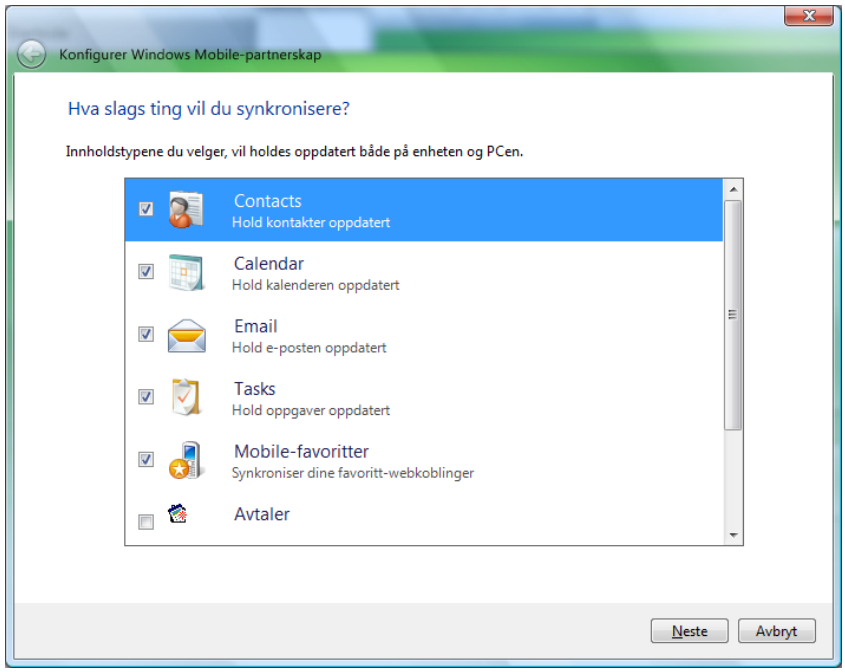
The application will now be stored under the "Program Files" folder.

Windows Mobile Device Center (Windows vista):

Open ActiveSync on the phone and select Connect via Bluetooth. Wait a few seconds for the devices to connect and open the Windows Mobile Device Center. It is important that the initial pairing is done before you try to connect to Windows Mobile Center.

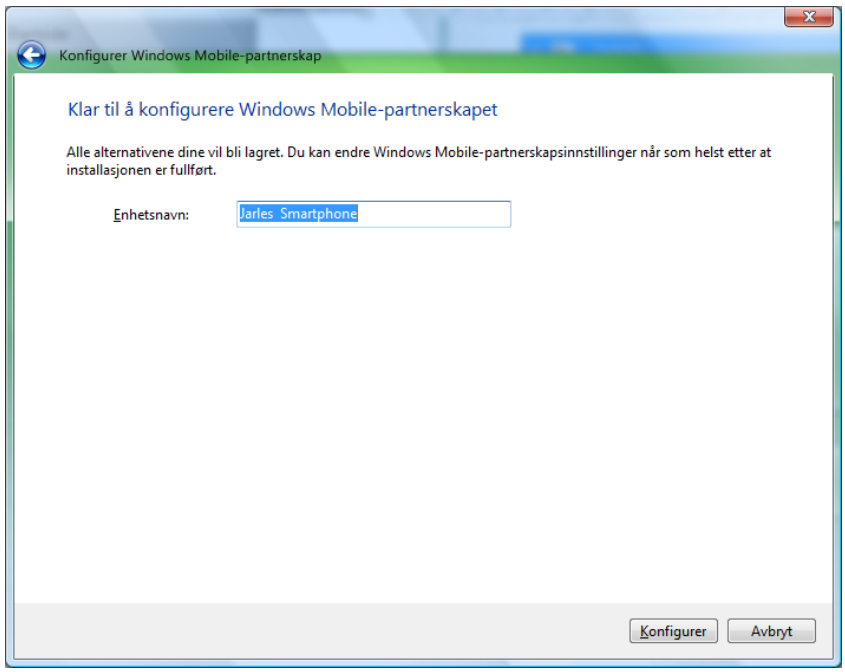
Windows Mobile Device Center.

Select "Konfigurer enheten" and then select what you want to synchronize.



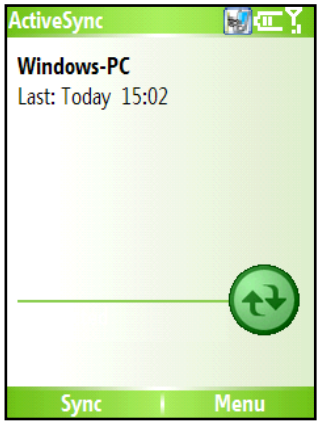
Windows Mobile Device Center.

Press “Neste” and write what you want to call the device.



Windows Mobile Device Center.

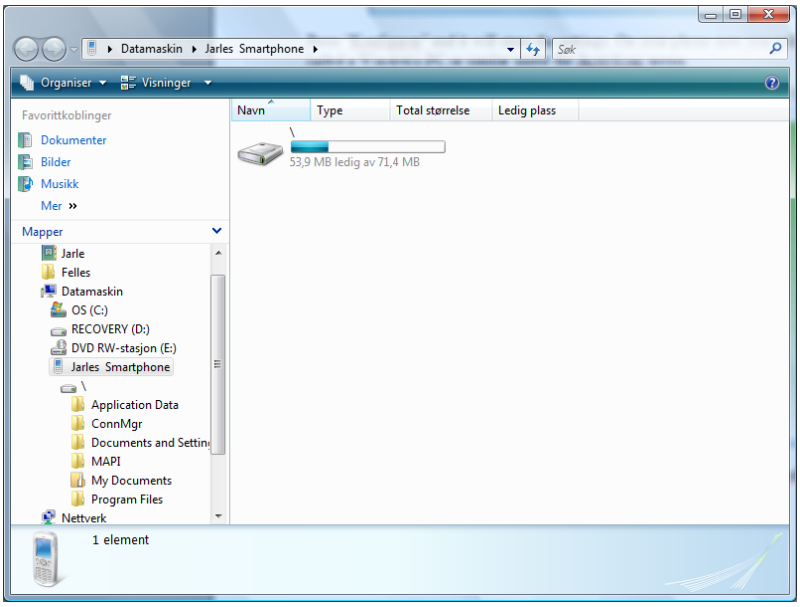
Press “Konfigurer” and it will store the settings. On your phone now you will have something called a Windows-PC or similar under the ActiveSync menu.



Windows Mobile ActiveSync.

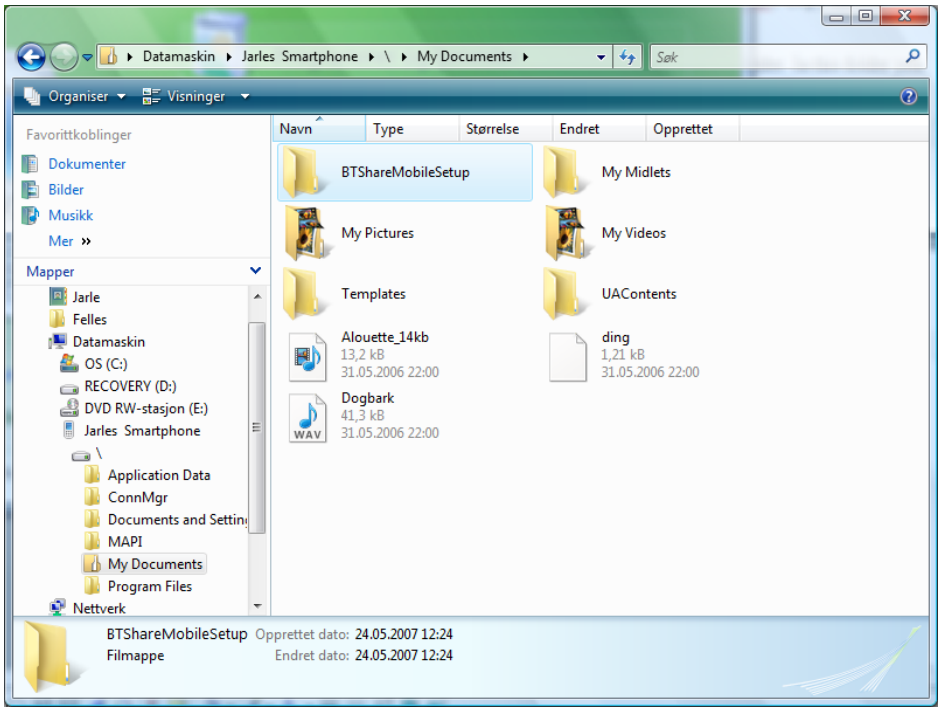
Windows Mobile Device Center.

To install the application open “Filbehandling” and select “Søk i innholdet på enheten”.



Smart phone storage.

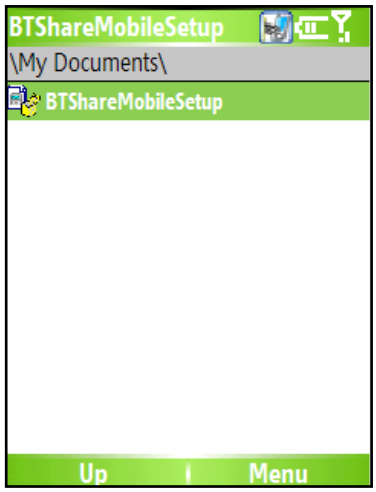
This opens a new window with the phones storage space. Double click the hard disk and select MyDocuments.



Smart phone storage.

Under MyDocuments create a new folder (for example call it BTShareMobileSetup). Paste the install file (BTShareMobileSetup) to this folder. The file you copied from the “Windows klient” folder on the computer.

On the phone, use the file explorer to find the BTShareMobileSetup folder. In this folder you will see the file you just transferred.



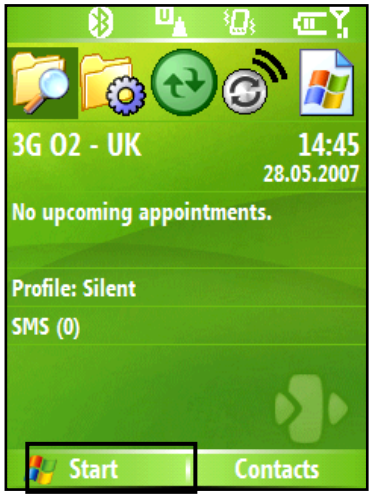
File explorer.

Use the select button (Press down the directional joystick) to start the installation. Press "Yes" if you get a dialog window asking if you want to continue.

Mobile phone application installation.

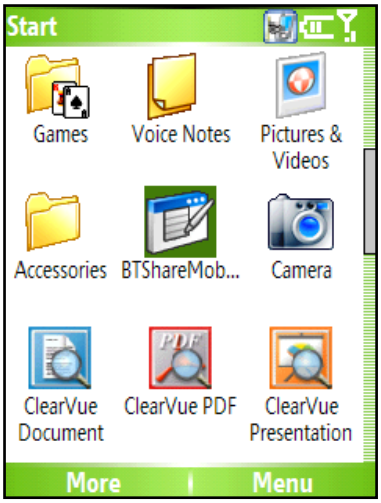
The application will now be stored under the "Program Files" folder.

Start the application



Windows Mobile main menu.

Press the "Start" button on the main screen.

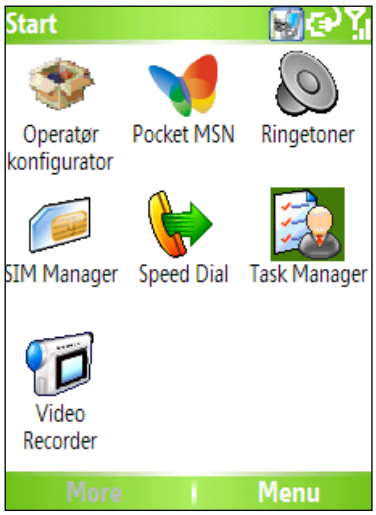


Windows Mobile start menu.

Navigate to the BTShareMobile icon and then select it by pushing the directional key.

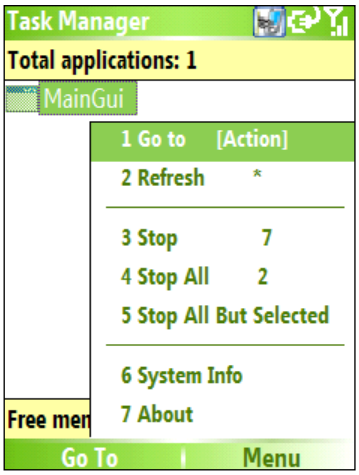
Do not leave the application running if you are not using the phone. Windows Mobile will minimize the application if you leave it running for a while without pressing any buttons.

If the application has been minimize you can open it again by pressing the BTShareMobile icon or by entering the "Task Manager" from the start menu.



Windows Mobile start menu.

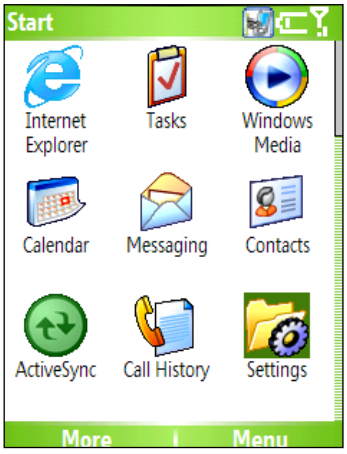
The BTShareMobile application will be shown as “MainGui”.



Task Manager.

Select the “MainGui” icon, and press menu. “Go to” starts the application again, or you can exit the application by pressing “Stop”. If you have any problems, like the application not responding or the loading screen will not go away, use the “Stop” feature in the “Task Manager” to end the application.

Uninstall the application

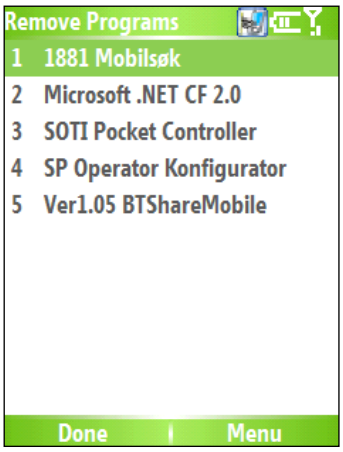


Windows Mobile start menu.

Enter the Start menu, and find the “Settings” option.

Settings.

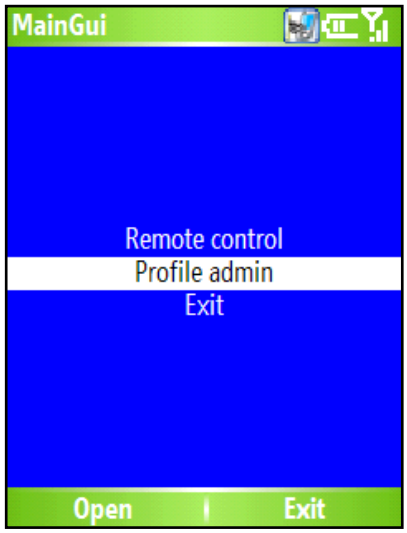
Press the “Remove Programs” option.



Remote Programs.

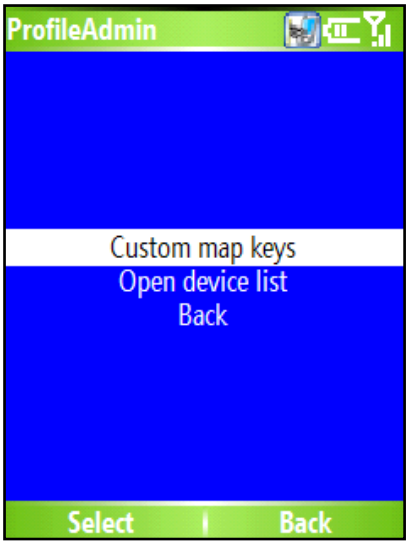
Find the program called “Ver1.05 BTShareMobile” and press “Menu” -> “Remove”. This will remove all the files that have been installed with the application.

Profile administrator



Remote Control main menu.

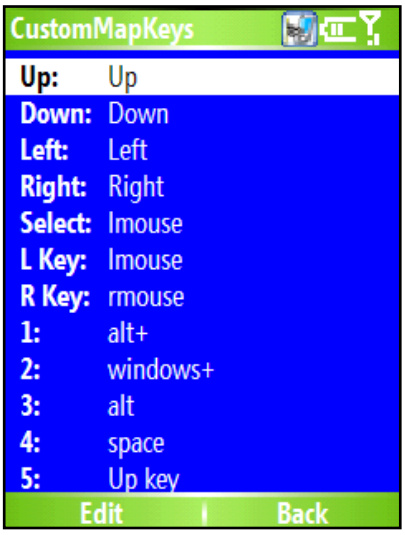
In the main menu you can select between 3 options (Remote Control, Profile admin and Exit) by using the directional keys and one of the softbuttons. To open the Profile administrator select “Profile admin” and press Open.



Main menu.

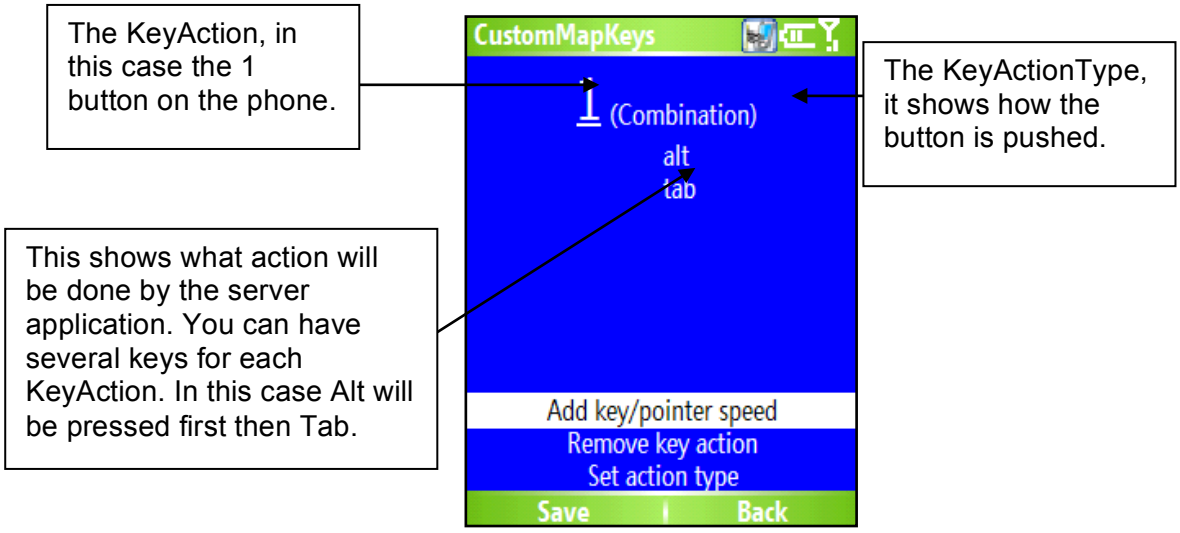
Custom map a key action

To open the Custom map keys menu simply press the “Custom map keys” option. You will then get to select the button you want to change and press “Edit” to change the key actions.



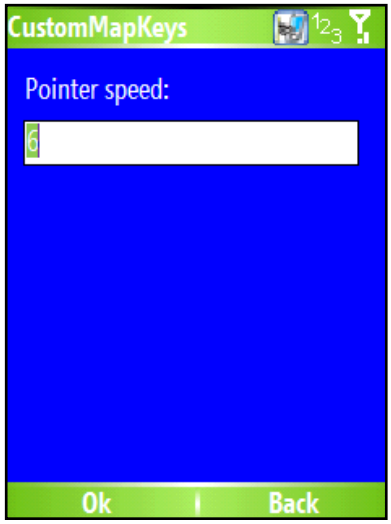
Custom map keys menu.

When you have selected a button you want to change, the next screen will show what actions that are already saved on it and also what type of action.



KeyAction menu.

If it is a normal button (not a mouse movement) you can choose to add a new key. The second option is to remove a key action that is stored for this button.
If the button is a mouse movement (Up, Down, Left or Right) you can set the pointer speed with the first option.

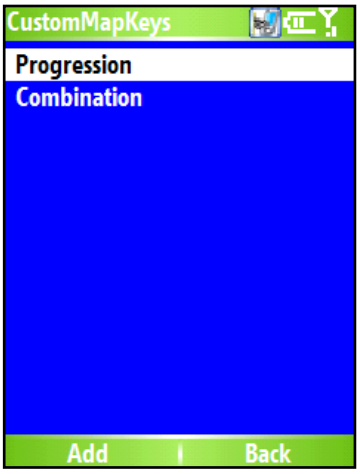


Pointer speed menu.

The last option is to change the key action type. For a mouse movement button there are two options: Single and Press/Release.

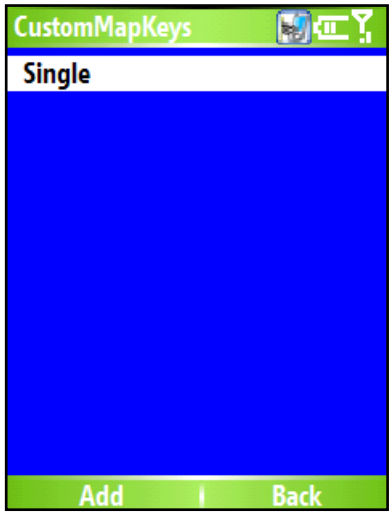
Action type menu.

For a normal button you will have the option to choose Progression or Combination.



Action type menu.

If there is only one key action set to this button you will of course not have the option to set the type to progression or combination.

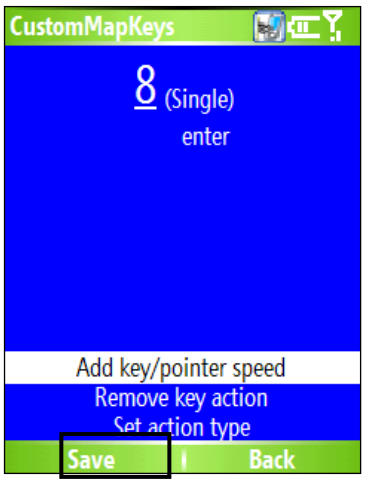


Action type menu.

Press “Add” to change the key action type.

Key Action types:

- Single: The key will be pressed and released once.
- Combination: All the keys will be pressed and then released.
- Progression: The keys will be pressed and then released one by one.
- Press/Release: First key action from the phone will press the button, the next will release it. Especially useful for mouse movement and the possibility to select text.

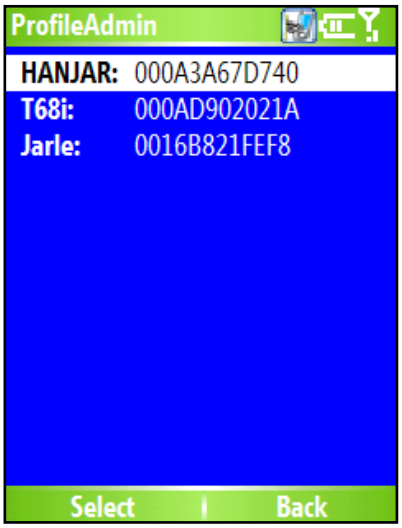


KeyAction menu.

When you are finished editing the KeyAction, press “Save” to store the changes in the database.

Open Bluetooth device list

Select ”Open device list” in the Profile administrator menu. The entire list of saved devices will then be shown.

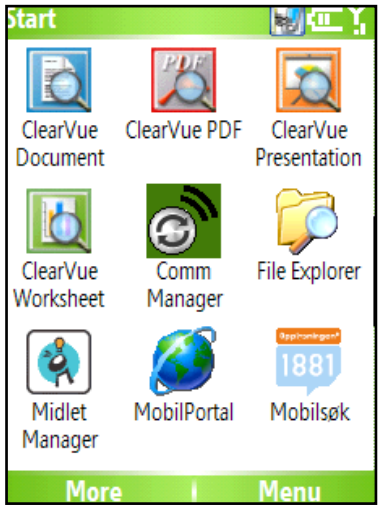


Bluetooth device list.

Search for Bluetooth devices

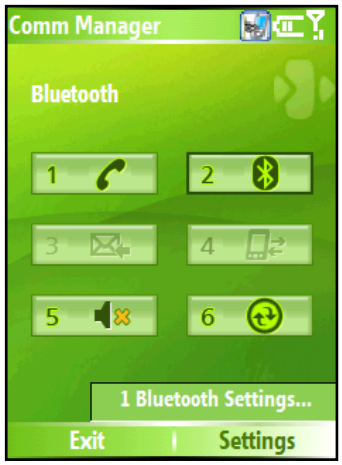
If the initial pairing with the computer or phone you want to communicate with is already done, you do not need to continue with the steps below. If it is a new Bluetooth device you will need to search and then to the initial pairing.

To add a new device, go to the start menu and open the “Comm Manager”.



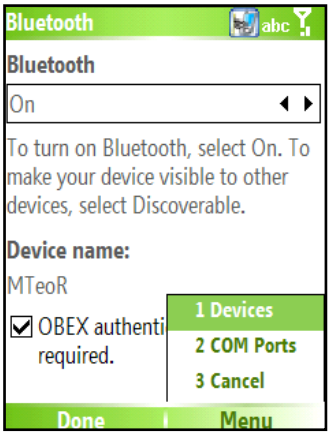
Start menu.

From Settings select “Bluetooth Settings”.



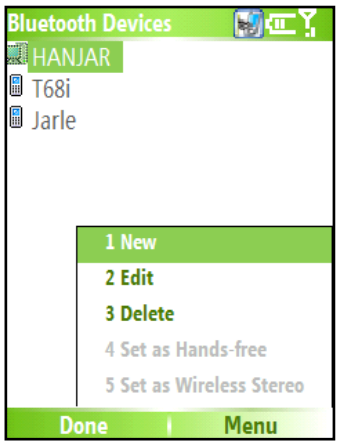
Comm Manager.

From Menu open “Devices”.



Bluetooth settings.

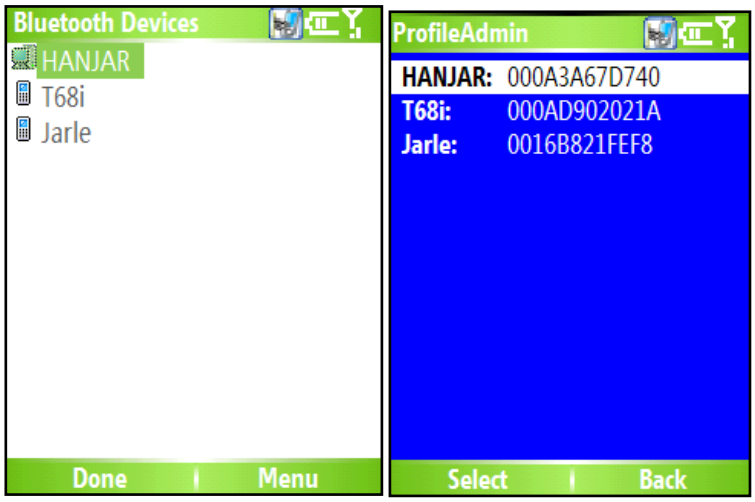
You will then get the list of paired devices. If you want to add a new device, select Menu and “New”.



Bluetooth devices.

It will then search for new devices. When it has found the device you want to pair with follow the instructions on the screen and remember to add a passkey that is at least 8 digits long.

Set the Bluetooth settings on the other Bluetooth device to visible (Discoverable). After the device is saved turn off visible mode again, this is an easy security measure. Also when you do not use Bluetooth deactivate it.



Bluetooth device list.

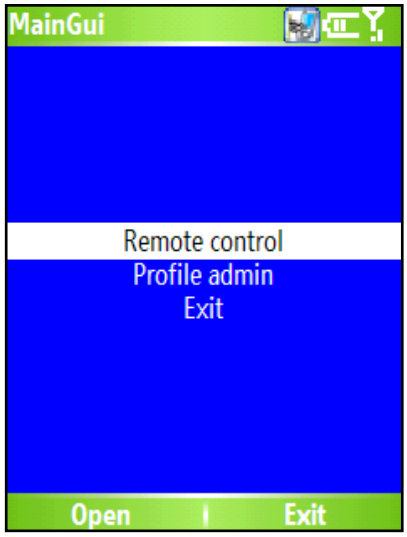
All devices that are stored by Windows Mobile will show up in the Remote Control application. The picture on the left is from the Comm manager in Windows Mobile the picture on the right is from the Remote Control application.

Delete a saved Bluetooth device

From the Comm manager you can also delete devices. Enter Bluetooth Settings and then Devices. From the Menu button select Delete.

Bluetooth devices.

Remote Control

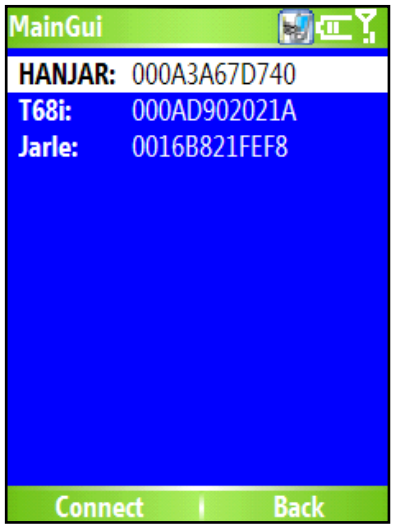


Main menu.

In the main menu select “Remote Control” and press open to start the Remote Control application.

Connect to the server application

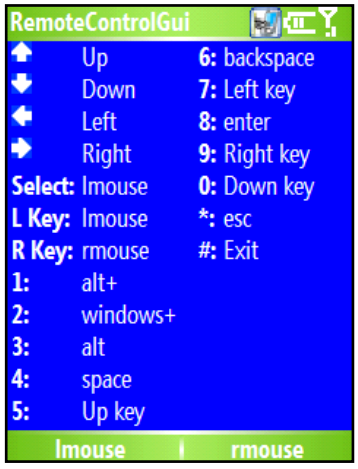
In the main menu select Remote Control and press “Open” with the softbutton. A list of all the stored devices will then appear. Select the one you want to connect to and press “Connect”.



Saved Bluetooth devices.

Remember you have to do the initial pairing before connecting to the server application. If the server application is up and running the phone will then ask to connect.

If the user accepts the connection the Remote Control GUI will appear. It contains information about all the key mappings on the phone. It shows a list of all the commands. If a command has a “+” after the name (like windows+) it means that it has more than one key mapped to it.



Remote Control GUI.

Default action keys

Softbuttons:
1. (Left softbutton): Left mousebutton.
2. (Right softbutton: Right mousebutton.

Directional buttons:
Up: Mouse up.
Down: Mouse down.
Left: Mouse left.
Right: Mouse right.

Select: Left mouse button, press/release. Makes it possible to select text.

Numeric keys:
1: Alt + Tab, switch programs.
2: Windows + D, minimize all windows.
3: Alt, useful to access toolbar options.
4: Spacebar.
5: Up arrow (Keyboard).
6: Backspace.
7: Left arrow (Keyboard).
8: Enter.
9: Right arrow (Keyboard).
***:** Esc.
0: Down arrow (Keyboard).
#: Exit.

Default key mappings.

Appendix D: *Bluetooth Ping* Source Code (experiment 2)

Source code available at: https://github.com/jarlehansen/brunel

```
public class BluetoothDevicePingImpl implements BluetoothDevicePing {
[…]
    public void startSearch(final List<String>
registeredBluetoothDevices) {
        if (registeredBluetoothDevices.size() == 0) {
            bluetoothDeviceHandler.bluetoothDevicePingFinished();
        } else {
            executorService.execute(new
            BluetoothDevicePingInternal(bluetoothDeviceHandler,
            registeredBluetoothDevices));
        }
    }

    public void stopSearch() {
        continueSearch.set(false);
        executorService.shutdown();
    }

    private static class BluetoothDevicePingInternal implements
    Runnable, DiscoveryListener {
    […]
        public void run() {
            int counter = 0;

            while (continueSearch.get() && bluetoothDevices.size() >
            0) {
                try {
                    if (counter >= bluetoothDevices.size()) {
                        counter = 0;
                    }

                    devicePingCompleted.set(false);
                    String btAddress = bluetoothDevices.get(counter);


            LocalDevice.getLocalDevice().getDiscoveryAgent().searchServ
            ices(null, new UUID[]{BT_SERVICE}, new
            BluetoothDeviceImpl(btAddress), this);

                    while (!devicePingCompleted.get()) {
                        synchronized (lock) {
                            if (!devicePingCompleted.get()) {
                                lock.wait();
```

```
                            }
                        }
                    }
                } catch (InterruptedException ie) {
                } catch (BluetoothStateException bse) {
                } finally {
                    counter++;
                }
            }

            if (bluetoothDevices.size() == 0) {
                bluetoothDeviceHandler.bluetoothDevicePingFinished();
            }
        }

        public void servicesDiscovered(int transactionId,
        ServiceRecord[] serviceRecords) {
            if (serviceRecords.length > 0) {
                  final RemoteDevice remoteDevice =
                  serviceRecords[0].getHostDevice();

                if (logger.isDebugEnabled()) {
                        logger.debug("Bluetooth device and service
                        found: " + remoteDevice.getBluetoothAddress());
                }

                final BluetoothDevice bluetoothDevice =
                createBluetoothDevice(remoteDevice);

bluetoothDeviceHandler.addFoundDevice(bluetoothDevice);

removeFoundBluetoothDeviceFromPingList(bluetoothDevice.getBluetoothAdd
ress());
            }
        }

        private BluetoothDevice createBluetoothDevice(final
        RemoteDevice remoteDevice) {
            String name = "";
            try {
                name = remoteDevice.getFriendlyName(true);
            } catch (IOException io) {}

            final BluetoothDevice bluetoothDevice;
            if (name == null || "".equals(name)) {
                  bluetoothDevice = new
                  BluetoothDeviceImpl(remoteDevice.getBluetoothAddress(
                  ));
            } else {
                  bluetoothDevice = new
```

```
                    BluetoothDeviceImpl(remoteDevice.getBluetoothAddress(
                    ), name);
                }

                return bluetoothDevice;
            }

                private synchronized void
                removeFoundBluetoothDeviceFromPingList(String
                foundBtAddress) {
                    foundBtAddress = foundBtAddress.toUpperCase();

                for (int i = 0; i < bluetoothDevices.size(); i++) {
                    String btAddress =
                    bluetoothDevices.get(i).toUpperCase();

                    if (btAddress.equals(foundBtAddress)) {
                        bluetoothDevices.remove(i);
                        break;
                    }
                }
            }

            public void serviceSearchCompleted(int transactionId, int
            responseCode) {
                synchronized (lock) {
                    devicePingCompleted.set(true);
                    lock.notifyAll();
                }}
        […]
}
```

Appendix E: Android Touch Detectors Source Code (experiment 3)

Source code available at: https://github.com/jarlehansen/brunel

```
class SimpleTouchDetector extends
GestureDetector.SimpleOnGestureListener {
        @Override
        public boolean onSingleTapConfirmed(MotionEvent e) {
            jpctModel.onModelTouchEvent(e.getX(), e.getY());
            return true;
        }

        @Override
        public boolean onScroll(MotionEvent e1, MotionEvent e2, float
      distanceX, float distanceY) {
            jpctModel.moveHorizontal(distanceX * 3.4F);
            jpctModel.moveVertical(distanceY * 3.4F);
            return true;
        }
    }

    class PinchDetector extends
    ScaleGestureDetector.SimpleOnScaleGestureListener {
        @Override
        public boolean onScale(ScaleGestureDetector detector) {
            inZoomMode.set(true);

            if ((detector.getCurrentSpan() -
            detector.getPreviousSpan()) > 0)
                jpctModel.zoom(detector.getScaleFactor() * 24F);
            else
                jpctModel.zoom(-(detector.getScaleFactor() * 24F));

            inZoomMode.set(false);

            return true;
        }
    }
```

Appendix F: Simple-C2DM (experiment 4)

During the development of the Customised Android Home Screen experiment we gained experience with the C2DM technology. We could see the benefits it provided, such as increased battery life by using a common push-message technology. However, we also were confronted with challenges and saw the potential for improvements.

Background Information

We started using C2DM as part of experiment 4, the *Customised Android Home Screen*. We found the technology very interesting, but we also saw the potential for improvement and especially with the developer tools and API. To target the main drawbacks we found during our investigation, we introduced a new open source library providing novel features not part of the standard Android platform.

Design and Architecture

We implemented a new open source library called Simple-C2DM[46]. It was created specifically to simplify the development of applications using C2DM on the Android platform. We start by presenting the library, with an introduction of the main components and architecture.

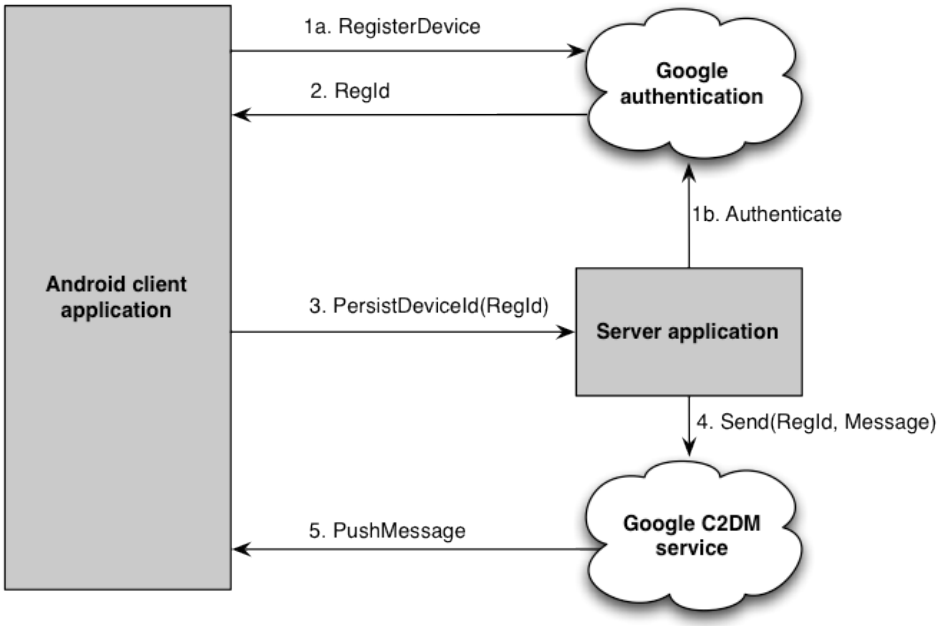
System Components

C2DM was created by Google and initially released in 2010. One of the main reasons why it is such a compelling technology is because it is part of the standard Android platform, where it shares functionality with other applications such as *Gmail* and

[46] http://code.google.com/p/simple-c2dm/

Android Market. This has the benefit of using one shared push messaging connection for multiple applications, thereby saving battery power in the mobile device.

We will provide a short introduction of how C2DM works, before presenting the open source library we created. An overview of the C2DM process is shown below.



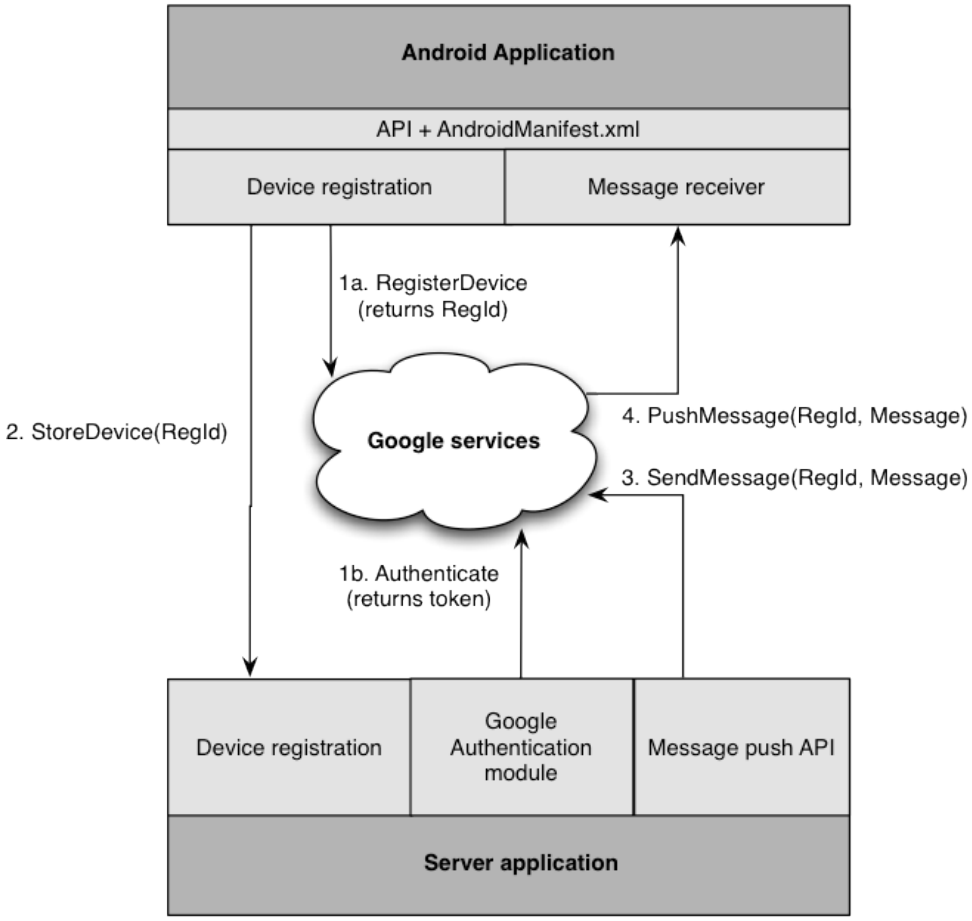
C2DM system overview.

The Android client starts by registering the device with the Google authentication servers (1a). The return value after a successful registration is an identification number. Similarly, the server application is required to retrieve an authentication token, which will be used to send push messages to the devices (1b). Both the mobile client id number (3) and the authentication token are usually persisted on the server application. The next step is to create the actual message (4) in the C2DM format, which consists of the client registration id, authentication token, and the message. Finally, the message is sent through the C2DM servers (5) and then received on the mobile client.

When working with C2DM we found the API and tools to be limited in a few important aspects. Accordingly, there were two main reasons why we wanted to provide a new implementation:

1. **To create a higher level of abstraction for certain key features.** We felt that the abstraction level was too low level in parts of the C2DM library cases, such as when extending the base class.
2. **To provide features not available in standard C2DM.** We saw the potential for adding useful features not part of the standard library. This included the management of registration ids and automatic XML generation.

The Simple-C2DM library was created to target these features, and the implementation consisted of three main components: 1) *Android client library*, 2) *Server utilities*, and 3) *manifest generator*. The figure below shows an overview of the Simple-C2DM library, and presents the general steps involved in sequential order.



Simple-C2DM overview.

The Android client library was a jar-file that each project, which wanted to use Simple-C2DM, had to include. It consisted of the library classes in addition to the standard

c2dm.jar-file, which we extended in our own classes. The features of this component included annotation support and direct callback classes.

Server utilities were packaged as another jar-file, provided for the server application. It included support for handling registration ids, which was sent from the Android clients. Additionally, it also provided functionality to easily generate a Google authentication token and made it easier to construct the C2DM message format.

The final component was the manifest generator. Within Android projects, there is a file called *AndroidManifest.xml*[47] that is responsible for essential information about the application, such as the permissions required. The manifest-generator was created to avoid all the manual steps involved in this process, by automatically generating parts of the XML-file.

System Architecture

In this section we go into more detail on the implementation aspects of Simple-C2DM. In particular, we focus on the fact that we wanted *to create a higher level of abstraction for certain key features* and *to provide features not available in standard C2DM*.

Starting with the higher level of abstraction, we solved this by introducing support for annotations. To utilise standard C2DM in a project, the developer needs to extend a base class, namely the *C2DMBaseReceiver*, and then override the required methods. Usually this is one or more of the following: *onMessage*, *onError*, and *onRegistered*. In Android, and Java in general, one is only allowed to extend from one class. In this scenario, this has the disadvantage that once the class extended the base receiver, which is required, it cannot extend anything else. Additionally, it also implies that the methods inherited must be exactly the same as the base class, both the method name and variables.

[47] http://developer.android.com/guide/topics/manifest/manifest-intro.html

In Simple-C2DM we provided annotation support to solve this issue. Annotations provide metadata that will not directly affect program semantics[48] and are usually handled by tools and libraries. The main feature we found particularly useful and important by introducing annotation support was flexibility. Developers were no longer strictly forced to follow a specific pattern in the source code. For instance, one benefit from this feature was the ability to split the receiver class into different classes. This meant that the developers could include one class for handling registration and another taking care of messages and errors. We also added a feature to the *@OnRegistered*-annotation that sends the registration id automatically to the server. If this feature was not needed, the developer will simply not add the URL to the annotation.

Another benefit of using annotations was the ability to make the method signatures adaptable. We did not depend on the method names, since we scanned for known annotations. Specifically, the annotations we supported was *@OnRegistered,* *@OnMessage,* and *@OnError*, which are all shown below.

```
public class AnnotatedCallback {

    @OnRegistered("http://ping-servlet.appspot.com")
    public void registered(String registrationId) {
        SC2DMLogger.i("registered() called: ", registrationId);
    }

   @OnMessage
    public void message(Context context, String msg) {
       SC2DMLogger.i("message() called: ", msg);
   }

    @OnError
    public void error(String errorMsg) {
        SC2DMLogger.i("error() called: " + errorMsg);
    }

}
```

Annotation support.

The second task we wanted to introduce in Simple-C2DM was new features not available in the standard implementation. One of the major issues with C2DM in our opinion is the *AndroidManifest.xml*-file, and especially the required configuration that

[48] http://docs.oracle.com/javase/1.5.0/docs/guide/language/annotations.html

needs to be provided. Not only does one lose the good refactoring support when using XML-files instead of Java-files, but also more importantly one has to use a cryptic syntax when defining the C2DM service. One needs to copy the package name (*com.googlecode.sc2dm.activity* in the example below) of the C2DM receiver class and then add, for example, a permission-tag, shown in the figure below.

```
<permission android:name=
      "com.googlecode.sc2dm.activity.permission.C2D_MESSAGE"
      android:protectionLevel="signature"/>
<uses-permission android:name=
      "com.googlecode.sc2dm.activity.permission.C2D_MESSAGE"/>
```

AndroidManifest.xml example.

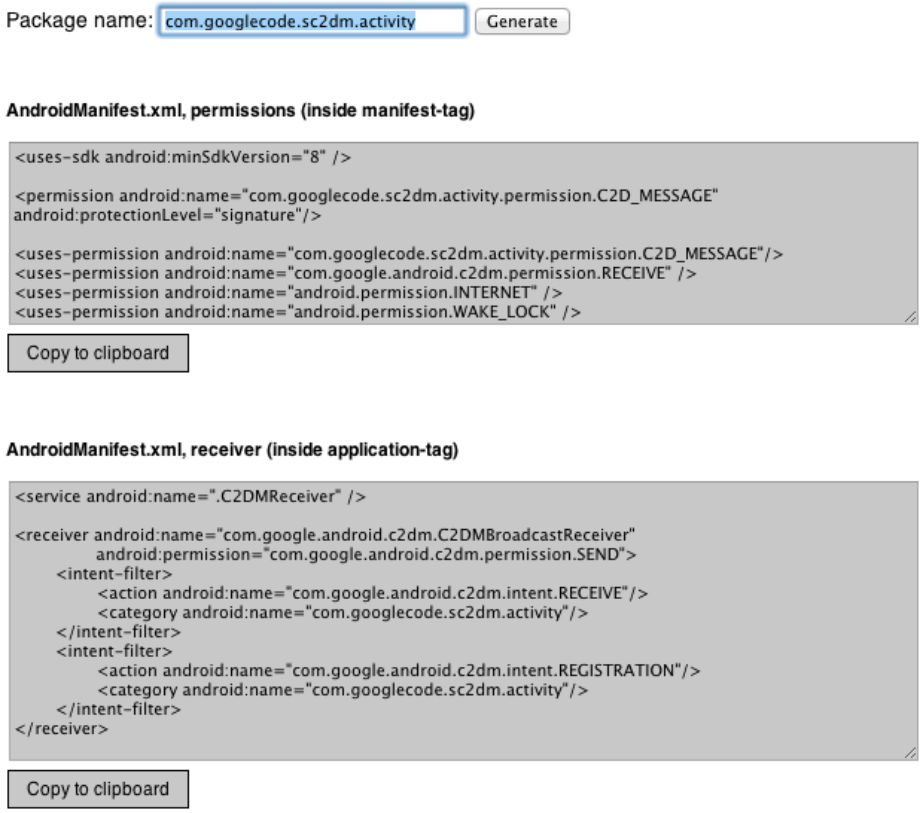
If the developer accidentally adds a small typo or forgets something in the manifest-file, the application will not receive any push messages. The error message shown on the mobile client can be very hard to debug and it is difficult to find the root cause. In the table we have added the common problems with the configuration of C2DM and their respective error messages.

Error/typo in	In file	Error message
package name, *permission* tag	AndroidManifest.xml	Permission Denial: receiving Intent
package name, *uses-permission* tag	AndroidManifest.xml	Receiver package not found
service name	AndroidManifest.xml	Unable to start service Intent
registration package, inside *receiver* tag	AndroidManifest.xml	*No error message*
senderId (e-mail address)	Class extending C2DMBaseReceiver	Error=MismatchSenderId (This error is shown on the server-side when trying to send message).

Common C2DM errors.

The need to copy-paste and edit several XML-tags manually is a very error-prone process. In Simple-C2DM we tried to solve this problem by offering two alternatives. If the developer does not want to, or cannot for some reason, use code generation, we

created a manifest-generator[49] hosted on the Google App Engine. This is a webpage that will take the package-name as input and generate the needed XML-tags, the image below shows a screenshot of the application. One still needs to copy this into the *AndroidManifest.xml*, but does not have to edit the XML since it was generated with the correct package-name.



Manifest generator.

The second option, which in our opinion is the best, is to generate the XML-tags automatically using an annotation processor. This part of the Simple-C2DM library was created as an experimental feature to figure out if we were able to automatically generate these XML-tags without the need to manually type in the package name. Because this uses an annotation processor, there is a configuration step involved when setting up the development environment. After this is completed, the annotation processor automatically runs when the project is compiled. In our project, it scans for an Activity-class marked with the *@SC2DMAutomaticManifest* annotation as shown in the figure below. It starts by creating a backup of the old *AndroidManifest.xml* file, and then

[49] http://sc2dm-manifest-generator.appspot.com/

automatically generating the required XML-tags. This eliminated the error-prone and tedious task of manually managing the XML-file. It also had the benefit that if there were any changes to the configuration, such as a new package name, it was automatically updated by our library.

```
// Main Android activity class
@SC2DMAutomaticManifest
public class MainActivity extends Activity {
      [...]
}
```

Annotation processor.