Pers Ubiquit Comput (2014) 18:883–894
DOI 10.1007/s00779-013-0698-3

ORIGINAL ARTICLE

Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing

Tor-Morten Grønli · Gheorghita Ghinea · Muhammad Younas

Received: 10 December 2012 / Accepted: 17 May 2013 / Published online: 28 June 2013

Abstract 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, and in this paper, we present our work, which explores the overlap of these two research areas. Accordingly, this paper explores the notion of cross-source integration of cloud-based, context-aware information in ubiquitous computing through a developed prototypical solution. Moreover, the described solution incorporates remote and automatic configuration of Android smartphones and advances the research area of context-aware information by harvesting information from several sources to build a rich foundation on which algorithms for context-aware computation can be based. Evaluation results show the viability of integrating and tailoring contextual information to provide users with timely, relevant and adapted application behaviour and content.

Keywords Cloud computing · Mobile computing · Context awareness · Information tailoring · Push messaging

T.-M. Grønli · G. Ghinea
Norwegian School of IT, Oslo, Norway
e-mail: tmg@nith.no

G. Ghinea
e-mail: george.ghinea@brunel.ac.uk

G. Ghinea
Brunel University, London, UK

M. Younas (✉)
Oxford Brookes University, Oxford, UK
e-mail: m.younas@brookes.ac.uk

1 Introduction

The notion of context-aware computing is generally the ability for the devices to adapt their behaviour to the surrounding environment, and ultimately enhancing usability [7]. In particular, context awareness is becoming an important factor when it comes to mobile devices, as users bring their smartphone or tablet just about everywhere, highlighting the need for adapting the content to the user's current situation. In a seminal article, Dey and Abowd [7] have remarked that if we fully understand context in a given environment and setting, we would be better able to adapt context-aware behaviour in our applications. The consequence of this desideratum 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 [2].

Context-aware solutions have been around for some years, and have been used in a variety of applications [1, 10, 11, 18, 19]. The issue with much of the earlier approaches is that they have either only looked at one source of context-aware information or, if more than one source was used, each was utilized separately; indeed, even if multiple sources of contextual information were pulled together, these were not integrated with a cloud infrastructure.

Cloud computing focuses on sharing data and makes it possible to distribute storage and computations over a scalable network of nodes. Large IT companies like Microsoft, Google and IBM, all have initiatives relating to cloud computing [5, 13, 14]. One of the key features 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. [13] highlighted

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 focuses on *data access transparency*, where clients transparently will push and pull for data from the cloud, and *adaptive behaviour of cloud applications*. We adapted the behaviour of the Google App Engine server application based on context information sent from the users' devices, thus integrating context and cloud on a mobile platform. Further, as advocated by Bellavista et al. [4], our approach provides facilities for run-time dynamic adaptation of context data according to the given conditions and situation.

The paper is structured as follows: the next section details and differentiates our approach from existing work; the design and development of a proof-of-concept application which showcases our ideas is then described. This is then evaluated and the results are presented and discussed in the context of related work in subsequent sections. Lastly, conclusions and possibilities for future work are described.

2 The proposed approach

There exist various models and architectures in order to manage and process context data. Bellavista et al. [4] give a detailed survey of such models and architectures and propose a logical architecture in order to discuss and analyse existing approaches in terms of context data distribution. As described above, our proposed approach focuses *data access transparency*, which is one of the important aspects of context data distribution. Though the logical architecture presented (and the related work surveyed) in Bellavista et al. [4] has some common features with our proposed approach, there are significant differences.

Our proposed approach utilizes web resources for effectively tailoring the user experience in cloud-based and context-aware mobile settings. To this end, we use the cloud infrastructure to combine context-aware information from several sources to customize and dynamically change a user's smartphone interface.

We chose a cloud-computing solution in order to have a feature rich, scalable and service-oriented server framework. Traditional REST framework services were considered [6], but they were found to be insufficiently scalable and extensible to add and remove context-aware sources in an ad hoc manner. The cloud-based approach also has the advantage of being run as a platform-as-a-service instance in the separate hosting instance of the Google App Engine.

2.1 Google App Engine

The Google App Engine makes it possible to run web applications on the Google infrastructure. App Engine applications are easy to build, maintain and scale [9]. The Google App Engine is a cloud-based platform as a service (PaaS); the platform is pre-configured by Google and provides a much higher abstraction than an infrastructure as a service (IaaS) platform such as the Amazon elastic cloud. The App Engine is well integrated with Google Apps services like docs, Gmail and authentication. Although the platform imposes certain limitations on developers (for example, no threading API), by enforcing these Google is able to provide very high scalability. The Google App Engine supports run-time environments for Java, Python and Go. Each environment provides standard protocols and common technologies for web application development [9].

The Google App Engine has been used as the main server installation for this research and has proven to be a stable environment for deploying the Java server applications without having to consider hardware requirements and software installations.

2.2 Meta-tagging and context computation

In our approach, users also have the added possibility to tag their appointments and contacts information on the Google cloud with context information, by employing special meta-tags. To this end, by adding a type tag, for example, $[type = work] or $[type = leisure], user would be able to indicate whether they had a business meeting or a leisure activity. We then filtered the contacts based on this information. If the tag $[type = 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. Conversely, 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.

With this information in place, in any given situation, the user context can be expressed as the combination of the user's activity (α), light conditions (β), social setting (γ) and geographical location (δ):

$$\text{User context} = f(\alpha, \beta, \gamma, \delta).$$

Based on this function, unique rules can be applied for each situation, and thereby have the system act differently and accordingly tailor the user's smartphone interface. In this function, user activity α is represented as an element collected from the entries in the user's calendar. This activity can then be further queried to collect information such as room, participants and agenda.

$\alpha \in \{$'collected calendar information'$\}$

The light conditions given by β are calculated by extracting information from the sensors of the device to measure light lux in the surroundings of the device and matching this with the device timestamp.

$\beta \in \{$'screen brightness harmonizing to timestamp and light lux'$\}$

The user's social setting, γ, is depicted by an activity/appointment extracted from the device or cloud calendar application. γ is represented in the form of a meta-tag appended to appointments and contacts. The details for γ are expressed below:

$\gamma \in \{\$[\text{type} = \text{'tag name'}]\}$

The final ingredient in the algorithm for computing the user context is the use of geographical position. The values were extracted from the GPS unit of the device and stored locally for use. A timestamp indicates the validity of the stored value. This is used to be able to be offline when obtaining a new position is impossible until a GPS fix again can be obtained.

$\delta \in \{$'GPS read position'$\}$

The running of the algorithm will provide several outputs and trigger actions on the device. Depending on the calculated result applications displayed can be changed, device screen brightness increased/decreased and contact repository updated.

Accordingly, in our work, by taking data from sensors and context-aware services and integrating it in a mobile application, we reduce the user workload and cognitive stress. Thus, in combination with a cloud-based server application, we are able to remotely configure smartphone interfaces independent of the device types. This creates a new concept of context awareness and embraces the user in ways previously unavailable.

3 Design and implementation

We have implemented an application suite, which provides a fully functional demonstration of the system. One of the main technical goals with the system was to make the interaction between the cloud and the mobile device as seamless as possible for the user.

The system was designed with three major components: an android client, a cloud server application and the remote Google services. Figure 1 gives an overview of the implementation of the system (blue boxes in the diagram represent the parts of the system we created). The white boxes, like Google calendar and contacts, are external systems we communicated with. The server application was deployed remotely in the cloud on the Google App

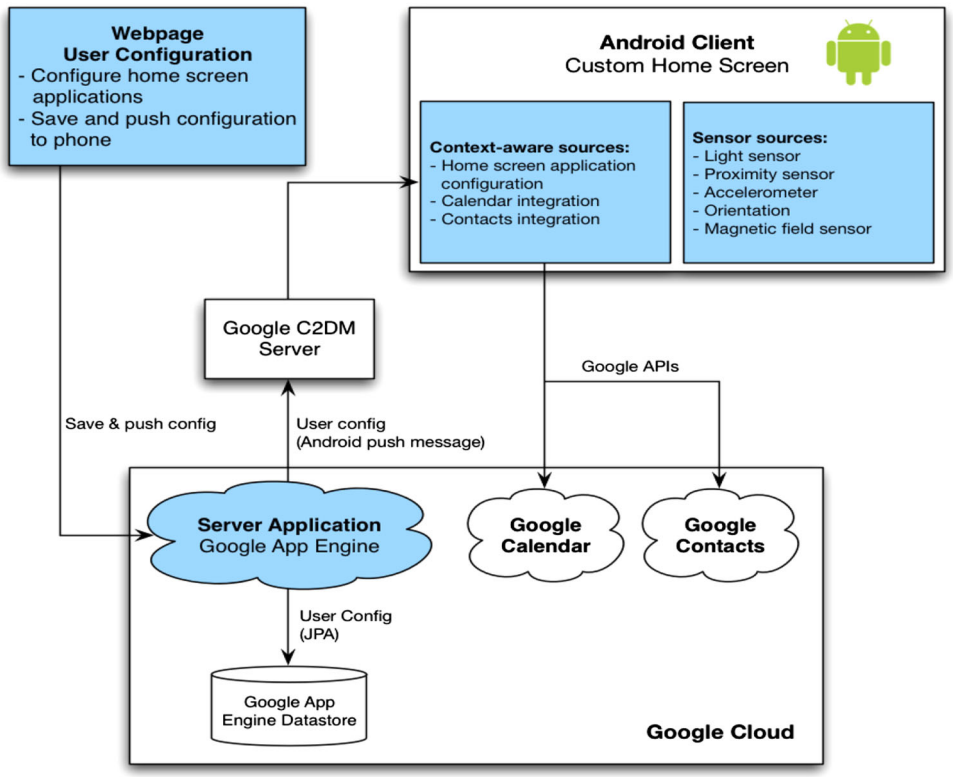


Fig. 1 Application suite architecture

Engine, whilst data were also stored remotely in Google cloud services. After the Android client was installed on the mobile device, the device will register itself to the Google services as illustrated in the code snippet of Fig. 3.

The users would start by logging into the webpage. 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 for the user sharing of their private calendar appointments and contacts stored in their Google cloud account without having to locally store their credentials. OAuth allowed us to use tokens as means of authentication and made it thereby possible for us to act as a third party granted access by the user. After a successful authentication, the user is presented with a webpage showing all configuration options.

Because the configuration for each user is stored in the cloud, we avoided tying it directly to a mobile device. One of the major benefits of this feature is that users did not need to manually update each device; they have a "master configuration" stored externally that can be directly pushed to their phone or tablet. It is also easier to add more advanced configuration options, enabling the user to take advantage of the bigger screen, mouse and keyboard of a desktop/laptop PC. This is done on the configuration webpage, by selecting the applications the user wants to store on the mobile device and pressing the "save configuration" button, the effect of which is to send a push message to the client application.

3.1 Android client

The client application was implemented on an Android device. This application utilized context-aware information from the device in the form of time, location and sensors. Additionally, it utilized context-aware information from the cloud-integrated backend to acquire dynamic interface content, contacts and calendar data. At launch, the application would look as illustrated in Fig. 2.

This interface would change depending on the user's given context. The available applications would be adapted and customized to match the current computed user context and thereby unobtrusively alter the user experience.

3.2 Cloud-to-device messaging

The adaptation message from the cloud to the smartphone 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. When a message needs

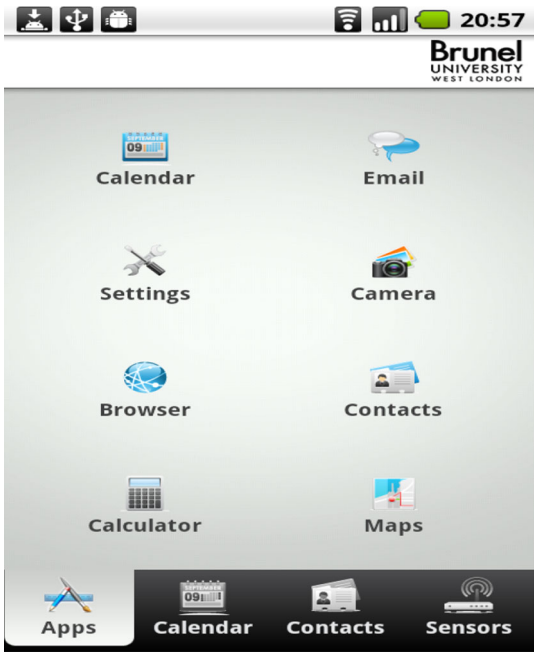


Fig. 2 Home screen interface at launch of application

to be sent, the "save configuration" button is pushed. We composed the message according to the C2DM format and sent it with the registration ID as the recipient. These messages are then received by the Google C2DM servers and finally transferred to the correct mobile device. A snippet from this process is shown below in Fig. 3.

The C2DM process is visualized in Fig. 4. This technology has a very few 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 future since the product is currently in an experimental stage, but it requires the developer to work with details like device registration and registration ID synchronization. Although C2DM does not provide any guarantees when it comes to the delivery or order of messages, we found the performance to be quite good in most of the cases. It is worth mentioning that we did see some very high spikes in response time for a few requests, but in the majority of cases, the clients received the responses within about half a second. Performance measurements we recorded, whilst doing the user experiments, were on average 663 ms of response value. It is also important to note that issues like network latency will affect the performance results.

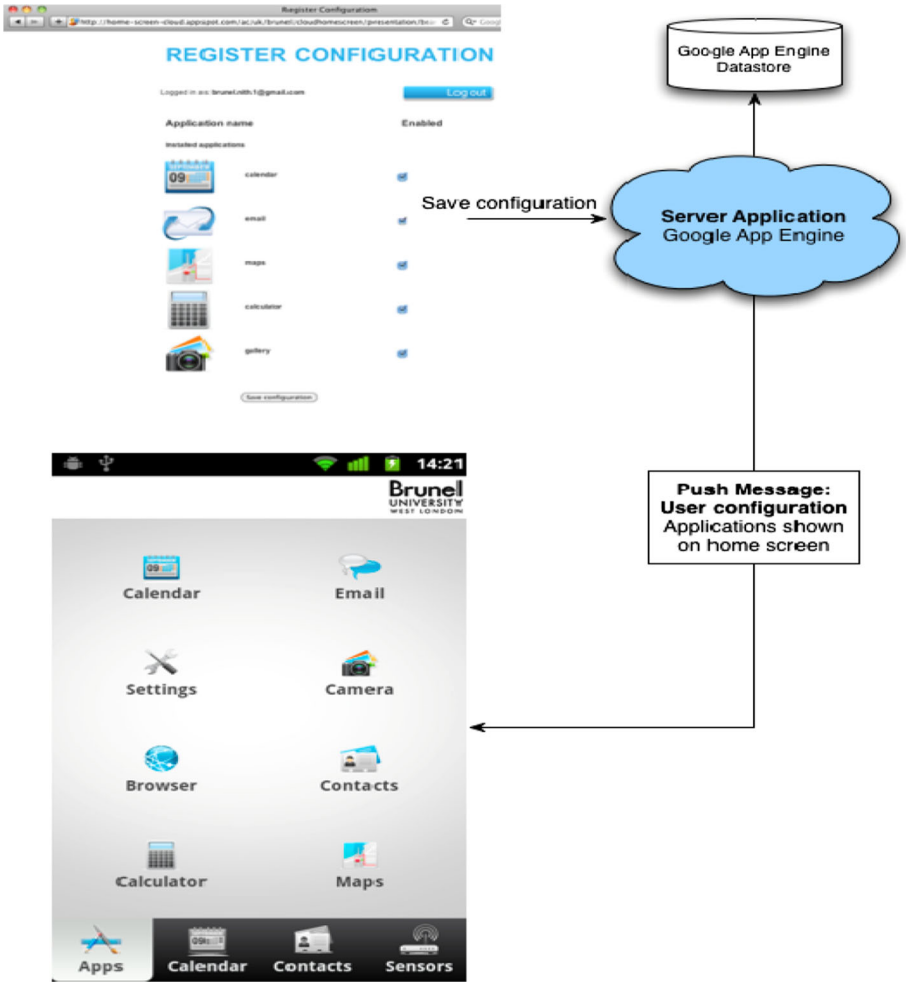
The calendar and contacts integration was also an important part of the Android application. We decided to

Fig. 3 Excerpt from the device messaging process

```
@Override
public void onRegistrered(Context context, String registrationId) throws IOException {
    callbackHandler.deviceRegistered(registrationId);

    if (registrationServerUri == null || "".equals(registrationServerUri)) {
        new Thread(new RegistrationIdSender(registrationId, Sc2dmDeviceRegistration.email)).start();
    } else {
        new Thread(new RegistrationIdSender(registrationId, Sc2dmDeviceRegistration.email, registrat
    }
}
```

Fig. 4 C2DM message cycle



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 contacts API and then using XML parsers to extract the content.

3.3 Using sensors as adaptation triggers

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 et al. [15] 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 that there is still work to do, like improving integration.

This is what we addressed in our work, where sensor data from the two mobile devices employed in our study were integrated with cloud-based services and used as input to the proof-of-concept application. Accordingly, 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 (the HTC Nexus One, for example, had 5 sensors available: accelerometer, magnetic field, orientation, proximity and light.). We ended up using two features directly in the prototype, namely the accelerometer and the light sensor. The accelerometer was used to register whether the device was shaking. If the device is shaking, it probably means that the user is 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 (Fig. 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 the colour when the lighting values measured from the environment changed.

4 Evaluation results

The developed prototype was evaluated in two phases. In the first, a pilot test was performed with a total of 12 users. Secondly, in the main evaluation, another 40 people participated in evaluating the application. All participants were classified according to a computer experience classification and answered a questionnaire after performing the instructed tasks with the application.

4.1 Participants

The first pilot test was performed with a total of 12 users. These users were of mixed age, gender and computer expertise. The results from this phase were fed back into the development loop, as well as helped remove some unclear questions in the questionnaire. In the second phase, the main evaluation, another 40 people participated. Out of the 40 participants in the main evaluation, two did not complete the questionnaire afterwards and were therefore removed making the total number of participants 38 in the main evaluation. All 12 from the pilot test and the 38 from the main test session were aged between 20 and 55 years old. All participants had previous knowledge of mobile phones and mobile communication, but had not previously used the type of application employed in our experiment. None of the pilot test users participated in the main evaluation. From the user computer experience classification (asserted based on a questionnaire employing the taxonomy of McMurtrey [12]), we learnt that the majority of the users had a good level of computer expertise.

4.2 Materials

Our prototype was evaluated on two mobile devices, the HTC Nexus One and the HTC Evo. The HTC-manufactured Nexus One represents one of the first worldwide available commercial Android phones. The Nexus One features dynamic voice suppression and has a 3.7-inch AMOLED touch sensitive display supporting 16 M colours with a WVGA screen resolution of 800 × 480 pixels. It runs the Google Android operating system on a Qualcomm 1-GHz Snapdragon processor and features 512-MB standard memory, 512-MB internal flash ROM and 4-GB internal storage.

The HTC Evo 4G is an Android phone shipped by the Sprint operator for the American CDMA network. The HTC Evo 4G features a 4.3-inch TFT capacitive touchscreen display supporting 64 K colours with a screen resolution of 480 × 800 pixels. It runs the Google Android operating system on a Qualcomm 1-GHz Scorpion processor and features WI-FI 802.11 b/g, 512-MB standard memory, 1-GB internal flash ROM and 8-GB internal storage.

4.3 Results

The results presented here illustrate different parts of the questionnaire: statements 1 to 3 target the *user interface*, statements 4 to 6 regard *sensor integration*, statements 7 to 9 focus on *the web application*, statements 10 to 13 centre on *context awareness*, whilst statements 14 to 17 are about *cloud computing*. The questionnaire ends with *overall usefulness*, which is in an open-ended question for

Fig. 5 Background light adjustment

Table 1 User evaluation questionnaire and results

	Statement	Mean	SD	*t* test
Statements regarding user interface				
Statement 1	It is easy to see the available functions	3.50	0.51	0.000
Statement 2	The features of the application are hard to use	1.89	0.73	0.378
Statement 3	The adaptability of the application is a feature I approve of	3.45	0.55	0.000
Statements regarding sensor integration				
Statement 4	The background colour in the application changes when the lighting in the room changes	3.55	0.65	0.000
Statement 5	When moving around, a simplified user interface is not presented	2.11	1.06	0.544
Statement 6	I found sensor integration annoying and would disable it on my device	1.84	0.72	0.183
Statements regarding the web application				
Statement 7	I was able to register my device application configuration in the web application	3.61	0.59	0.000
Statement 8	I was not able to store and push my configuration to my mobile device from the web page	1.47	0.80	0.000
Statement 9	I would like to configure my phone from a cloud service on a daily basis	3.18	0.69	0.000
Statements regarding context awareness				
Statement 10	The close integration with Google services is an inconvenience	1.76	0.88	0.107
Statement 11	Calendar appointments displayed matched my current user context	3.58	0.55	0.000
Statement 12	The contacts displayed did not match my current user context	1.29	0.52	0.000
Statement 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 and Google+)	3.29	0.73	0.000
Statements regarding cloud computing				
Statement 14	I do not mind cloud server downtime	2.08	0.78	0.539
Statement 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	2.16	0.79	0.225
Statement 16	Storing data in the Google Cloud and combining this with personal information on the device is a useful feature	3.26	0.60	0.000
Statement 17	I find the cloud-to-device application useful	3.53	0.51	0.000
Open comment question				

comments. The statements are given below (Table 1) together with the mean, standard deviation and the results of applying a one-sample *t* test.

4.4 User interface

Statements 1 to 3 deal with the user interface (Fig. 6). These results reveal positive facts about the interface, highlighting that the majority of the users found it easy to see all available functions; moreover, the vast majority (37/38) also finds the adaptability of the application a feature that they approve of. However, looking at statement 2, their opinions are split in terms of whether the features are hard to use, and the results in this respect are not statistically significant. Overall, in this category, the results indicate that it is easy to get an overview of the application and the test candidates find adaptability a positive feature.

4.5 Sensor integration

Opinions are split regarding sensor integration. Users agree that the light sensor is working as expected, but disagree

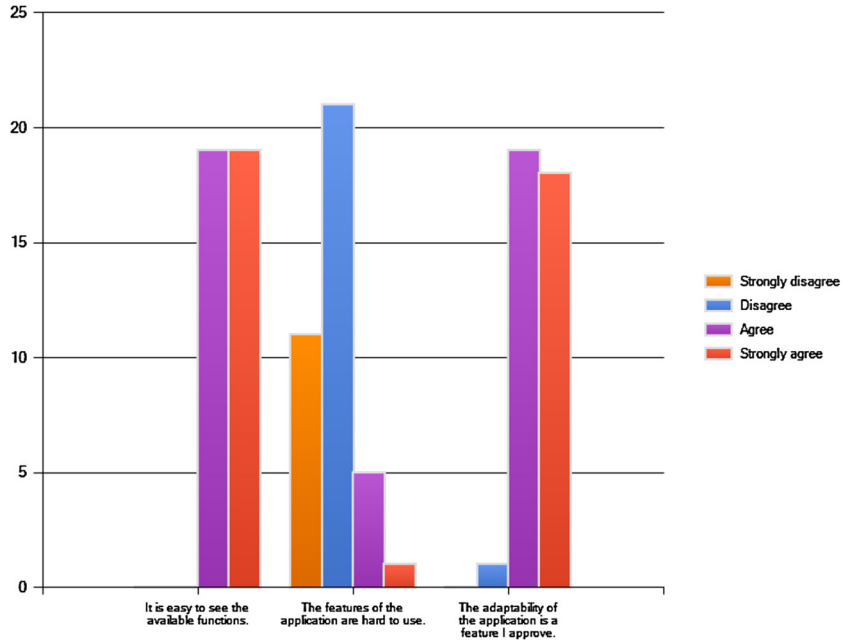


Fig. 6 Statements regarding user interface

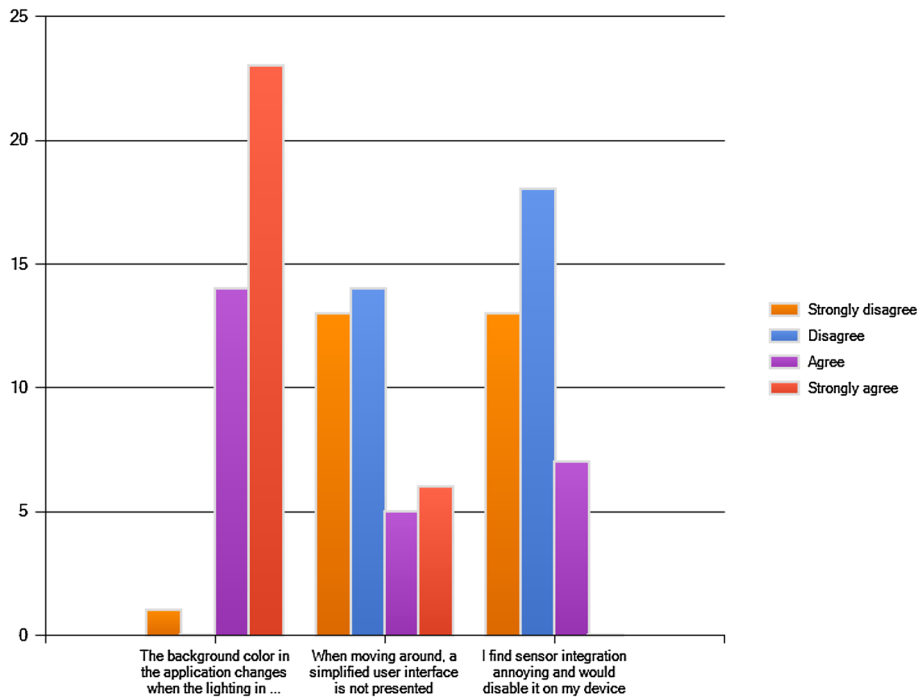


Fig. 7 Statements regarding sensor integration

whether the simpler user interface changes (Fig. 7). This can be due to the sensitivity threshold programmed for the sensor, and should be verified by more comprehensive testing. The majority, 32 out of 38, would not deactivate sensor integration and this is a useful observation, highlighting that sensors should be further pursued as context-aware input.

4.6 Web application

The statements dealing with the web application at the Google App Engine (Fig. 8) show that the web application performed as expected, by letting participants register their devices as well as pushing performed configurations to the devices. Moreover, answers from statement 9 are also quite

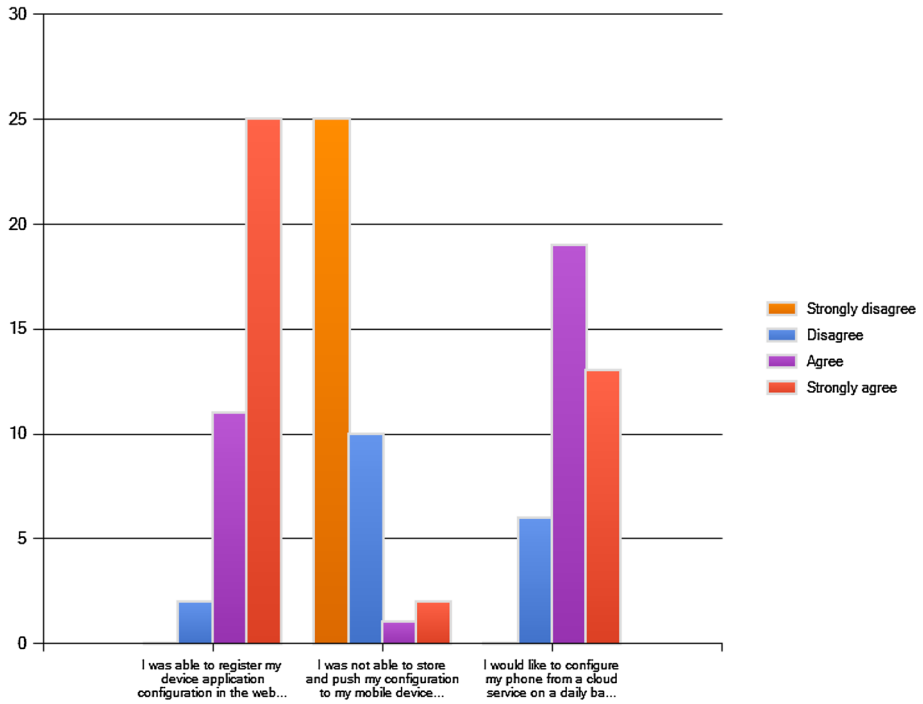


Fig. 8 Statements regarding web application

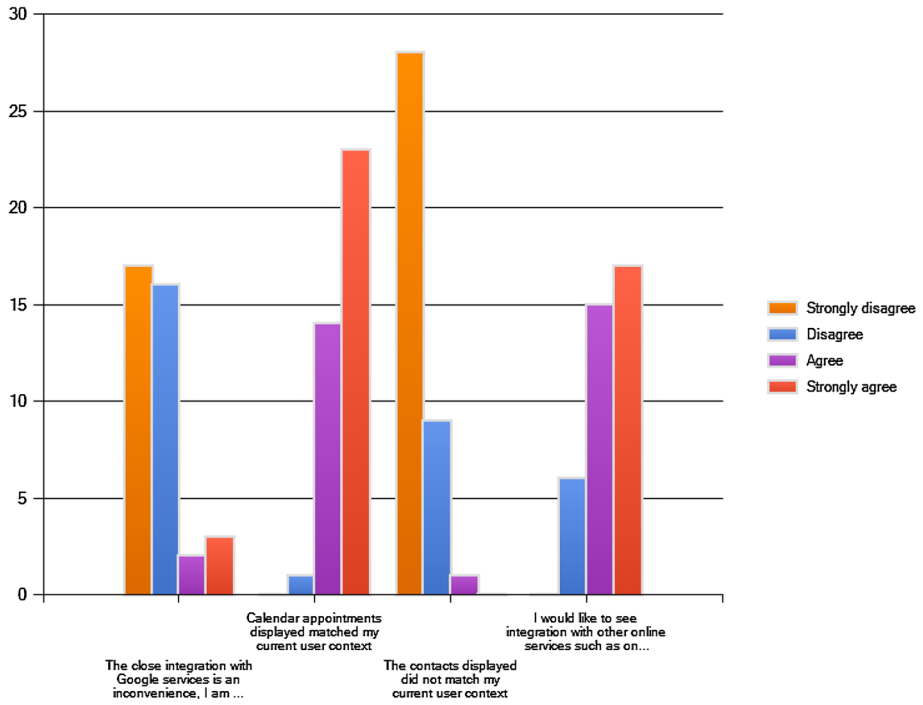


Fig. 9 Statement regarding context awareness

interesting ("*I would like to configure my phone from a cloud service on a daily basis*"), highlighting a positive attitude towards cloud-based services (32 out of 38 are positive).

4.7 Context awareness

In terms of context-aware information, the participants were asked to take a stand in respect of four statements, with results shown below (Fig. 9). For the first statement in this category ("*The close integration with Google services is an inconvenience*"), although a clear majority supported this assertion (33/38), opinions are somewhat spread and this answer is not statistically significant. For the next two questions, a very positive bias is shown, indicating correctly computed context awareness and correct presentation to the users. Again for statement 13 ("*I would like to*

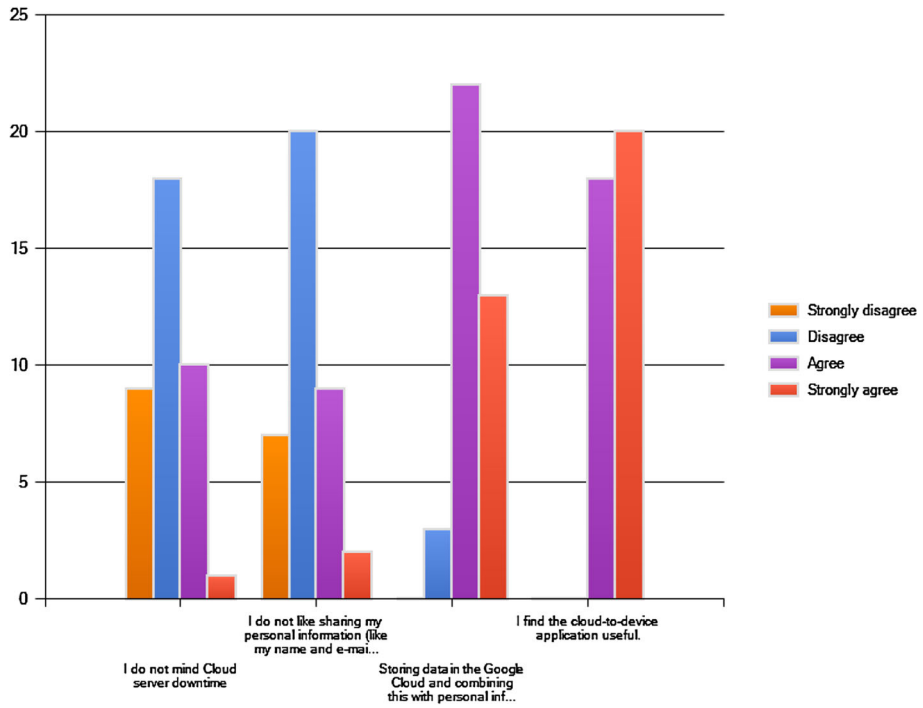


Fig. 10 Statement regarding cloud computing

see integration with other online services…"), users indicated their eagerness to see more cloud-based services and integration.

4.8 Cloud computing

When inspecting results from the cloud-computing section, results are mixed and differences in opinions do occur. For statements 14 ("*I do not mind Cloud server downtime*") and 15 ("*I do not like sharing my personal information … to a service that stores the information in the cloud*"), the results are not statistically significant, but they indicate a mixed attitude towards cloud vulnerability and cloud data storage. In respect of the two statements in this category with statistically significant results (statements 16 and 17), participants find storage of data in the cloud and using this as part of the data foundation for the application a useful feature and are positive towards it. Their answers also suggest a fondness for push-based application configuration (Fig. 10).

5 Related work and discussion

From the literature, we point at the ability for modern applications to adapt to their environment as a central feature [7]. Edwards [8] argued that such tailoring of data and sharing of contextual information would improve user interaction and eliminate manual tasks. Results from the user evaluation support this. The users find it both attractive as well as have positive attitudes towards automation of tasks such as push updates of information by tailoring the interface. This work has further elaborated on context-aware integration and shown how it is possible to arrange the interplay between on device context-aware information such as that provided by smartphone sensors, and cloud-based context-aware information such as calendar data, contacts and applications. In doing so, we build upon suggestions for further research on adaptive cloud behaviour as identified by Christensen [6] and Mei et al. [13].

In early work, Barkhuus and Dey [3] conducted a study to examine the effects of context on user's control over mobile applications. The study defined three level of interactivity between users and mobile devices including personalization, passive context awareness and active context awareness. User preferences were then studied according to these three levels. The study showed that in the case of passive and active context-awareness scenarios, users felt less in control of their mobile applications. But the overall conclusion was that users were willing to compromise on losing some control if they could get some useful reward in return. Our results show that things have not changed in this respect recently—users who evaluated the developed prototype appreciated the adaptation features that cloud-based data push enables.

Wei and Chan [17] incorporated a decade of work on context awareness and investigated the matter further. They presented three characteristics of context-aware applications:

- Context is application specific
- Context is external to applications
- Context can be used to change behaviour, data or structures

These characteristics are suggested to be adopted in future research. This is indeed what we have done; we have used application-specific context information (sensor data, calendar and contacts data), together with external context information stored in the cloud in order to change application structure, behaviour and interface. Wei and Chan [17] also make the point that the more fundamental the adaptation is (e.g. changing structures), and the later it occurs (e.g. at run-time), the harder it would be to be implemented. However, our work has taken up this challenge and shown how run-time structures and application adaptation can be achieved by using modern cloud architecture, all showcased through an implemented proof-of-concept prototype.

Satyanarayanan [16] exemplified context-aware attributes as: physical factors (location, body heat and heart rate), personal records and behavioural patterns. He stated that the real issue was how to exploit this information and how to deal with all the different representations of context. Whilst we have pursued, in our work, the ideas of different representations of context, further research is needed to further integrate other dimensions of context (e.g. physical factors, behavioural patterns).

To register the user tags, the standard Google Calendar and Contacts web interface were used. Such a tight integration with the Google services and exposure of private information was not regarded as a negative issue. As shown in the evaluation results of our developed prototype, most of the users surveyed disagreed that this was an inconvenience. This perception makes room for further integration with Google services in future research, where, amongst them, the Google + platform will be particularly interesting as this may bring opportunities for integrating the social aspect and possibly merge context awareness with social networks.

Sensors are an important source of information input in any real-world context and several previous research contributions look into this topic. The work presented in this paper follows in the footsteps of research such as that of Parviainen et al. [15], and extends sensor integration to a new level. By taking advantage of the rich hardware available on modern smartphones, the developed application is able to have tighter and more comprehensively integrated sensors in the solution. We have shown that it is feasible to implement sensors and extend their context-aware influence by having them cooperate with cloud-based services. However, from the user evaluation, one learns that although sensor integration as a source for context awareness is well received, there is still research to do. In particular, this should establish thresholds for sensor activation and deactivation.

6 Conclusions

This paper proposes the novel idea of using cloud-based software architecture to enable remote, context-aware adaptation. This, we argue, creates a new user experience and a new way to invoke control over a user's smartphone. Through a developed proof-of-concept application, we have shown the feasibility of such an approach; moreover, this has been reinforced by a generally positive user evaluation. Future research should continue to innovate and expand the notion of context awareness, enabling further automatic application adaptation and behaviour altering in accordance with implicit user needs.

References

1. Baldauf M, Dustdar S, Rosenberg F (2007) A survey on context-aware systems. Int J Ad Hoc Ubiquit Comput 2(4):263–277
2. Baltrunas L, Ludwig B, Peer S, Ricci F (2012) Context relevance assessment and exploitation in mobile recommender systems. Pers Ubiquit Comput 16(5):507–526
3. Barkhuus M, Dey AK (2003) Is context-aware computing taking control away from the user? Three levels of interactivity examined. In: Proceedings of the 5th international conference on ubiquitous computing (UbiComp), Seattle, WA, 12–15 Oct 2003, LNCS 2864 Springer, ISBN:3-540-20301-X, 149-156
4. Bellavista P, Corradi A, Fanelli M, Foschini L (2012) A survey of context data distribution for mobile ubiquitous systems. ACM Comput Surv 44(4):24–45
5. Binnig C, Kossmann D, Kraska T, Loesing S (2009) How is the weather tomorrow?: towards a benchmark for the cloud. In: Proceedings of the second international workshop on testing database systems, ACM, Providence, Rhode Island
6. Christensen JH (2009) Using RESTful web-services and cloud computing to create next generation mobile applications. In: Proceedings of the 24th ACM SIGPLAN conference companion on object oriented programming systems languages and applications, ACM, Orlando, FL
7. Dey A, Abowd GD (1999) Towards a better understanding of context and context-awareness. In: 1st international symposium on handheld and ubiquitous computing
8. Edwards WK (2005) Putting computing in context: an infrastructure to support extensible context-enhanced collaborative applications. ACM Trans Comput Hum Interact 12:446–474
9. Google (2013) What is Google App Engine? [Online]. Available: http://code.google.com/appengine/docs/whatisgoogleappengine.html
10. Kapitsaki GM, Prezerakos GN, Tselikas ND, Venieris IS (2009) Context-aware service engineering: a survey. J Syst Softw 82(8):1285–1297
11. Malandrino D, Mazzoni F, Riboni D, Bettini C, Colajanni M, Scarano V (2010) MIMOSA: context-aware adaptation for ubiquitous web access. Pers Ubiquit Comput 14(4):301–320

12. Mcmurtrey K (2001) Defining the Out-of-the-Box experience: a case study. In: Annual conference society for technical communication
13. Mei L, Chan WK, Tse TH (2008) A tale of clouds: paradigm comparisons and some thoughts on research issues. In: Proceedings of the 2008 IEEE Asia-Pacific services computing conference (APSCC 2008), IEEE Computer Society Press, Los Alamitos, pp 464–469
14. Mell P, Grance T (2011) The NIST definition of cloud computing. National Institute of Standards and Technology, Special Publication 800-145
15. Parviainen M, Pirinen T, Pertilä P (2006) A speaker localization system for lecture room environment. In: Machine learning for multimodal interaction, Springer, Berlin, pp 225–235
16. Satyanarayanan M (2011) Mobile computing: the next decade. SIGMOBILE Mob Comput Commun Rev 15:2–10
17. Wei E, Chan A (2007) Towards context-awareness in ubiquitous computing. In: International conference on embedded and ubiquitous computing (EUC 2007), Taipei, Taiwan, vol 4808, Dec 2007, LNCS Springer, Berlin, pp 706–717
18. Younas M, Awan I (2013) Mobility management scheme for context-aware transactions in pervasive and mobile cyberspace. IEEE Trans Ind Electron 60(3):1108–1115
19. Zhou J, Gilman E, Palola J, Riekki J, Ylianttila M, Sun J (2011) Context-aware pervasive service composition and its implementation. Pers Ubiquit Comput 15(3):291–303