Mobile Context Toolbox
An Extensible Context Framework for the Maemo Platform

Arkadiusz Stopczynski, Jakob Eg Larsen, Lukasz Skomial
Technical University of Denmark
DTU Informatics, Section for Cognitive Systems
Richard Petersens Plads, Bld. 321, DK-2800 Kgs-Lyngby, Denmark
Email: s081608@imm.dtu.dk, jel@imm.dtu.dk, s073383@student.dtu.dk

Abstract

In this paper we describe an open framework utilizing sensors and application data on the Maemo mobile platform enabling rapid prototyping of context-aware mobile applications. The framework has an extensible layered architecture allowing new hardware and software sensors and features to be added to the context framework. We present initial results from in-the-wild experiments where contextual data was acquired using the tool. In the experiments 6 participants were using a Nokia N900 mobile phone continuously with a logger application for an average of 33 days. The study has provided valuable insights into human behavior in terms of places visited, people met, etc. Moreover, it has provided useful insights into platform issues of the system deployed in real-life usage situations, including the stability and power consumption.

Index Terms: context, context-awareness, mobile, framework, toolbox, application prototyping, Maemo

I. Introduction

With the recent increase in their popularity and features offered mobile phones have become ubiquitous devices, following their users during everyday activities. Due to embedded sensors becoming an integral part of modern smart phones, context-awareness has gotten increased attention. Standard off-the-shelf mobile phones have several embedded sensors, such as GPS, accelerometer, light sensor, proximity sensor, microphone, camera, as well as multiple network connectivity options, such as GSM, WLAN, and Bluetooth. However, utilizing these sensor inputs for developing context-aware mobile applications is a complex task. The complexity of development include accessing all of the available sensors in a simple, unified way and translation of low-level raw data into a meaningful description.

A variety of different applications utilize sensors and context-awareness, including novel games [1], tourist guides [2] and augmented reality applications [3]. Building such rich applications is however difficult, as it usually requires direct interfacing with various sensors and APIs. This raises the need for software tools and frameworks simplifying access to the data and shifting the attention from how to get to how to use. Not only access to raw sensors data must be easily obtained, it is also necessary to map it out to a contextual information capturing human behaviour. GPS coordinates are usually not suitable for direct context-oriented usage, but must be first translated into human readable labels or visual representations. Both access to the raw data and translation to high-order information has to be done in a similar way in

many applications. We propose a flexible context framework with a set of components that can infer such information and make it available across various applications on the platform.

The framework has a layered architecture, which abstracts the complexity of multiple low-level sensors from the application programmer. Sensors is interpreted broad, as it includes embedded sensors such as accelerometer, microphone, camera, etc, as well as networking components, phone application data (calendar, address book, phone log, etc), and phone state (profile, charge level, etc). Our current implementation exploits the potential of using such sensors in modern off-the-shelf Maemo-based mobile phones for novel applications. The focus has been on the assessment of the potential for such a framework taking into consideration the increased performance requirements in such systems due to the additional CPU, memory and power consumption introduced.

The framework presented in this paper is suitable for usage not only in the context-aware end-user applications but also in the experiments involving context-logging. Such experiments carried over extended periods of time (days or weeks) can provide a valuable insight into human behavior. Gathered data from such experiment, with context-logger build using Mobile Context Toolbox is presented in this paper. We report from initial deployment of the system that included continuous use by 6 participants, who were provided with a standard mobile phone (Nokia N900) with Mobile Context Toolbox installed along with an application that would continuously log the data acquired from all sensors currently supported by our framework. The experimental results from real-life use of the device for a period 33 days on average have provided interesting insights into the potential of such system for obtaining behavioral data.

II. Related Work

The idea of a context framework for mobile devices has so far been implemented in several projects on a variety of mobile platforms [4][5][6][7][8]. Previously we have created and experimented with the Mobile Context Toolbox for the S60 platform (Python-based), as described in [9]. This implementation presents a layered approach to data gathering and processing and features a widget layer where data from various sensors is gathered and translated into meaningful labels. A related project on the Maemo platform is ContextKit, a framework for collecting contextual information from the bowels of the system, cleaning them up and offering them through a simple API.[1] This framework is multi-platform (including Maemo/MeeGo) and provides a unified access to many embedded sensors. It is however complex to deploy and use, not suitable for rapid prototyping, providing only bottom layer of the architecture, namely access to the raw/filtered data.

Roto and Olasvirta [10] have studied mobile users on the move while they were using web-browsers on mobile phones, by employing multiple cameras worn by the test participants. A shorter attention span when using applications on the move compared to use in laboratory settings were reported [11]. However, as such methods are resource demanding several have experimented with the use of automatically logging embedded sensor data in order to study human behaviour [12] or predictability [13][14][15] or to obtain insights into the use of mobile applications in context. One example of the latter is Froehlich et al. [16] having created a mobile tool capable of reading data from multiple embedded sensor similar to our approach, including device logging of application use and context using embedded sensors. However,

[1] http://maemo.gitorious.org/maemo-af/contextkit/blobs/master/README

their aim was to combine quantitative and qualitative methods for in-the-wild collection of data about usage. Recently, we have used the approach to study mobile use in context of the media player application for music playback [17]. Boehm et al. [18] describe the mobile application IYOUIT that collects contextual data, but aims to appeal to the end-user of the application with a mobile user interface that allow them to actively see the contextual data, as well as to share it in a social network. CenceMe also emphasize the social aspects [19].

III. Mobile Context Toolbox

The Mobile Context Toolbox is build for the Maemo platform running on Nokia N900 smart-phones. The reason for choosing this platform is the wide array of embedded sensors as well as relatively easy and unrestricted access to them. As such N900 constitutes an interesting research platform in the context-awareness and cognitive domain mentioned above.

A. Mobile Phone Sensors as Context Sources

In principle, mobile devices can acquire context data through a large variety of sensors embedded in the device and in the surrounding environment, as well as from online sources.

- **GPS** provides an absolute localization system which was designed for surveying tools and navigation aids with global coverage.
- **GSM/UMTS** provides country and area code as well as cell ID, uniquely identifying base station.
- **Bluetooth** can be used for scanning for neighbouring devices, providing information about location and people around.
- **WLAN** can be used for scanning for neighbouring networks, providing information about location and movement.
- **Correspondence logs** are used to monitor users activity.
- **Audio, pictures, and video** can be used for discovering context of the user, including such abstract concepts as mood or emotions.
- **System state (profile, battery, network)** is useful for enriching the context and smart approach to phone behavior.
- **Explicit user entry (application data and prompts for tags)** is possibly a very useful source of information, however very intrusive.

Context-awareness originated from ubiquitous computing and has been introduced by Schilit in 1994 [20]. When the term context is considered, there are several aspects which have to be taken into account, including:

- **Location** – information concerning the current users position or his perceived path of interest.
- **Time** – interpretation of current time or time interval in relation to users current activities.
- **Activity** – what is being done by the user.
- **People** – determining presence of other people in the neighborhood.
- **Reason** – reasons of current users behavior.

Figure 1 depicts the layered architecture of the Mobile Context Toolbox for the Maemo platform. At the bottom we have the operating system that delivers certain services, mostly through the D-Bus system. On top of that, is the layer of Adapters, providing programmatic

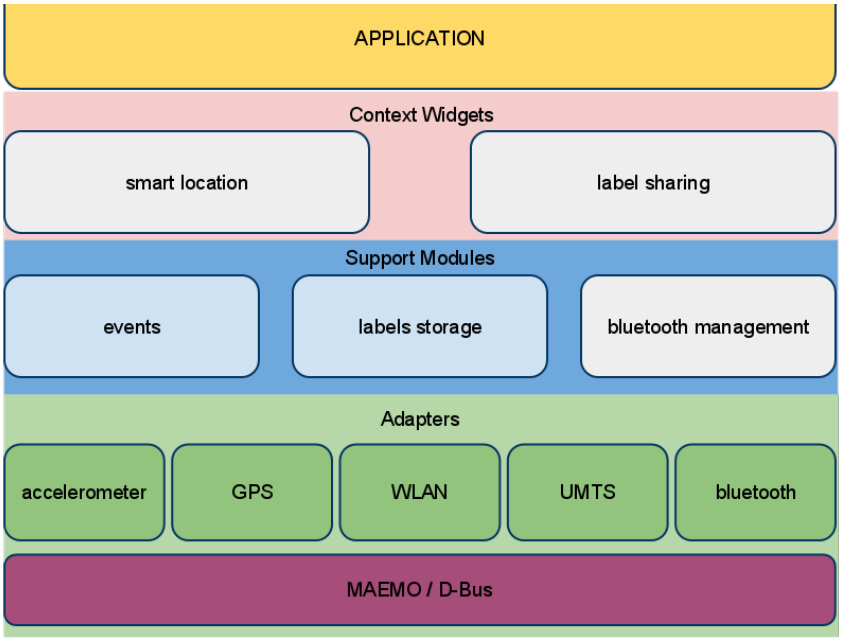

Figure 1. Overview of the Mobile Context Toolbox system architecture and components. Raw sensor data originates at the bottom and is collected in the Support and Widgets layers to provide more descriptive information to the Application

access to single sensors in a unified way. Information from single sensors can be managed and joined in the layer of Support Modules. Those modules deliver extra functionalities for the toolbox, mainly simplifying common tasks. Context Widgets is the layer using both Support Modules and single Adapters to create a more complex context. Applications can use all the below layers in a unified way, so they are provided with different levels of the information granularity (from raw data to complex contextual states).

B. General architecture

The layered structure allows easy adding, replacing or removing of modules. The general assumption is that data flows only upwards, which makes the modules dependencies very simple: it is possible to deploy only the necessary adapters as there are no horizontal dependencies. The modules communicate using unified and well defined interfaces, which makes it easy to introduce new adapters, support modules or widgets as they are developed. It is an important property, as similar frameworks tend to break quickly once more components are introduced.

The implementation is based on system-level multi-threading (POSIX threads) which makes it framework agnostic. Written in C++ it can be used with GTK+, Qt, wrapped for Python or in any other C++ based framework. Using pthreads for multithreading is sub-optimal in most of the applications, where the reading code could be smartly placed in any of the loops (e.g. using GTK+ main loop). This is however the cost of a generic approach, where our libraries do not depend on Qt/GTK+ or any other framework.

C. Sensor adapters

The first layer in the architecture consists of simple wrappers over single sensors. There is currently no global system for sharing the results of readings, meaning that if two separate applications use for example WLAN scanning, it will be performed twice.

1) Bluetooth module: The libmctbluetooth library is responsible for retrieving a list of bluetooth devices present in the devices range. The discovery process starts by getting adapters power state, recording it, and powering on if necessary. The loop begins when a request for a discovery session is sent through the DBus system and then signals from the adapter are being caught.

2) Accelerometer module: This library provides methods for accessing accelerometer data. There are three main modes of the accelerometer operation: real-time, single and burst readings. The first mode is useful for applications that use the accelerometer for the real time controls. Single readings can be performed in a scheduled way, in a loop with a large sleep interval, or without using a loop. The burst mode is suitable for many context-aware applications, as it can be used to identify various states of the device (e.g. detecting states such as walking or running).

3) Location module: This library is responsible for providing information about the current location. It is a simple wrapper over liblocation module, already available in the Maemo platform. It preserves all the possibilities of liblocation, simply hiding the complexity of setting up a connection, adding callbacks etc. Our wrapper provides access to all the information available in liblocation, i.e.: latitude, longitude, timestamp, altitude, speed, track, climb, and accuracy of those fields.

4) Cell ID module: This library provides methods for reading GSM/UMTS cell ID. There are four main fields available on the Phone.Net.get registration status interface: Mobile Country Code, Mobile Network Code, Location Area Code, and cell ID. The combination of those four values can be used to uniquely identify location of the base station/device.

5) WLAN module: The primary function of this library is to provide methods for scanning for available WLAN networks. The properties of the network include: essid, bssid, type, encryption, channel, quality. Scanning can be performed in a loop in a non-blocking way or on-demand, in a blocking way.

D. Support modules

1) Events module: The function of this library is to provide methods for events scheduling. It is a common situation in context-aware applications, that it is required for one event to act as a trigger for another, for example discovering a new cell ID may trigger GPS to get more accurate location. This support library provides method for setting up a trigger-event pair. Checking of the trigger value is done in a loop, in a separate thread for every event. The Event class represents such a pair, together with information about how often the trigger should be checked and how many times this should be done (an infinite loop or a limited number of checks).

2) Label storage module: This module is currently in implementation. It will eliminate the necessity to set up and manage contextual data storage. Since the module will provide a uniform way of storing contextual information, it will also be possible to share the data among different applications on the device.

The layered and modular architecture of the module is depicted in Figure 2 having three levels with different functional blocks. The top layer represents the interface which is exposed by the module to the user. The second layer is responsible for implementation of internal functionalities and data structures. The functionalities implemented here are responsible for mapping the data types given by the public interface to internal data structure representation. In order to provide flexibility the internal data structure is represented by several embedded containers which keep different sensors data, as shown in Figure 3.

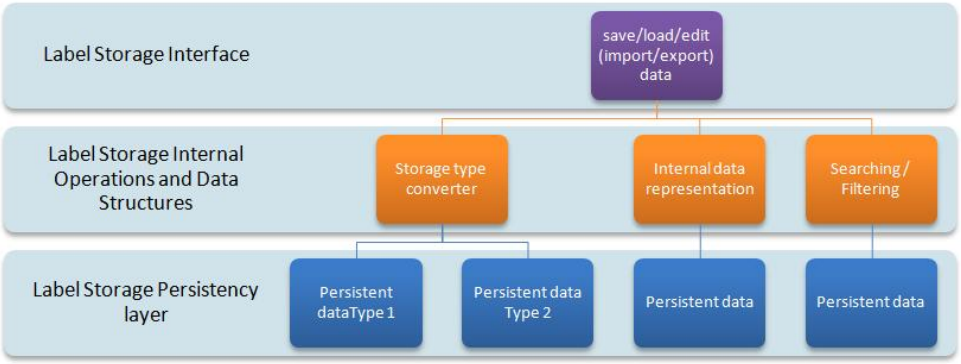

Figure 2. Proposed architecture of the Label Storage module

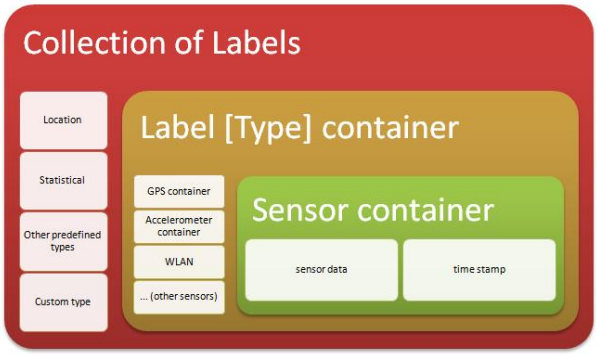

Figure 3. Internal data structure of the Label Storage module

E. Context Widgets

The Context Widgets layer contains modules collecting data from several adapters and translating them into meaningful description. They can also provide extra functionalities to be used by applications as building blocks. Examples of such widgets are Smart Location and Labels Sharing. Smart Location is designed to use data from accelerometers, GPS, WLAN and cellular network to provide information about location with best accuracy and power efficiency. The mode of operation can be seen as a hierarchical sensor reading: movement detected by accelerometer indicates that the position is changing, which can be later tracked using WLAN or cellular network. If an application using this module requires better accuracy, GPS will be eventually turned on to provide location with requested precision. Label Sharing is a widget for a collaborative approach to context-awareness. Instead of trying to infer the whole context separately on each device, they can create ad-hoc networks (using Bluetooth) and share raw data and labels.

F. Context data logger application

The context logger application has been created for the purpose of collecting data from phone sensors over long time periods. It utilizes the MCT sensor adapters layer for accessing particular data from the phone wrapped using the Qt framework which has been used for threading, process management, accessing the file system and user interface development. A simplified structure of the application is presented in Figure 4.

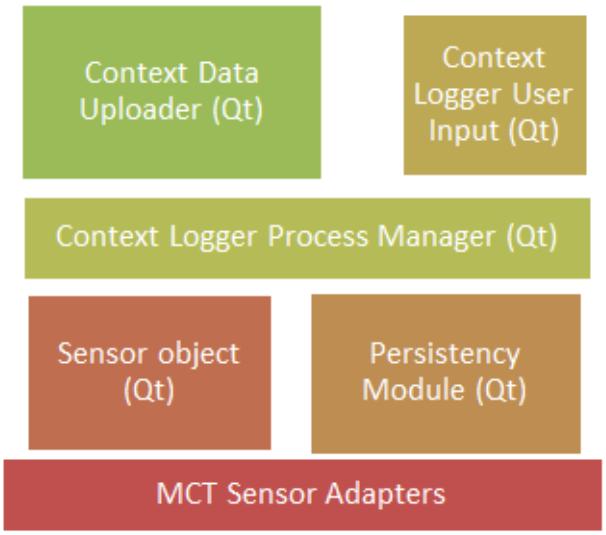

Figure 4. Simplified structure of the DataCollector solution. MCT is used for accessing data from sensors

The entire solution consists of three separate applications:

- **DataCollector** – reading out sensors data and storing it in files on the device.
- **ContextDataUploader** – Qt based application allowing users to submit results stored on the device to the server.
- **ContextUserInput** – a Qt based application allowing users to tag places, people and activities.

The DataCollector application has to run reliably on the devices for longer time periods as a background daemon, requiring no user interaction.

IV. Experiments and Results

We have carried out initial experiments with our mobile context platform involving N=6 test participants that carried a mobile phone (Nokia N900) for a duration of 33 days on average. The purpose of the experiment was to collect a variety of data about everyday activities of participants. Since the number of participants was limited by availability of devices it was decided to create two sub-groups among them. Each group consisted of participants knowing each other who were also meeting on a regular basis. Additionally one of the participants were known by both groups. The members of the first group selected were students living in the same dormitory. The second group consisted of three employees working in the same office. Both groups were asked to carry the devices with them as much as possible and preferably to use test devices as their primary phones. This was accomplished by four participants (having SIM cards in the phones, as shown in Table I). Apart from keeping the devices charged all of the participants were asked to use two additional applications. One application was used for uploading the collected data to the server (for back-up purposes) once a day. The second application was used for tagging the current location/activity/people around as often as they would find it necessary. It was left to the participants to decide which situations they wanted to tag.

The data collecting system was recording readouts from the available sensors all of the time when the phone was switched on. The sampling rates used were:

- Wireless Networks scanning – every 10 minutes
- GSM cell readout – every 5 minutes
- Accelerometer – every 1 minute with burst of 10 readouts/sec
- GPS readout – every 2 minutes
- Bluetooth scanning – every 10 minutes scanning for 30 seconds

The sampling rates were chosen empirically – in order to provide as much data as possible but also keep the battery life of the device at a reasonable level. The selected values provided eight to twelve hours of devices battery life, which was acceptable for this experiment.

The survey started on 11-07-2010 and ended on 22-08-2010. Table I presents an overview of the collected data from the 6 devices used.

TABLE I
OVERVIEW OF THE DATA ACQUIRED FROM THE 6 TEST PARTICIPANTS OVER A DURATION OF 5 WEEKS

User	Duration (days)	Acc. readouts	GSM cell readouts	Unique GSM cell readouts	Unique Wi-Fi networks	Wi-Fi networks	GPS readouts	Bluetooth devices	Unique bluetooth devices
1	41	170721	10374	178	15272	1177	27067	10820	772
2	41	1757863	11732	83	83595	1013	26191	17046	1122
3	22	101576	2129	75	31511	556	13075	5137	258
4	32	131023	0 (no SIM)	0	39295	1662	20093	9031	632
5	42	257092	0 (no SIM)	0	28995	127	22346	358	15
6	22	1062945	13304	92	55473	854	14279	10731	494
Total	200	3481220	37539	395	254141	4640	123051	53123	3151

Figure 5 is an example of analysis of collected data. It depicts an overview of the registered wireless networks registered by one selected device in the course of 40 days. Each readout is represented by a block of a color generated based on the networks BSSID thus the same network can be identified in different time sections. The blank spots in the diagram represent the point when the device has either been turned-off or no networks have been registered. It is clear from the figure that the uptime is over 95%, giving almost full overview of person's locations in 40 days. When coupled with user-generated tags, visible networks can be used to accurately pinpoint location of the user, as well as transitions between those.
Such location data from several participants can be also used to generate clusters of networks seen by participants, revealing how often they have been meeting. This reveals social relations of users and can be coupled with data collected from additional sensors, notably from bluetooth scanning.

V. DISCUSSION

Mobile phones carried by users throughout the day enable a unique opportunity to capture life data from a wide range of sensors through long periods of time. Together these sensors provide an interesting source of information about activities, people, places and other entities. As such, the mobile phone serves as a proxy in terms of providing information about the context of the human user. With mobile phones being ubiquitous devices, context frameworks like ours have interesting potential within several application areas. Continuous logging of

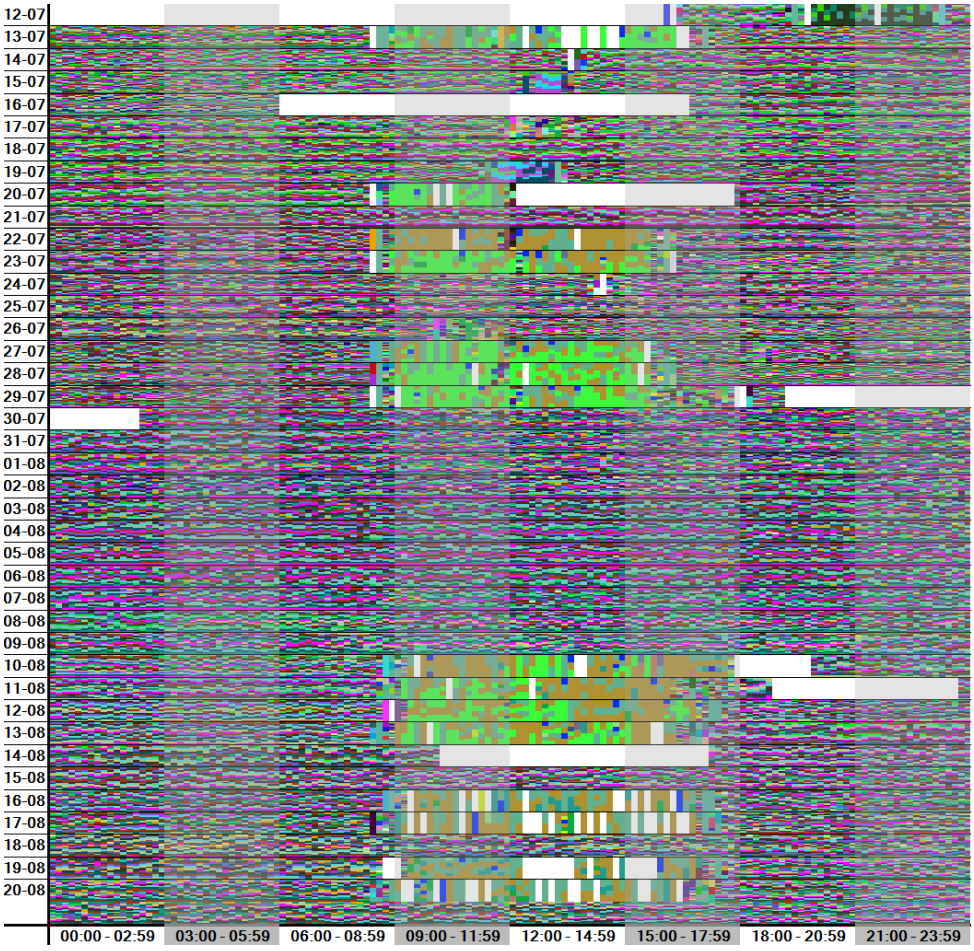

Figure 5. Wireless networks readouts for a selected participant. The white areas denote periods where the phone was switched off or no WiFi access points were available. Each color denotes one WiFi network

sensor data allow detailed information about our lives (private, work and social context) to be discovered. This could be used for instance in health applications for self monitoring with the purpose of self-reflection. Another area is information retrieval and information management, where the logs could allow context-based retrieval of information.

The presented experiment demonstrates that the implemented framework simplifies development of context-aware applications and loggers. It also shows that the architecture is stable enough to run for several days with numerous sensors being read continuously. With the DataCollector process running, the devices could operate for about 10 hours without charging. For comparison, the same device without the logger running, lasted for 72 hours. Significant power drain was to be expected with all the sensors running constantly. It is however important to note that even with such heavy usage, the device could last through the typical days of the participants.

The presented data, although not yet analyzed in details, show the potential of obtaining interesting insights into participants everyday life. The domains covered include places, people and activities and form a basis for analysis of behavioral patterns.

VI. Future Work and Conclusions

We plan to continue the work on the Mobile Context Toolbox and to make it available as open source in order to allow others to utilize the framework for further field studies like the one presented here and to contribute to the further development of the MCT itself. Obvious extensions include additional adapters providing more phone information, status, application events and data, and access to additional embedded sensors. Additional sets of context widgets can be developed utilizing information aggregation to infer more precise context information based on multiple sensors. Thereby, the Mobile Context Toolbox can become an even more useful tool in terms of lowering the barrier to the development of and experiments with context-aware mobile applications. In the near future, we plan to adapt the toolbox to the MeeGo platform. Due to the layered design, this will require only changes and adjustments in the Adapters layer, as the system APIs providing raw data are changed. The other parts of the toolbox can be directly deployed on MeeGo.

We have demonstrated the Mobile Context Toolbox for the Maemo platform as a framework for prototyping context-aware mobile applications and carrying out context-oriented experiments. It is our finding that the Maemo platform is suitable for deploying such experiments. The operating system stacks are stable and can handle the requests and make data available in a reliable way. It is an advantage that applications can run with user privileges, not requiring certificates or root access, which means that applications built using our framework are fully native applications, which makes them easy to develop, install and manage. The presented framework offers not only simple adapters for single sensors, but also more advanced modules for gathering and translating data from multiple sources in the widgets layer, which distinguishes it from other context-based solutions available.

Our experiments gathering behavioral data through context-logging have demonstrated the stability and flexibility of the current prototype implementation. The results are yet to be analyzed in more details, but our initial analysis of the data collected has shown the potential of using smartphones as a research instrument for obtaining insights on human bahaviour.

Acknowledgment

The authors would like to thank participants of the experiment. Furthermore thanks to Forum Nokia and Nokia Denmark for Maemo devices used as part of this work.

References

[1] Cheok, A.D., Yang, X., Ying, Z.Z., Billinghurst, M. and Kato, H. Touch-Space: Mixed Reality Game Space Based on Ubiquitous, Tangible, and Social Computing, Personal and Ubiquitous Computing, Vol. 6 , No. 5-6, pp. 430–442, Springer-Verlag, London, UK, 2002.

[2] Cheverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C. Developing a context-aware electronic tourist guide: some issues and experiences. Proceedings of the SIGCHI conference on Human factors in computing systems (2000), Pages: 17–24

[3] Rekimoto, J., Ayatsuka, Y. and Hayashi, K. Augment-able reality: Situated communication through physical and digital spaces. 2009 International Symposium on Wearable Computers Rekimoto (1998) vol:1998 pg:68

[4] Salber, D., Dey, A.K., Abowd, G.D.: The context toolkit: aiding the development of context-enabled applications. In: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, pp. 434–441. ACM Press, New York (1999)

[5] Dey, A.K., Abowd, G.D., Salber, D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction 16(2), 97–166 (2001)

[6] Bardram, J.E.: The Java Context Awareness Framework (JCAF) a Service Infrastructure and Programming Framework for Context-Aware Applications. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 98–115. Springer, Heidelberg (2005)
[7] Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: ContextPhone: A prototyping platform for context-aware mobile applications. IEEE Pervasive Computing 4(2), 51–59 (2005)
[8] Oulasvirta, A., Raento, M., Tiitta, S.: ContextContacts: re-designing SmartPhones contact book to support mobile awareness and collaboration. In: Proceedings of the 7th international conference on Human computer interaction with mobile devices & services, pp. 167–174. ACM, New York (2005)
[9] Larsen, J. E., & Jensen, K. Mobile Context Toolbox. An Extensible Context Framework for S60 Mobile Phones. In P. Barnaghi et al. (Eds.): EuroSSC 2009, LNCS 5741, pp. 193–206, (2009)
[10] Roto, V. & Oulasvirta, A. Need for non-visual feedback with long response times in mobile HCI. Special interest tracks and posters of the 14th Int. conf. on World Wide Web (2005) 775–781.
[11] Oulasvirta, A., Tamminen, S. and Roto, V. & Kuorelahti, J. Interaction in 4-second bursts: the fragmented nature of attentional resources in mobile HCI. Proc. of the SIGCHI conf. on Human factors in computing systems (2005) 928.
[12] Eagle, N., Pentland, A.: Reality mining: Sensing complex social systems. Personal and Ubiquitous Computing 10(4), 255–268 (2006)
[13] Song, C., Qu, Z, Blumm, N., & Barabsi, A. Limits of predictability in human mobility. Science, vol. 327, no. 5968, pp. 1018–1021 (2010)
[14] Jensen, B. S., Larsen, J., Hansen, L. K., Larsen, J. E. and Jensen, K. Predictability of Mobile Phone Associations. Int. Workshop on Mining Ubiquitous and Social Environments (2010)
[15] Jensen, B. S., Larsen, J. E., Jensen, K., Larsen, J. and Hansen, L. K. Estimating Human Predictability From Mobile Sensor Data. IEEE Int. MLSP Workshop (2010)
[16] Froehlich, J., Chen, M. Y., Consolvo, S. & Harrison, B. and Landay, J. A. MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones. Proc. 5th Int. conf. on Mobile systems, applications and services (2007) 57–70.
[17] Larsen, J. E., Petersen, M. K., Handler, R., & Zandi, N.. Observing the Context of Use of a Media Player on Mobile Phones using Embedded and Virtual Sensors. NordiCHI2010 workshop on Observing the Mobile User Experience, Reykjavik, Iceland (2010)
[18] Boehm, S., Koolwaaij, J., Luther, M., Souville, B., Wagner, M., Wibbels, M.: Introducing IYOUIT. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 804817. Springer, Heidelberg (2008)
[19] Miluzzo, E., Lane, N., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S., Zheng, X., Campbell, A.: Sensing meets mobile social networks: The design, implementation and evaluation of the CenceMe application. In: Proceedings of the 6th ACM conference on Embedded network sensor systems, pp. 337–350. ACM, New York (2008)
[20] Schilit, B., Adams, N. and Want, R. Context-aware computing applications. IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94) (1994), Santa Cruz, CA, US. pp. 89–101.