Mobility, Context and Cloud – Exploring Integration Issues in a Meeting Room Scenario

http://dx.doi.org/10.3991/ijim.v6i2.1927

T.M. Grønli[1,2], J. Hansen[2] and G. Ghinea[1,2]
[1] NITH, Oslo Norway
[2] Brunel University, London, UK

***Abstract*—In this paper we pursue the context-aware paradigm in a distributed mobile environment. Context-awareness plays a significant role in the domain of implicit human computer interaction and we present an intelligent context-aware meeting room application. The distributed mobile application consists of multiple mobile operating system clients, a Google Android based presenter that, through proximity technology, communicates with a server in the meeting room, and a Google Cloud backend server. Over 40 users have evaluated the application. Results show that the context aware meeting room was well received by users who displayed a strong positive bias towards cloud integration and, in particular, automatic meeting-note push. We conclude that our context-aware meeting assistant can provide valuable assistance to end users and that there is promise for such use of heterogeneous mobile context-aware platforms.**

***Index Terms*—Distributed Information Systems, Computer Applications, Mobile Communication and Software, Context-aware**

I. INTRODUCTION

Context-sensitive (or aware) applications have, in recent years, moved from the realm of possibilities to that of ubiquity. One exciting research area that is still very much in the realm of possibilities is that of cloud computing. In this paper we present our work, which explores the overlap of these two research areas, specifically through the prism of an application integrating heterogeneous mobile devices in a meeting room scenario.

There are several contributions that focus on context aware applications in a meeting. Chen et al. [1] propose the use of Bluetooth by creating a smart meeting room. The idea is to provide relevant services and information to the meeting participants based on their context. The services include presentation-, lighting control- and greeting service, where Bluetooth is used to register the participants that are in the meeting room. Another research contribution investigating the aspects of localization and identification is that of Bourcier et al. [2], who focus on the notion of service oriented computing in the meaning of automated discovery and connection to devices. Through the work presented, Bourcier et al. [2] highlight the importance of connecting heterogeneous devices through a generic protocol and strategy.

Smart meeting rooms can also assist with managing the schedule of the meeting attendees, updating related documents during the meeting and providing a communication facility. They may also filter out information based on user groups, where part of the information is only received by a certain group of users. To this end, Ahmed et al. [3] present the design of a smart meeting room that not only detects the beginning and end of a meeting, but also provides support of participants' context information, knowledge usability and ephemeral group communication. Group members can share resources within the same cluster, at the same time maintaining privacy and confidentiality. Here, the context and situation information in the meeting are aggregated using location-tracking devices. The emphasis is on making meeting attendees free from worrying about various lower level details which includes changing cell phone ringer mode, transferring confidential documents to authorized persons and preparing and sending meeting minutes [3]. Their proposed solution uses the Microsoft .NET compact framework.

In a smart meeting room context, it is especially important that computing devices need to be transparent to the users and the services should be non-intrusive. In this spirit, Dai and Xu [4] present a Dynamic Context Model to solve the problem of context awareness toward group meeting analysis and services, which includes multimodal analysis of group interaction scenarios and provision of attentive services to the users. They aim to achieve online analysis and services during group meetings. The objective is to make a computing system understand the current context of group meetings based on the audio-visual information at each step and provide intelligent services to the participants instantly. Inside the meeting room, cameras and microphones are placed to collect information during the meeting.

Indeed, in a meeting room scenario, there are a variety of applications in which a sound source localization system may be useful. Automatic translation to another language, retrieval of specific topics and summarization of meetings in a human-readable form are a few of the possibilities. Moreover, one can identify a participant who is speaking at a particular time, as well as performing speech activity detection and source localization, all briefly described by Parviainen et al. [5] in the context of the system developed in their research. The data from their work are comprised of meeting sessions that took place in a room equipped with microphones and other recording hardware. The system is based on spatially separate sensor stations, with each sensor being able to determine the direction of arrival of a sound source. The results from the test sessions were good, but there are still a few remaining issues like integrating the solution with a speech activity detection technique [5].

There are also attempts to take the smart meeting rooms even further by adding the possibility for a virtual meeting. Nijholt et al. [6], for instance, discuss how to go from captured data in a smart meeting room situation to a virtual meeting room. They look at issues including turn taking and gaze behaviour of meeting participants, influence and talkativeness, and virtual embodied representations of meeting participants. Their work proposes a visualization of meeting events in virtual reality, interpretations of these meeting events in order to produce semantic preserving transformations and presentation of these meeting events using various media sources. The idea is that virtual meeting participants can mimic what is happening in the physical meeting room. The technology used within the virtual meeting room was considerably different than in normal video conferencing by sending the data in a format that enabled analysis and transformation.

Microsoft research on integrating context into applications exploiting desktop calendars can be found in the work of Cutrell et al. [7], who created an application integrated with the Microsoft Outlook calendar on a desktop computer. What is very interesting in this approach is that they introduce a tag library to tag email and file elements with a context label meaningful to the user. By using labels from a context familiar to the user, they attempt to ease the search and filtering of the tagged objects. A disadvantage of their application, however, is that with such a huge tag library, it takes a lot of time to maintain it. In addition new elements are not discovered in searches before they are tagged properly, which then entails a time penalty on new system objects.

Closely related to our work with context aware applications for mobile phones in a meeting environment, Göker et al. [8] describe a context-aware information system intended for mobile users. Their research demonstrates a special-purpose hardware device, called '*context tags*', which can work with mobile devices such as mobile phones, to provide ambient information to users on the move. A tag consists of hardware and software and it can detect the proximity of handheld devices [8]. The result is that relevant content can automatically be distributed and delivered onto each mobile phone in the vicinity of physical objects, rooms, and open areas. Context is in general constructed through the user profiles (typically local to a device) and the wireless tags can be embedded in the surroundings. The overall system architecture consists of three main cornerstones: the content service provider, the mobile user, and the context tag.

OneBusAway, a suite of transit traveler information tools provides real-time arrival information [9], a trip planner, a schedule and route browser, and a transit-friendly destination finder for Seattle-area bus riders. Ferris et al. [9] developed a location-aware native iPhone application for OneBusAway that leverages the localization technology in modern mobile devices. Their application communicates with a OneBusAway back-end server over the phone's network connection to request information about stops in a given area, information about particular routes, and, ultimately, real-time arrival information for specific stops.

From a context viewpoint the research presented in this paper differs from most of the other contributions by only using standard hardware (Bluetooth, 802.11b/g (wireless network), mobile phones and a server computer), and not needing cameras, microphones and sensors to detect the context of the meeting participants. By combining this approach with support for multiple platforms we are able to implement the system in a heterogeneous environment. Moreover, we use the Google cloud to facilitate a range of different devices.

Cloud computing focuses on sharing data and makes it possible to distribute storage and computations over a scalable network of nodes, with large IT companies like Microsoft, Google and IBM, all having initiatives relating to cloud computing [10]. One of the key features is under this paradigm is scalability on demand, where the user pays for the amount of computation and storage that is actually used. Moreover, this scalability is usually completely transparent to the user. Mei et al. [10] have focused on finding interesting topics for future research in the area of cloud computing, and have drawn analogies between cloud computing and service and pervasive computing. They identify four main research areas: 1) *pluggable computing entities*, 2) *data access transparency*, 3) *adaptive behaviour of cloud applications* and 4) *automatic discovery of application quality*. Our work is closely related to adaptive behaviour of cloud computing applications. In our work, we implement context-awareness in a mobile environment, described in detail in Section 2, and integrate it with cloud computing.

One important issue with cloud computing is privacy. Data is often stored on external servers that one has no control over. Mowbray and Pearson [11] have looked at how it can be 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. This is implemented using an obfuscation and de-obfuscation service to reduce the amount of sensitive information held within the cloud. They have also implemented a feature to allow users to express privacy preferences about the treatment of the personal information.

In our research effort we investigate the collaborate interplay of heterogeneous context-aware applications in a cloud environment. In answering this research question, we investigate a set of objectives:

- Cloud integration of mobile services
- Exploitation of context-aware information
- Technological fundament in Bluetooth ping and in protocol buffer implementations

Accordingly, the structure of this paper is as follows: Section II describes the architecture of our application; this is then followed by a walkthrough of the evaluation in Section III. Thereafter Section IV presents the results, whereas Section V discusses the research contribution; the limitations are acknowledged in Section VI and Section VII concludes the paper.

II. PROJECT ARCHITECTURE

A. *Scenario Description*

A daily activity at a work place is to attend meetings. Moreover, this activity includes the workload necessary in the administration of the meetings. Not only does one have to attend a meeting, but also s/he is also required to keep track of the daily schedule, meeting subjects, write meeting minutes and more. Applications like Microsoft Outlook or Lotus Notes can help with some of the tasks, like keeping track of the schedule and administrating meeting invitations and responses. Still, when it comes to meeting minutes and direct involvement in the presentation, existing tools come short. This can be changed by the introduction of a context-aware meeting assistant. As the meeting participant walks around in the office landscape s/he approaches the room for an upcoming meeting. In the room, a server application is constantly on the lookout for devices in close range that should be included as meeting participants. Through the use of proximity technology, the server discovers the user's device and identifies if s/he is a meeting participant. The server module checks the device and automatically enables the use of intelligent meeting assistant in this meeting and welcomes the person to the meeting. Some time passes and all participants find their place at the meeting table. The presenter welcomes the audience and starts his presentation by moving a slide forward. Our users' devices then wake to life as the intelligent meeting assistant kicks in. When the presenter changes a slide, the notes for the new slide are sent to the users´ devices. The interface on the device consists of opportunities for moving forward and backward in the notes received so far. This enables a user to go back and look at previous key points as well as continuously keep track of the main ideas at the current slide. The intelligent meeting assistant also facilitates for storing the notes received so that you are able to have on device capabilities for browsing meeting minutes from older meetings.

B. *Application Design*

The application consists of three major parts. The first two are a mobile Android presenter application that communicates with a server application in the Google Cloud. The clients used by the meeting participants give the third part. These are made up of a mobile Java ME, Windows Mobile, Android and iPhone applications.

The different mobile phone applications are all connected by a server running on a normal desktop / laptop computer using both Bluetooth and wireless network (WLAN) to send the information. The application detects when a user is within the proximity of the server and if that user is registered for a meeting or not. When the meeting has started the presenter will be able to control the presentation with the Android presenter application. This application includes features for selecting next and previous slide to be displayed for the participants. When the next slide is shown, all connected participants will automatically receive small notes of text that accompany the current slide.

The server application is deployed in, as well as integrated with, the Google Cloud, using the online calendar feature to find upcoming meetings and registered presenter/participants. All calendar events are registered with the Google calendar web interface and are downloaded to our application using the Google calendar API.

The role of the Android application is to facilitate the presenter's role during the meeting. The presenter will use the application as a remote tool to control the presentation shown in the meeting. The application contains functions for moving back and forth in the presentation as well as starting and ending it. As previously mentioned, the meeting participants will receive the slide notes for each new slide displayed. This is fulfilled by the meeting room server application and the Google App Engine cloud application, both written in Java SE, and the notes are triggered when an event is sent from the Android presenter client. When the new slide is shown, the cloud server application will find the notes for the current slide from the presentation stored in Google Docs and distribute it to the connected meeting participants.

The common client is the application for the meeting participants. This application is started manually and then the device is recognized by the server application through proximity technology and an active Bluetooth mode on the device. When the client gets in reach of an upcoming meeting and is identified as a participant by the server, the context-aware meeting assistant application is automatically connected. The application gives the client possibilities for maintaining a connection to the server as well as receiving meeting information. The client can browse back and forth in the received notes, but never further than the most recent slide displayed.

The server application consists of two parts: the main server application containing the majority of the business logic running on Google App Engine and the standalone server application running in the meeting room. The Google App Engine based web-server application is responsible for storing user information to the database and integrating with external systems like Google Calendar and Google Docs. By taking advantage of the Google App Engines' resources our application is able to scale to meet huge traffic demands. The meeting room application, on the other hand, runs as a standalone service on a machine placed in the meeting room. The main purpose of this application is to connect clients as they show up within Bluetooth range and to distribute presentation information (the notes) for the presenter. The server maintains an updated list over participants for each meeting by sending requests to the Google App Engine cloud server. The cloud-based server will poll the calendar service as well as the application database to retrieve the meeting information. This means the meeting room server can list all upcoming appointments for a given time frame and when selecting a meeting, it can list all registered clients for that meeting. The clients are identified by the use of Bluetooth as proximity identification. The server is constantly looking for clients to connect to, and, on finding a match, the client is wired up if identified as a meeting participant. All clients as well as the presenter use wireless network (WLAN) and the HTTP protocol for the network communication. When the presenter starts the slide show, this is picked up and all clients are pushed the meeting note for the upcoming slide.

All this information collaboration and exchange is possible due to common formats by Google Inc, with the Google Calendar being the main source of information. This service is used both for entering appointments, inviting participants and keeping track of the upcoming schedule. All information transferred between presenter and server and client and server are transferred using the *Google protocol buffer* technology. This enables efficient exploitation of bandwidth and creates the opportunity for cross language and cross platform sharing of information. The sharing of information is made possible by the format of the protocol buffer messages and their predefined message structure.

The server UI includes a feature that lists the upcoming meetings. When the presenter is setting up for the meeting s/he will select the upcoming meeting and press the '*connect users*' button. This will let the server know a list of users (Bluetooth addresses) that it is going to connect to. The users registered for the meeting are the only ones that will be automatically started, and all other active Bluetooth devices will be ignored. This also has the added benefit of not starting the participants if the meeting is cancelled. If there are any changes or new meetings the UI has a button for refreshing/updating information from Google calendar.

C. *Technical Architecture*

The application architecture consists of a client/server model. One of the major decisions was to use a common network protocol, protocol buffers, for all platforms including the server and client applications. Protocol buffers [12] are a contract first network protocol supporting a large number of platforms and programming languages [13].

Protocol buffer messages are reported to provide considerable performance improvements over XML Web Services [14]. For Java ME, we created a new protocol buffer implementation since there were no implementations supporting this platform. For iPhone, Android and Windows Mobile there were already open source libraries available.

As previously described, the discovery and identification of meeting participants are done over Bluetooth communication. After this initial phase the clients have received the servers address and communication moves to the HTTP protocol. The client then performs a HTTP Get request to the address provided by the server and a protocol buffer file with a welcome message is received. Fig. 1 above describes the architecture and communication lines between modules. The presenter client also uses HTTP communication by sending a POST request containing a PresenterEvent protocol buffer object. In the remainder of this section, we will give a short introduction to the architecture and challenges of each of the four implementations.

D. *Presenter Application*

For the presenter client, a POST request containing the PresenterEvent protocol buffer message was sent when the user pushed a button on the phone. The client consisted of a very simple user interface containing two buttons that produced two events, namely next or previous slide. These messages were transformed on the server to actions, moving to either the next or previous slide in the presentation. We did not include any sign-on process for the presenter, since it was outside the scope of this research experiment, but for real world use of this would have to be added. The presenter client was implemented in Java for Android devices using the 1.5 Android SDK

E. *Participant Application*

The participant application was developed with 4 native clients; Android, Java ME, iPhone and Windows Mobile. All clients offered the same functionality. There are two main functions; 1) *receive meeting notes during a meeting, the device will automatically update when the presenter changes the slide*, and 2) *meeting notes history where all previous meetings are stored locally on the device*. Each meeting note sent to the devices is transferred across WLAN with the protocol buffer format. The clients were designed around simplicity. We wanted the cloud application to handle the heavy load, the clients were simply responsible for showing the meeting notes to the users and keeping a history of previous meetings.

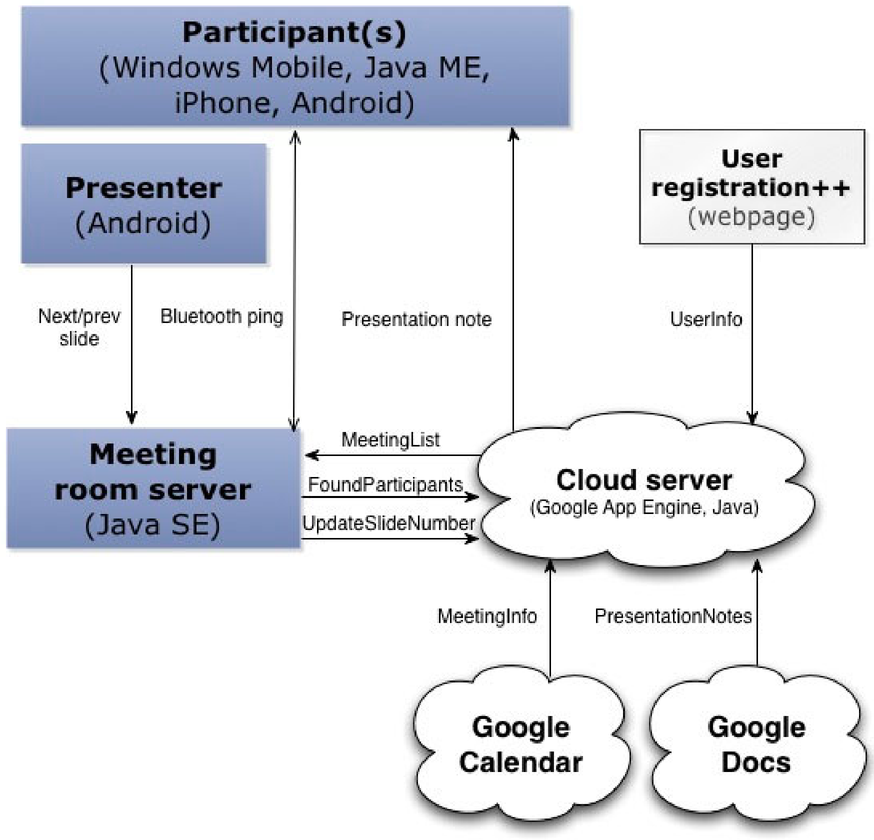

Figure 1. Application component architecture

F. *Meeting Room Server Implementation*

The server application was implemented running an embedded Jetty server. The meeting room server communicates with our cloud server application deployed in Google App Engine. The meeting room server is a multi-threaded application due to its many concurrent tasks. It continuously searches for Bluetooth devices while also sending out notes and keeping track of connected participants. The Bluetooth library used in our implementation was BlueCove [15] and the application was built using the Google Guice framework [16].

G. *Cloud Server Implementation – Google App Engine*

The notes for each slide sent to the participants are retrieved from actual presentations stored in Google docs. A Google API to access notes on a slide is lacking from Google at the moment and the technology used to acquire the slide note information is a custom written parsing implementation of the Google docs document. The step of harvesting information from the notes from Google docs is an important feature of the system architecture due to the fact that this loosely couples the presenter, the presentation and the machine the presentation is displayed from. The information related to meeting participants and schedules are stored in the Google cloud. The calendar integration was implemented using the Google Calendar API [17]. Using this API we were able to query Google for meeting information. To connect the Google Calendar with the correct presentation in Google Docs we used a system of context-aware XML tags to integrate the features.

III. EVALUATION

A. *Pilot Evaluation*

Before conducting our main evaluation we conducted a pilot evaluation with 12 users. By doing this, we were able to ensure technical stability and bug fix some minor issues as well. At this stage, we also unraveled two ambiguities in the user evaluation questionnaire, leading us to rephrase them appropriately.

B. *Participants*

Our context-aware meeting room prototype application was evaluated in several separate test sessions, with each session consisting of between five to eight participants. The details of these now follows. A total of 12 persons took part in the pilot evaluation and another 40 people participated in the formal evaluation. All of them were aged between 20 and 55. All participants had previous knowledge of mobile phones and mobile communication, but had not previously used the type of application employed in our experiment. From the user computer experience classification (asserted based on a questionnaire employing the taxonomy of McMurtrey [18]) we learned that the majority of the users had good knowledge in this area.

C. *Material*

In our tests, four different types of devices were present. The first was an *HTC Magic* device, running an Android operating system version 1.5, used both by the presenter and some participants in the evaluation. The second device, used by meeting participants, was a *Nokia E61 S60 3rd edition* running Symbian 9.1 operating system. The third device, also used by meeting participants, was a first generation *Apple iPhone* operating system v3.1.3. The final device, also used by the meeting participants, was an *HTC TYTN II* device, running Microsoft Windows Mobile 6.1 operating system. The server equipment included in this user evaluation was an Apple MacBook Pro with 2.4 GHz Intel Core 2 Duo with 4GB of DDR2 SDRAM. This machine was used for background tasks such as scanning for meeting participants and enabling the registration server services. In addition to the material described, we took advantage of the Google Cloud by deploying our backend application to a cloud computer service at the Google App Engine. The overall distribution of participants for a particular device is given in Table I.

TABLE I.
DISTRIBUTION OF MOBILE CLIENTS

Device		**Operating System**	**Number of participants using device**
iPhone 3G			5
HTC Magic			15
HTC Tytn II		Windows Mobile	10
Nokia E61		Java	10

D. *Process of User Evaluation*

Participants were presented with the application and then instructed to register their mobile device by entering Bluetooth id and their respective email addresses. Email addresses were used to keep a mapping between the Bluetooth id and the respective user. Thereafter, users approached the meeting room area, where the server used proximity information as well as information from the Google cloud to identify meeting participants. Those eligible for the meeting received information, a welcome message, on this in their context-aware meeting.

When all participants were present, the presentation started. During the 30-minute presentation, meeting notes from the 12 slides were pushed to the participants' devices. After the presentation, all participants were asked to answer a questionnaire to evaluate the exercise and application. Lastly, we mention that environmental variables were kept constant for all participants in the experiment, as the meeting room and its layout and lighting levels were unchanged from one evaluation session to the next. We would also like to mention that the IT-efficiency of

TABLE II.
QUESTIONNAIRE

Number	Statement
User Interface	
S1	It is easy to see the available functions
S2	The features of the application are hard to use
Performance	
S3	Browsing back and forward in the current meeting meeting-notes is quick
S4	The presentation notes displayed on the mobile device was synchronised with the audio presentation/presenter's narrative
User Features	
S5	I was not able to register as a participant in the web application
S6	I was able to look at next and previous notes during the presentation
Context-Aware Information	
S7	When the presenter started the meeting my mobile device was found and I began receiving meeting notes once in presentation mode
S8	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
S9	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.
S10	Storing meeting information in the Google cloud and combining this with personal information on the device is a useful feature.
Application Features	
S11	I find the ability to navigate notes during the presentation useful
S12	I do not find the "*meeting note*" history functionality useful
S13	I do not find the presenter/participant model useful for a meeting environment
S14	I would like to receive more content during the presentation (images, videos, files, web-pages etc)
Overall Impression	
S15	I find the context-aware meeting room application useful

participants was not the focus of the exercise, with participants being made aware that their skills/ability to interact with the device was not the issue.

E. Questionnaire

The questionnaire for evaluating the experiment consisted of 15 statements divided into six categories, as shown in Table II.

In order to reduce potential bias, the questionnaire consisted of a roughly equal amount of positive and negative statements randomly divided between the categories. For each statement answered the users were asked to indicate, on a four point Likert scale of 1-4, their opinions (ranging from '*1- Strongly Agree*' to '*4- Strongly Disagree*') with respect to the statement regarding the respective task / functionality. In the final part of the questionnaire, an open-ended request for comments and feedback were solicited from the participants. When presented to the participants the questions were presented in a random order.

TABLE III.
RESULTS

Question Number	Mean	Std. Deviation	T-test Value
1	3.58	.549	.000
2	1.50	.555	.000
3	3.58	.549	.000
4	3.13	.686	.000
5	1.25	.588	.000
6	3.78	.480	.000
7	3.83	.385	.000
8	1.83	.903	.000
9	2.08	.694	.000
10	3.38	.490	.000
11	3.55	.504	.000
12	1.75	.630	.000
13	1.68	.616	.000
14	3.53	.640	.000
15	3.33	.572	.000

F. Analysis

The data collected were analyzed with the IBM Statistical Package for Social Sciences (SPSS) for PC (release 15.0). The mean, standard derivation and significance were collected from the responses. A t-test and Analysis of Variance (ANOVA) respectively suited to identify statistically significant biases, and to test potential significant differences between the four user devices [19], were used to analyse the participants' responses. Moreover, we also used SPSS to perform Principal Component Analysis (PCA) in order to reduce the dimensionality of our data and extract common themes. A significance level of $p < 0.05$ and factor loadings greater than 0.4 were adopted for the study.

IV. RESULTS

Overall, in respect of user interface, performance, user features, context-aware information, application features and overall impression our results show a strong positive bias (Table III). From the results, we will remark that a one-way ANOVA found no statistically significant differences in the answers to questions where the operating systems used in the mobile device was the independent variable. We will in the following sections comment in detail on the results and their implications.

A. User Interface

The statements dealing with user interface, question 1 and question 2, show that the participants found the user interface easy to use – see fig 6 below. Responses to question 1 reveal that 39 out of 40 participants found it easy to see the functions available in the client interface. Following this, the participants were asked to evaluate if the features were hard to use, with 39 out of 40 respondents rejecting this statement.

In total, this indicates that the user interface is currently well suited for user interaction and not a hindrance for the application. One can argue that this is to be expected, as it is a relatively simple user interface with a limited number of functions and maneuverability.

B. *Performance*

Question 3 covered the speed of which it was possible to flip through the already received meeting notes. To this statement there was a strong positive bias (a mean rating of 3.6 out of 5) indicating the functionality to work properly. Question 4 dealt with the sync between the narrative of the presenter and the information sent to the mobile devices. This issue is quite important since the participant will expect the information on the device to keep being updated as slides change and they will not tolerate a greater delay. Also here we see a strong positive bias (3.1 out of 5) when answering this question, but 3 (out of 40) positioned themselves as negative to this statement. This is only a minor group, but still important of being aware of due to the importance of this functionality. The negative statements could also be due to network connection faults or other latency in network traffic. The numbers are illustrated in below (Fig. 3)

When asked to take a stand to the question regarding not being able to register as a participant, 39 people rejected this statement (Fig. 4). With only one person agreeing, this describes a situation where the web registration works as expected. Indeed network error and latency were behind the inability of this individual to register. Fallback mechanisms, however, secured this participant's further participation in the evaluation. Question 6 covered whether the meeting notes from the presenter were received and if they were able to flip through the notes received. To this statement there was again a strong positive bias (3.8 points out of 5) indicating the functionality to work properly.

C. *Context-aware Information*

In terms of context-aware information the participants were asked to take a stand to four statements with results shown below (Fig. 5) (question 7 to question 10). Firstly, they were asked whether the application started receiving information when the presenter started the meeting. Respondents were unanimous in giving positive answers to this, showing that the behind the scenes cloud integration worked properly and provided information output to only the authorized participants.

The system is tightly integrated with the Google cloud using Java libraries for connecting to the Google Calendar and retrieving the user information. The advantage of this solution is that we have an advanced calendar application already available with all the functionality that Google offers. All updates to the web interface of the calendar system can be immediately taken advantage of by the users our system. Also, new API functionality will be available in the future with an update of the runtime library provided by Google that will open for new opportunities. Two questions covered this topic, asking the participants to take a stand vis a vis the close integration with these services and whether or not they felt uneasy regarding the amount of personal information stored. Somewhat surprisingly, bearing in mind potential privacy concerns that could arise in this respect, almost all evaluators (both in pilot evaluation and formal evaluation) agreed they are comfortable with the meeting information being stored remotely by Google in the Google service cloud.

D. *Application Features*

Participants also had to position themselves in respect of four questions about the application features. Results are presented in (Fig. 6). Here, statements 11 and 12 dealt with situations of usefulness from the application. All participants agreed that the ability to navigate notes during the presentation was useful. This is also backed up by earlier statements dealing with whether this functionality worked. 38 out 40 asked also answers negatively to statement 12, building even more confidence for the navigation functions included.

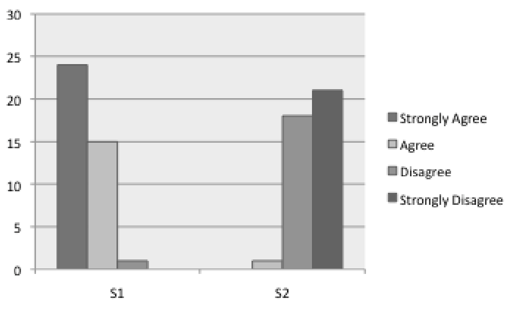

Figure 2. User Interface results

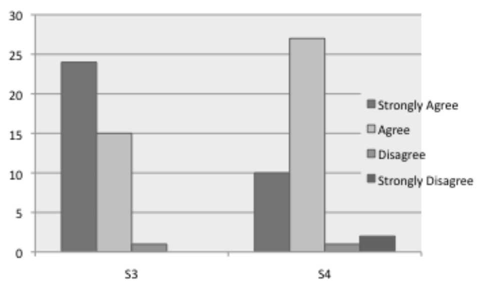

Figure 3. Performance results

E. *User Features*

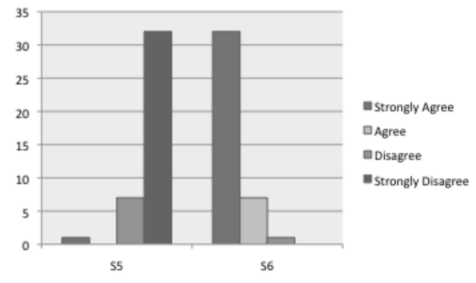

Figure 4. User Features results

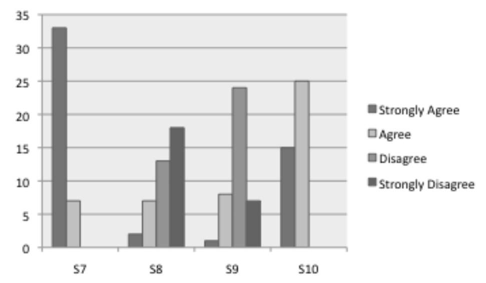

Figure 5. Context-aware Information results

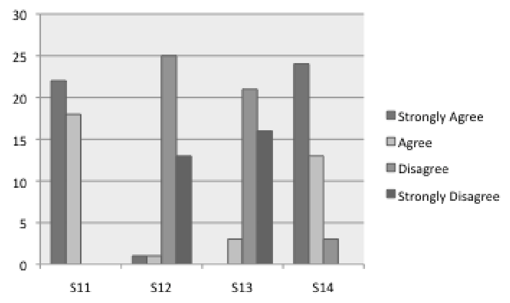

Figure 6. Application Features results

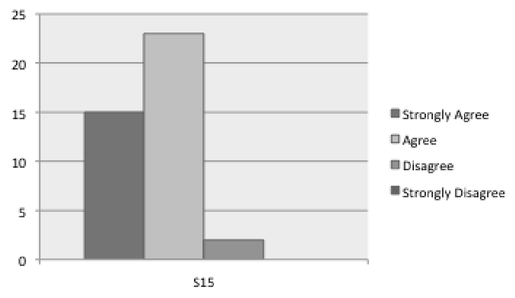

Figure 7. Overall Impression results

In statements 13 and 14 the applicability of the application in a meeting room setting was tested. As expected from the answers collected in the pilot evaluation, the application is per se identified as useful. However, the ability to receive more complex data such as images and its likes is highly wished for. We acknowledge this demand and think it could be a useful extension to further improve the usefulness of the application and thereby broaden its' acceptance.

F. *Overall Impression*

The overall impression from the users evaluating the application is very positive as seen in Fig. 7, with a large majority of users finding the application useful. Taking into account statements from the evaluation group, we see that the requests for a more diverse type of content to be transferred to the mobile devices as well as an even richer interface could be directions to even further improve an application, notwithstanding its already strongly positive evaluation.

V. DISCUSSION

We will in the following section inspect the implications of our work in relation to the general literature. In relation to context-aware computing we see that our work extends and build on the work of Chen et al. [1]. They propose the use of Bluetooth to create a smart meeting room and pursue the idea to provide relevant services and information to the meeting participants based on their context. Our work extends those aspects even further by using context as the primary factor for presenting information, the meeting notes. Ahmed et al. [3] present an interesting design of a smart meeting room that not only detects the beginning and end of a meeting but also extends context-aware aspects into the users cluster. Our work distinguishes from this by not only building on the principles of information sharing within a cluster but also extending the rules on which the cluster is defined. We use separate context-aware dimensions, such as proximity, availability and user status to create an ad hoc composition of user clusters in the meeting rooms and within these dynamically share information – the meeting notes. From answers given to statements 7, 11 and 13 we see a clearly positive bias to these issues. Cutrell et al.'s [7] work on calendar integration and the work of context-tags from Göker et al. [8] are both used as a foundation in our work. We see that our contribution extends this by combining the two approaches by tagging context-aware information in the Google calendar. Additionally we take their views one step further by exploiting the cloud services from Google as the center point of integration. Moreover, results from statements 4, 13 and 15 of the user evaluation reveal that users as well see this as an aspect enriching their collaborative work.

Mei et al. [10] suggest adaptive behaviour of cloud applications as one topic to be pursued in research relating to cloud computing. Our work extends this research topic by integrating context awareness into cloud services. Both proximity technology and user status computation are such context-aware dimensions we exploit. Security is an important aspect in cloud applications. Mowbray and Pearson [11], in their research, explore the opportunity of having a client-based privacy manager and highlight how this helps users control their sensitive information in such applications. We acknowledge this an important matter to include in research, but our efforts have not yet pursued this aspect. Despite this, we do see from the users responses in statements 8, 9 and 10 that the close integration with the cloud and the remote storage of personal information is not perceived as inconvenient. Even though the literature explored generally finds cloud computing an exciting trend in computing technology, there are some that have expressed negative opinions. Richard Stallman, creator of the GNU project and founder of the Free Software Foundation, has, for instance, said the cloud computing is a trap to force people into buying locked proprietary systems.

Although there can be negative impacts, like vendor lock-in, of cloud computing, we feel that it is an important step forward. Many people today are already using cloud services without thinking about it, like webmail (Google Mail), online documents (Google Docs) and backup systems (Dropbox filesharing). The work done by Google App Engine is especially important since it provides a service where developers can deploy their application into a standard platform, either Python or Java, which does not contain any vendor lock-ins.

In the entire system we used Protocol Buffers to handle the network communication. For the Java ME participant client there was no Protocol Buffer implementation available. One contribution from our work on the context-aware meeting room is that we created an implementation designed for Java ME that is compatible with the Google Protocol Buffers. This was made as an open source project and is currently available on Google code. The overall performance of the system, evaluated by the users in questions 3 and 4, was satisfactory. It is important to consider that the performance of the network communication in our tests was not only affected by the Protocol Buffer implementation, but also by the fact that we were running our application on a wireless network and sent and received network traffic from the Google App Engine servers.

To register the meeting participants we created a feature we call *Bluetooth ping*. This contribution was created because of the limitations on both the Android and iPhone platform. We did initially start with a normal Bluetooth search. This is a well-known feature that will find all nearby devices that have been configured as discoverable. The problem with the Android and iPhone devices was that they only could be set to discoverable for a limited amount of time. This became a problem for the users of our client application because they needed to make the phone discoverable each time they entered the meeting room. One of the main features of the system was to use the meeting participants context to automatically provide certain features, as we did not want the users to have to configure the Bluetooth settings before each meeting. Since we already registered the Bluetooth addresses of the mobile devices we were able to take advantage of a predefined Bluetooth service that is available on most new mobile phones. By pinging the *Bluetooth hands free service* for all the meeting participants we could find the nearby users without doing a standard search. This has the benefits of finding devices that are hidden, meaning that Android and iPhone only had to turn on Bluetooth and nothing else, and will not use time on unregistered devices since it only targets the registered addresses. In the user evaluation this method worked reliably, but we did experience some differences in the total time used for finding all registered devices. In some cases the entire search was completed in 20-30 seconds; however we did also experience delays of up to about 3-4 minutes. Our assessment is that this is more than acceptable for a meeting scenario where the Bluetooth ping feature will be running in the background. This is also verified by the user evaluation (question 7), which highlighted that users were mostly happy with the ability of our application to discover and automatically register mobile devices and then send them the corresponding meeting notes.

VI. LIMITATIONS

A possible limitation in the evaluation is the close connection to the Google cloud. Although this means a high density of available services, this also means vulnerability to service down time. With a high volume number of evaluation participants (100+), the results might have been influenced by such a factor. Nonetheless, we think that our results give a decent and valuable insight into the world of mobile context-aware systems.

VII. CONCLUSION

In this paper, we have presented work that takes advantage of the Google cloud paradigm to integrate heterogeneous mobile operating platforms via a proof-of-concept context-aware application. The developed context-aware application implements a meeting room scenario, where the system exploits context-aware information, proximity and cloud technologies. The system distributes presentation notes to all connected meeting participants when the presenter changes the current slide. As the results from the user experiment show, the context aware meeting room was well received by the users who evaluated our application. The strong positive bias shows that there is mileage for the ideas presented in this paper to move beyond the prototypical stage. Indeed, users appreciated cloud integration and automatic meeting-note push particularly. From the results we identified and discussed guidelines for five areas: *participation*, *context*, *performance*, *content* and *information sharing*.

Last but not least, multimedia meeting notes and tuning of cloud integration are noted as possible further improvements and can be pursued in future work. Both are worthy pursuits in efforts to integrate cloud and context.

REFERENCES

[1] H. Chen, F. Perich, D. Chakraborty, T. Finin, and A. Joshi, “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*, Washington, DC, USA, 2004, pp. 854–861.

[2] J. Bourcier, C. Escoffier, and P. Lalanda, “Implementing Home-Control Applications on Service Platform,” in *4th IEEE Consumer Communications and Networking Conference, 2007. CCNC 2007*, 2007, pp. 925-929. http://dx.doi.org/10.1109/CCNC.2007.187

[3] S. Ahmed, M. Sharmin, and S. I. Ahamed, “A Smart Meeting Room with Pervasive Computing Technologies,” in *Proceedings of the Sixth International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks*, Washington, DC, USA, 2005, pp. 366–371.

[4] P. Dai and G. Xu, “Context-aware computing for assistive meeting system,” in *Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, New York, NY, USA, 2008, pp. 4:1–4:7.

[5] M. Parviainen, T. Pirinen, and P. Pertilä, “A Speaker Localization System for Lecture Room Environment,” in *Machine Learning for Multimodal Interaction*, vol. 4299, S. Renals, S. Bengio, and J. G. Fiscus, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 225-235. http://dx.doi.org/10.1007/11965152_20

[6] A. Nijholt, R. Rienks, J. Zwiers, and D. Reidsma, “Online and off-line visualization of meeting information and meeting support,” *Vis. Comput.*, vol. 22, no. 12, pp. 965–976, Nov. 2006. http://dx.doi.org/10.1007/s00371-006-0041-3

[7] E. Cutrell, D. C. Robbins, S. T. Dumais, and R. Sarin, “Fast, Flexible Filtering with Phlat - Personal Search and Organization Made Easy,” *IN PROC. CHI 2006*, vol. 2006, p. 261--270, 2006.

[8] A. Göker et al., “An ambient, personalised, and context-sensitive information system for mobile users,” in *Proceedings of the 2nd European Union symposium on Ambient intelligence*, New York, NY, USA, 2004, pp. 19–24.

[9] B. Ferris, K. Watkins, and A. Borning, “Location-Aware Tools for Improving Public Transit Usability,” *IEEE Pervasive Computing*, vol. 9, no. 1, pp. 13-19, Mar. 2010. http://dx.doi.org/10.1109/MPRV.2009.87

[10] Lijun Mei, W. K. Chan, and T. H. Tse, “A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues,” in *IEEE Asia-Pacific Services Computing Conference, 2008. APSCC '08*, 2008, pp. 464-469.

[11] M. Mowbray and S. Pearson, “A client-based privacy manager for cloud computing,” in *Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*, New York, NY, USA, 2009, pp. 5:1–5:8.

[12] Google, “Protocol Buffers,” *Protocol Buffers*. [Online]. Available: http://code.google.com/p/protobuf/. [Accessed: 21-Dec-2011].

[13] Google, “Third-Party Add-ons for Protocol Buffers,” *Third-Party Add-ons for Protocol Buffers*. [Online]. Available: http://code.google.com/p/protobuf/wiki/ThirdPartyAddOns. [Accessed: 21-Dec-2011].

[14] Google, “Developer Guide,” *Developer Guide*. [Online]. Available: http://code.google.com/apis/protocolbuffers/docs/overview.html. [Accessed: 21-Dec-2011].

[15] BlueCove Team, “BlueCove,” *BlueCove*. [Online]. Available: http://bluecove.org/. [Accessed: 21-Dec-2011].

[16] Google, “Guice,” *Guice*. [Online]. Available: http://code.google.com/p/google-guice/. [Accessed: 21-Dec-2011].

[17] Google, “Google Data Protocol,” *Google Data Protocol*. [Online]. Available: http://code.google.com/apis/gdata/. [Accessed: 21-Dec-2011].

[18] Katherine McMurtrey, “Defining the Out-of-the-Box Experience: A Case Study,” *EServer TC Library*, 01-Jan-2001. .

[19] P. Stephen and S. Hornby, *Simple Statistics for Library and Information Professionals*, 2nd ed. Library Association Publishing (UK), 1997.

AUTHORS

T.M. Grønli is a PhD candidate at Brunel University, London, UK. He is also with the Norwegian School of Information Technology, Oslo, Norway (e-mail: tmg@nith.no).

J. Hansen is a PhD student at Brunel University, London, UK (e-mail: jarle.hansen@brunel.ac.uk).

G. Ghinea is a Reader in Computing in the Department of Information Systems and Computing, Brunel University, UK and has a visiting position with the Norwegian School of Information Technology, Oslo, Norway (e-mail: george.ghinea@brunel.ac.uk).

Received 21 December 2011. Published as resubmitted by the authors 30 March 2012.