Exploring solutions for mobile companionship: A design research approach to context-aware management

Tor-Morten Grønli, Gheorghita Ghinea, Bendik Bygstad

A R T I C L E   I N F O

Article history:
Available online 8 November 2012

Keywords:
Context awareness
Information management
Mobility
Companionship

A B S T R A C T

The mobile phone is not just another device; it is with you day and night, and you rely on its capabilities in work and in private. In short, the mobile phone is your companion. As your companion, it should understand your situational and informational needs. How do we increase the friendliness of your mobile phone, in order to fulfil this promise? In this paper, we explore how context awareness can be used for managing the user mobile experience. To this end, we employed a design research approach to integrate context-aware and cloud based services in an Android application. Through a user evaluation and proof-of-concept implementation we show how new technologies can increase the friendliness of your mobile phone. In so doing, we provide evidence that adaptive applications based on user context offer a fertile ground for taking mobile companionship to the next level.

1. Introduction

We now realize that the mobile telephone is something completely different than the old fixed-line telephone. Your smart phone certainly is a computer, with an impressive range of functionalities, including full Internet access. It is also your net bank, music player and chart plotter with GPS. It will probably become your future payment mechanism for the metro and car parking.

However, it also something more emotional; it stays with you day and night, it increasingly contains much of your personal life, and it is something you really dread to lose. In short, it is your companion. A companion is someone that can be trusted, and knows your situation. A companion is sensitive to your needs and makes your daily life easier. In order to do so, the companion must know your current context, and being able to adapt to various contexts.

This scenario is not yet fully achieved by mobile technology, but in this article we will argue that context awareness is a key concept to understand how mobile phones can become true companions. Context-aware solutions and applications have been around for some years, successfully enriching applications (Cheverst, Davies, Mitchell, Friday, & Efstratiou, 2000; Dey & Abowd, 1999; Lovett & O’Neill, 2010; Schilit, Adams, & Want, 1994). In this work we seek to build further on these achievements and utilize context as a source of information for user information and interface tailoring. Earlier approaches have been limited, in that they have only looked at one source of context-aware information or more than one source, but utilized separately. Building on design research (Hevner, March, Park, & Ram, 2004) we propose a different approach, where we combine context-aware information from several sources to build a rich foundation to base our solutions on. Furthermore, we exploit cloud-computing technology to create a new user experience and a new way to invoke control over your phone.

Our research question is: How can we use design research to explore context-aware mobile solutions?

We chose a design research approach because it allows for the design of a working solution, with the appropriate and systematic evaluation of the results. Our solution is a remote configuration of an Android phone, which uses context-aware information to constantly adapt and change in response to the user’s implicit requirements.

Technologies for context-aware mobile computing, in this section we offer working definitions of our key technology terms; context aware computing, sensor based input, cloud computing and Google App Engine.

1.1. Context-aware computing

The notion of context-aware computing denotes the ability for the devices to adapt their behaviour to the surrounding environment, ultimately enhancing usability (Dey & Abowd, 1999). In particular, context-awareness is becoming an important factor when it comes to mobile devices, as users bring their mobile phone...
or tablet just about everywhere, highlighting the need for adapting the content to the users current situation.

Dey and Abowd (1999) have remarked that if we fully understand context in a given environment and setting we are better able to adapt the context-aware behaviour of our applications. The consequence of this is devices that adapt content based on the user’s context, eliminating the need for users to manually do these tasks. Indeed, context is a major part of our daily life and support for sharing and using contextual information will improve user interaction. As such, it is important to take advantage of this by viewing people as consumers of the information and not other systems.

In our research, we combine context information received both from the phone itself with information retrieved from cloud-based servers. All data is integrated to create a context-aware mobile device, where we implement a new customized home screen application for Android-enabled devices.

1.2. Sensor based input

Sensors are an important source of information input in any real-world context and several previous research contributions look into this topic. For instance, Parviainen, Pirinen, and Perttilä (2006) approached this area from a meeting room scenario. They found several uses for a sound localization system, such as: automatic translation to another language, retrieval of specific topics, and summarization of meetings in a human-readable form. In their work, they find sensors a viable source of information, but also acknowledge there is still work to do, like improving integration.

Modern smart phones have a number of built in sensors. For example, in our test phone (the HTC Nexus One) we had 5 sensors available: accelerometer, magnetic field, orientation, proximity and light. All of these sensors can be accessed as local services through well-defined APIs in the Android platform. In our work, by taking the aspect of sensors and context-awareness and integrating it in a mobile application we reduce the workload for the end users. Thus, in combination with a cloud-based server application we are able to remotely configure the mobile devices independent of the device types. This creates a new concept of context-awareness and embraces the user in ways previously unavailable.

1.3. Cloud computing

Cloud computing has received considerable attention in the software industry (Armbrust et al., 2010; Borenstein & Blake, 2011). It is a buzzword frequently used for a variety of services, ranging from hosted virtual machines to simple web based email applications. We have used the definition created by National Institute of Standards and Technology (Mell & Grance, 2011), United States:

“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”.

Large IT companies like Microsoft, Google and IBM, all have initiatives relating to cloud computing (Guan, Ke, Song, & Song, 2011) which have spawned a number of emerging research themes, among which we mention: cloud system design, benchmarking of the cloud (Mei, Chan, & Tse, 2008) and provider response time comparisons (Binnig, Kossmann, Kraska, & Loesing, 2009). Mei et al. (2008) have pointed out 4 main research areas in cloud computing that they find particularly interesting, namely pluggable computing entities, data access transparency, adaptive behaviour of cloud applications and automatic discovery of application quality.

One of the major concerns with cloud computing is privacy, as user data is stored in an infrastructure that the cloud provider operates. Addressing this issue, Mowbray and Pearson (2009) looked at how it is possible to implement a client-based privacy manager in a cloud-computing environment. Their solution reduces this risk by helping users to control their own sensitive information. They implemented an obfuscation and de-obfuscation service to reduce the amount of sensitive information held within the cloud. Another feature was also implemented to allow users to express privacy preferences about the treatment of the personal information. Although security is outside the main scope of work reported in this paper, we did address it by using state of the art Open Authentication services implementation in our application and integrating with the Google login process. We also asked the users that were testing the system if they had any concerns about storing their data in the cloud. An in-depth look at the results from the user experiment is presented further on in this article.

Our work focuses on data access transparency, where clients transparently push and pull for data from the cloud, and adaptive behaviour of cloud applications. We adapted the behaviour of the server application based on context information sent from the users’ devices thus integrating context and cloud on a mobile platform.

1.4. Google App Engine

For our experiment we used a cloud-based solution called Google App Engine. This system is based on a Paas (Platform as a Service) model, in which the cloud provider offers web servers for developers to publish their applications on in addition to resources like a database and an administrator console.

Our cloud based server application handled the integration with the mobile clients and other services such as Google calendar and contacts. Microsoft has done similar work, integrating context into applications by exploiting desktop calendars (Cutrell, Robbins, Dumais, & Sarin, 2006). They also introduced a tag library, to tag email and file elements with a context label that is meaningful to the user. Extending this work, we further improved on it and added functionality to our system by allowing users to tag their meetings in the Google cloud with context.

2. Methodology

Based on the literature review performed and the tasks at hand we chose design research as the methodology approach. In this section we describe this approach and relate our research to the design research framework realization.

Innovation processes have been studied for three decades in information systems research, mainly from a descriptive perspective. The past decade has seen the growth of Design Research, which focuses on the development and evaluation of the designed artefact. Design Research strives to “create things that serves human purposes” (March & Smith, 1995), and is concerned with both the process of constructing the artefact, and with the designed end product. This differs from the natural sciences where the goal is to understand reality. Hevner and Chatterjee (2010) discuss several benefits in using the Design Research approach. They state that the goal of Design Research is utility as such and that the continuous building and evaluation of artefacts accomplish this.

Design Research is particularly relevant for understanding fast growing, new areas/services. The application area is of the work described in this paper is new, which calls for experimentation. Also, the design challenges of mobile computing are particularly difficult, because of the limited size of the physical artefact. Well-known design research frameworks realizations are March and
In our research, the framework by Vaishnavi and Kuechler (2011) has been choice of implementation. The framework is illustrated in Fig. 1. This approach describes a dynamic design process in five steps. The first is an awareness of the problem, which is achieved through consulting a number of sources, both industrial and academic. The purpose of this phase is to identify and understand the context of the problem/problem area. This leads to a suggestion of a tentative design of a contribution. The third step is the development of an artefact, realizing the suggested solution identified and fortified in the business domain. The fourth step is evaluation, which may take form of a formal evaluation, test activities and/or usability studies. Outputs from this step are typically performance measures. The last step is the conclusion, which assesses the overall results of the design process. As illustrated in Fig. 1 the process is iterative; it is repeated several times, where experience and knowledge from the last iteration flows back into the early steps of the process. Based on a SCRUM (Schwaber & Beedle, 2002) inspired approach from software engineering, a limited set of tasks is realized in each iteration, based on a highest priority first ordering.

As described previously, the framework of Vaishnavi and Kuechler (2011) has been the choice for realization of research project. Table 1 illustrates the different steps in the framework with the different sections and phases of the research.

<table>
<thead>
<tr>
<th>Step of framework</th>
<th>Corresponding part of research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Awareness of the problem</td>
<td>An awareness of the problem was achieved through consulting a number of sources such as relevant literature and industry contacts. The novelty of the approach was assessed and confirmed.</td>
</tr>
<tr>
<td>Suggestion</td>
<td>Based on the awareness, a literature review was conducted to know state of the art. Additionally, a set of requirements was developed to describe the proposed solution. Further, a minor prototype was developed as a technology spike to test/verify important technical issues.</td>
</tr>
<tr>
<td>Development</td>
<td>A full-scale solution was developed according to the software development methodology.</td>
</tr>
<tr>
<td>Evaluation</td>
<td>The evaluation was conducted in two steps. First a formal software testing and analysis was done in accordance with techniques from software engineering. Then a real life user experiment was conducted, which provided feedback from participants.</td>
</tr>
<tr>
<td>Conclusion</td>
<td>Results were summarized, and opportunities for further research were identified.</td>
</tr>
</tbody>
</table>

3. Design and implementation

For our user experiment we implemented a proof of concept application suite, a fully functional demonstration of the mobile companion application. In keeping with the principles underlying such applications, from the outset one of its main technical goals was to make the interaction between the cloud and the mobile device as seamless as possible for the user.

The mobile companion application was designed with three major components: an Android client, a cloud server application, and the remote Google services. Fig. 2 gives an overview of the implementation of the system (shaded boxes in the diagram represent the parts of the system we created). The white boxes, like Google calendar and contacts, are external systems the mobile companion communicated with. The server application was deployed remotely in the cloud on the Google App Engine, whilst data was also stored remotely in Google cloud services.

After the Android client was installed on the mobile device, the device will register itself with the Google services. The users would then start by logging on to the webpage from our cloud-hosted application. This webpage is part of the server application hosted on the Google App Engine. The login process uses the Google username/password. By leveraging the possibilities with Open Authorization (OAuth) we facilitated user sharing of their private calendar appointments and contacts stored in their Google cloud account without having to store their credentials locally.

After a successful authentication the user is presented with a webpage showing all interface configuration options, allowing for in depth configuration. Because the configuration for each user is stored in the cloud, we thus avoided tying it directly to a mobile device. One of the major benefits of this feature is that the user does not need to manually update the preference settings of each device s/he uses; the user has a “master configuration” stored externally that can be directly pushed to their phone or tablet. It is incorporated back into the development loop. The application suite was incrementally increased in an iteration producing a new version. The second round of testing involved a larger number of users. A total of 38 participants were involved in the second round. All participants were of mixed age, gender and computer expertise, although all had to some extent technical knowledge. They were given a set of tasks to perform with the phone and application and answered a questionnaire when finishing the tests. An equal number of positive and negative questions were included in the questionnaire and all answers were indicated on a 4-point Likert scale (a scale with an even number of points was chosen in order to force participants to take a non-neutral stance in their opinions). In the evaluation phase we analyzed the results from the test sessions and we found valuable data supporting our research. Building on data collected as well as artefacts produced we were able to make a novel research contribution and support our research question.
also easier to add more advanced configuration options when the user can take advantage of the bigger screen, mouse and keyboard on a desktop/laptop PC for entering configuration values other than those found on mobile devices.

On the webpage, by selecting the applications s/he wants to store on the mobile device and pressing the “save configuration”-button, a push message is sent to the client application, via a cloud to device mechanism which we describe in the next section.

4. Cloud to device messaging

The message is sent with a push feature for Android called C2DM (cloud to device messaging), available from Android 2.2. The C2DM feature requires the Android clients to query a registration server to get an ID that represents the device. This id is then sent to our server application and stored in the Google App Engine data store.

The C2DM process is visualized in the figure below (Fig. 3). This technology has a few very appealing benefits: messages can be received by the device even if the application is not running, saves battery life by avoiding a custom polling mechanism, and takes advantage of the Google authentication process to provide security.

Our experience with C2DM was mixed. It is a great feature when you get it to work, but the API is not very developer friendly. This will most likely change in the future since the product is currently in an experimental state, but it requires the developer to work with details like device registration and registration id synchronization.

The calendar and contacts integration was also an important part of the Android mobile companion. We decided to allow the Android client to directly send requests to the Google APIs instead of going the route through the server. The main reason for this is that we did not think the additional cost of the extra network call was justified in this case. The interaction is so simple and there is very little business logic involved in this part so we gave the clients the responsibility for handling it directly. The implementation worked by simply querying the calendar and contact API and then using xml parsers to extract the content.

4.1. Meta-tagging

To make it possible for users to tag their appointments and contacts with context information we added special meta-tags. By adding a type tag, for example $\{\text{type} = \text{work}\}$ or $\{\text{type} = \text{leisure}\}$, we were able to know if the user had a business meeting or a leisure activity. We then filtered the contacts based on this information. For instance, if the tag $\{\text{type} = \text{work}\}$ was added, this lets the application know that the user is in a work setting and it will automatically adapt the contacts based on this input. In a work context only work related contacts would be shown. To add and edit these tags we used the web-interface of Google contacts and calendar.

4.2. Sensors as input data

The sensors on the mobile device were also used as input to the application. We used the API available on the Android platform and through a base class called SensorManager we were able to access all of the built-in sensors on the mobile device.

We started out by just showing the input values from the sensors in our pilot study. When we expanded the application after the
initial tests we wanted to use the sensor input to further enhance the user experience. We ended up using two features directly in the application, the accelerometer and the light sensor. The accelerometer was used to register if the device was shaking. If the device is shaking it means that the user probably on the move, for example running or walking fast. In these cases, we automatically change the user interface to a much simpler view that has bigger buttons and is easier to use when on the move.

The second sensor we used in our experiment was the light sensor. By constantly registering the lighting levels in the room we adjusted the background colour of the application (Figs. 4 and 5).

We changed the background colour of the application very carefully, as it would be very annoying for the users if colour changes were happening often and were drastic. Accordingly, we gradually faded colour when the lighting values measured from the environment changed.

5. Results

An important part of our research was the introduction of context-awareness from multiple sources. We built, as described, the system around three main components, the calendar/contacts integration, cloud based server and sensor input data. As previously mentioned, the user evaluation was conducted in two rounds, where the first round involved 12 users. Results from this preliminary test session where used as basis for the product increment. The second round of testing involved a larger number of users (38 participants of mixed age, gender and computer expertise), who were given a set of tasks to perform with the phone and application and answered a questionnaire consisting of 17 statements (as shown in Table 2).

As our distribution is normal and interval, we used the one-sample t-test to analyze our results.

5.1. Calendar and contact services integration

The calendar and contacts integration added the idea of tagging a context to the meeting. The users added a predefined custom tag to the meeting and contact description that filtered the contacts based on the upcoming meeting. For instance, in Fig. 6 the next upcoming meeting is tagged as leisure related. This means only contacts with the same tag are shown.

The feedback from the user evaluation was that this feature is useful, particularly in a work setting where you would like to only be shown the work related contacts. Further improvement could be to include a filter button on the contacts tab where users could easily do fine-grain filtering of the contacts.

To register the tags we used the standard Google calendar and contacts web-interface. Such a tight integration with the Google services and exposure of private information was not regarded as a negative issue. As shown below (Fig. 7) most of the users

<table>
<thead>
<tr>
<th>Table 2</th>
<th>User questionnaire statements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement</td>
<td>Statement text</td>
</tr>
<tr>
<td>S1</td>
<td>The close integration with Google services is an inconvenience, I am not able to use the system without changing my existing or creating a new e-mail account at Google</td>
</tr>
<tr>
<td>S2</td>
<td>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</td>
</tr>
<tr>
<td>S3</td>
<td>When moving around, a simplified user interface is not presented</td>
</tr>
<tr>
<td>S4</td>
<td>The background colour in the application changes when the lighting in the room changes</td>
</tr>
<tr>
<td>S5</td>
<td>The adaptability of the application is a feature I approve</td>
</tr>
<tr>
<td>S6</td>
<td>Open comment question—(participants were free to express their thoughts)</td>
</tr>
</tbody>
</table>
surveyed (33 out of 38) disagreed that this was an inconvenience. This makes room for research of further integration with Google services, where amongst them the Google+ platform will be particularly interesting.

5.2. Cloud integration

As previously mentioned, we used the state of the art feature C2DM to push messages to the mobile devices. The technology is cutting edge, released for public at Google I/O 2011, and has several benefits, including security and increased battery life for the mobile device. The API is not (yet) very developer friendly and the response times in our experiment did have some significant spikes. Previously, one of the common ways of providing up-to-date data on clients was to poll the server for changes with a fixed time interval. Not only does this mean that each application needs to implement their own solution, but it also drains battery life because of constantly sending and receiving data across the network. With C2DM there is a standardized way of doing it, shared between all applications on the phone that use the push feature.

The entire system was closely integrated with a cloud based server application. In the questionnaire we asked users if they had any reservations or concerns about storing their information in the cloud. Most users (35 of 38) indicated that they did not see any issues with giving out their information, like e-mail address, to a cloud based system and found cloud storage at Google a useful feature (Fig. 8).

We did, however, get some user comments stating that they were not comfortable sharing too much information in the cloud. Moreover, some did express reservations on their personal information being stored on some unknown place in the cloud provider data centre.

Privacy is also an important issue when dealing with cloud computing. We used the Google authentication process, with OAuth, to validate the username and password of the users. This is one of the major benefits of using cloud computing, and we were able to take advantage of the cloud provider technology, Google in our case, for security without the need of implementing our own solution. The Google authentication process is used in applications like Gmail and is therefore thoroughly tested and is continually improved.

The use of sensors on the phone was the last context source we used. The Android platform has good support for reading data from the sensor and we used and integrated several sensors as sources of information. The first sensor we used as input to the application was the accelerometer (Fig. 9).

When the device was shaking, the mobile companion switched automatically to a simpler layout to help the user navigate even when on the move (Fig. 10), and as can be seen in the graph below (Fig. 11), the accelerometer did for the most function properly.

One issue we did experience during the user experiments was that 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. Adding logic to the application that take cares of issues such as the amount
of movement and how long the device was moving, is necessary to achieve good use of the sensor. From Fig. 12 above this deviation can be read from the small bias in the answers in the question.

The second sensor we utilized was the light sensor. By registering the light levels in the room the mobile companion automatically adjusted the background colour on the mobile device. To avoid any end-user frustration the mobile companion changed colours very gradually, using fade effects during transitions. This feature worked well in the user experiment and was well received from the test groups.

5.3. Context-aware information management

Context-aware information management and integration is very important for the enhanced user experience and automatically tailoring of the phone application for the user, and this is certainly the case for the mobile companion application at the heart of our work described in this paper. By integrating multi-dimensional information from relatively disparate sources as previously described, the mobile companion application is able to compute a user context. The combination of local phone information (i.e. GPS location and sensors), together with cloud context information (e.g. Google services) and information on user behaviour/social settings make for a very broad context-aware foundation. The mobile companion application is thus able to respond pro-actively to a user’s need, without s/he having to take explicit actions to get the device to adapt. Indeed, from the users’ responses in our evaluation we see that an overwhelming majority found the automatically adaptation of the phone application to be a desirable feature (Fig. 13).

6. Conclusion

In this article we introduced the concept of mobile companionship, and explored it in a design research project. We believe that sensitive and responsible applications should be at the core of mobile development. They provide for a supportive user experience, in contrast to a function-rich technology. After all, a companion provides a space of shared knowledge where many things are implicit and handled in effortless ways. In technical terms, this is what is expressed in context aware information management.

By integrating multi-dimensional information from relatively disparate sources the mobile companion application is able to
compute a user context and manage information for the user benefit. The combination of local phone information (i.e. GPS location and sensors), together with cloud context information (e.g. Google services) and information on user behaviour/social settings make for a very broad context-aware foundation. Cloud services have not previously been integrated in context aware mobile applications, and we have shown the feasibility of integrating such services in context aware applications. We found that users respond positively to context aware tailoring of the interfaces on their mobile device. Another aspect of companionship is trust, and we also found that users are not overly concerned with storing limited amounts of personal information on the cloud.

Our mobile companion application is not a completed product, but a prototype designed to show an avenue for further research. Overall, we argue that our approach increases the level of mobile companionship. The mobile experience is easier, friendlier and more sensitive to situational contexts, which are key criteria of companionship.

References


Tor-Morten Grønli received a Masters degree in distributed information systems and computing in 2008 and was awarded a PhD in 2012 for research conducted on cloud computing and context-aware mobile computing - both from Brunel University, Uxbridge, UK. He is currently an associate professor at the Norwegian School of IT. His current research interests include mobile solutions, pervasive and ubiquitous computing, context-aware systems and software engineering.

Gheorghita Chinea received the BSc and BSc (Hons) degrees in Computer Science and Mathematics, in 1993 and 1994, respectively, and the MSc degree in Computer Science, in 1996, from the University of the Witwatersrand, Johannesburg, South Africa; he then received the PhD degree in Computer Science from the University of Reading, United Kingdom, in 2000. He is a Reader in the School of Information Systems and Computing at Brunel University, United Kingdom. Dr. Chinea has over 100 refereed publications in his research interests, which span human computer interaction, semantic media management, and perceptual multimedia networking.

Bendik Bygstad holds a PhD in computer science and a Master of Sociology. He worked 15 years in the IT industry. He is currently a professor at the Norwegian School of Information Technology. His main research interest is IT-based innovation and IS research methods. He has published articles in journals such as Information Systems Journal, Journal of Information Technology and International Journal of Project Management.