A Framework for Mobile User Activity Logging

Wolfgang Woerndl, Alexander Manhardt, Vivian Prinz

TU Muenchen, Chair for Applied Informatics / Cooperative Systems (AICOS)
Boltzmannstr. 3, 85748 Garching, Germany
{woerndl, manhardt, prinzv}@in.tum.de

Abstract. The goal of this work is a unified approach for collecting data about user actions on mobile devices in an appropriate granularity for user modeling. To fulfill this goal, we have designed and implemented a framework for mobile user activity logging on Windows Mobile PDAs based on the MyExperience project. We have extended this system with hardware and software sensors to monitor phone calls, messaging, peripheral devices, media players, GPS sensors, networking, personal information management, web browsing, system behavior and applications usage. It is possible to detect when, at which location and how a user employs an application or accesses certain information, for example. To evaluate our framework, we applied it in several usage scenarios. We were able to validate that our framework is able to collect meaningful information about the user.

Keywords: user modeling, mobile, activity logging, personal digital assistant, sensors

1. Introduction

Mobile devices like Smartphones and personal digital assistants (PDAs) are becoming more and more powerful and are increasingly used for tasks such as searching and browsing Web pages, or managing personal information. However, mobile information access still suffers from limited resources regarding input capabilities, displays, network bandwidth etc. Therefore, it is desirable to tailor information access on mobile devices to data that has been collected and derived about the user (the user model).

When adapting information access, systems often apply a general user modeling process [1]. Thereby, we can identify three main steps (Fig. 1): 1. collecting data about the user, 2. analyzing the data to build a user model, 3. using the user model to adapt information access.

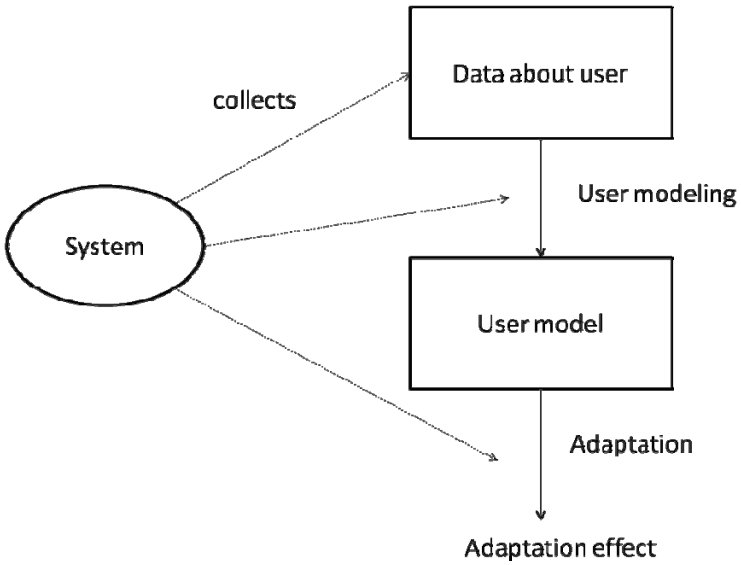

Fig. 1. User modeling process [1]

In this work, we focus on the first step of this user modeling process: the collection of data about the user in a mobile environment. The goal of this work is a unified approach for recording user actions on mobile devices in a granularity appropriate for user modeling. To fulfill this goal, we have designed and implemented a framework for mobile user activity logging on Windows Mobile PDAs. The framework handles different kinds of hardware and software sensors in a combined and consistent way.

The remainder of this paper is organized as follows. The next section describes requirements and related work. In Section 3 we explain the design and implementation of our framework for mobile user activity logging. Section 4 covers the evaluation of our approach. Finally, we give a summary and outlook for future work in Section 5.

2. Requirements and Related Work

2.1. Requirements

The most important feature of our mobile user activity logger is to cover all user actions that can occur on a mobile device with associated sensor data. Since the goal of this work is collecting data for a specific purpose (user modeling), it is important to consider the *granularity* of the data recording. To test the usability of a mobile software application, for example, it may be necessary to record movements on a

touch screen, single keystrokes or exactly where a user hits a button. This may lead to too much data that has to be handled and stored. On the other hand, if a system only records that a user has been starting the mobile web browser, for example, this information may not be sufficient to be able to derive knowledge about what the user is interested in. For our purpose of user modeling, it is useful to also collect which web sites the user has visited or which keywords she has entered for a web search, for instance.

For hardware sensors, an activity logging system shall record data when user actions lead to a change in the situation the user is in. For example, the system should log when a user is driving or walking around and thus changing her position. Alternatively, the system could record a snap shot of the sensor status at fixed time intervals. This may lead to a lot of redundant data and is not preferable since resources such as storage capacity are limited on the mobile device.

Another focal point to consider is *implicit* versus *explicit* user profile acquisition. Due to the limitation of the mobile user interface, necessary user interactions should be kept at a minimum. Users do not like to fill out forms or answer questions on a mobile device. In addition, the system should take into consideration the mobile-users' limited attention span while moving, changing locations and contexts, and expectations of quick and easy interactions [2]. Therefore, the data collection should be based on observing the user in her ongoing activities without distracting her too much. It is desirable to collect real usage data as it occurs in its natural setting [3]. Explicit user interaction could be optionally used to augment the implicitly collected data from hardware and software sensors. For example, the system could optionally ask whether a user is in a "work" or "leisure" setting in a particular location. By doing so, the user modeling system could later aggregate information from different "leisure" situations.

Finally, every system that collects data about the user has to consider users' *privacy* concerns. For mobile user modeling this is especially important since additional information such as the user position is available. Sensor data may be even more sensible than information users provide in a web form. Therefore, it is desirable to keep the collected data on the mobile device and not send it to a server over a network. By doing so, the user can always shut down the data recording or delete the data. She thus is able to retain control over the collected data. In addition, the user should have an option to manually disable individual sensors. This option is also beneficial to be able to save battery power. An example is to disable the GPS sensor when a user is inside a building for a whole day.

2.2. Related Work

In a nutshell, existing related work is either focused on gathering data in a non-mobile desktop setting, do collect data from specific sensors only (e.g. analysis of user location based on GPS logs) or were created for different purposes other than user modeling.

An example for activity logging in a desktop setting is the approach by Chernov et.al. [4]. Similar to our aim, their goal is to collect data sets about user behavior using a single methodology and a common set of tools. One of their main considerations is

to protect the data from unauthorized access. Because all the data is stored directly on the user's computer, it is up to the user to decide to whom and in what form the data should be released. However, it is not available and usable for mobile devices.

There is plenty of work in capturing and analyzing user movement using GPS and other positioning technologies. An example is the Geolife project [5]. The goal is to mine interesting locations and classical travel sequences in a given geospatial region based on GPS trajectories of multiple users. Their model infers the interest of a location by taking several factors into account. However, this work and other similar approaches in mobile user modeling only focus on single or a few sensors such as GPS and do not attempt to record all user actions on mobile devices.

The Mobile Sensing Platform described in [6] is an interesting system designed for embedded activity recognition. It incorporates multimodal sensing, data processing and inference, storage, all-day battery life, and wireless connectivity into a single wearable unit. However, it is an extra device the user has to carry, and the system cannot capture all the everyday activities users perform on their PDAs.

MobSens is a system to derive sensing modalities on smart mobile phones [7]. The authors discuss experiences and lessons learned from deploying four mobile sensing applications on off-the-shelf mobile phones in the framework that contains elements of health, social, and environmental sensing at both individual and community levels. However, the system's focus is on hardware sensors. Actions that users perform with software on a mobile device are not integrated.

MyExperience is an interesting project as it allows for capturing both objective and subjective *in situ* data on mobile computing activities [3]. The purpose of MyExperience is to understand how people use and experience mobile technology to be able to optimize the design of mobile applications, for example. Hence the system is not tailored towards user modeling, but it can serve as a foundation for our implementation, since the framework is extensible. We will therefore describe the MyExperience project in more detail in the next section.

3. Design and Implementation of the Mobile User Activity Logger

In this section we discuss issues concerning the design and implementation of our mobile user activity logger including the various hardware and software sensors. The logger is based on the MyExperience framework.

3.1. The MyExperience Project

MyExperience is a software tool for Windows Mobile PDAs and smartphones based on the Microsoft .NET Compact Framework 2.0 and the Microsoft SQL Compact Edition database. The software is available as a BSD-licensed open source project [8]. MyExperience runs continuously with minimal impact on people's personal devices. It has an event-driven, "Sensor-Trigger-Action" architecture that efficiently processes a variety of sensed events [3]. The collected data is enhanced by direct user feedback to enable capturing both objective and subjective information about user actions.

MyExperience is based on a three-tier architecture of sensors, triggers and actions. Triggers use sensor event data to conditionally launch actions. One novel aspect of MyExperience is that its behavior and user interface are specified via XML and a lightweight scripting language similar to the HTML/JavaScript paradigm on the web [8].

3.2. Overview of our mobile activity logger

As part of this work, we implemented 27 new hardware and software sensors and we used 11 existing sensors from the MyExperience project. Figure 2 gives an overview of available hardware and software sensors. Note that in our work a "sensor" more precisely is a piece of code that either connects to an actual hardware sensor on the mobile device, or reacts to software events or user input.

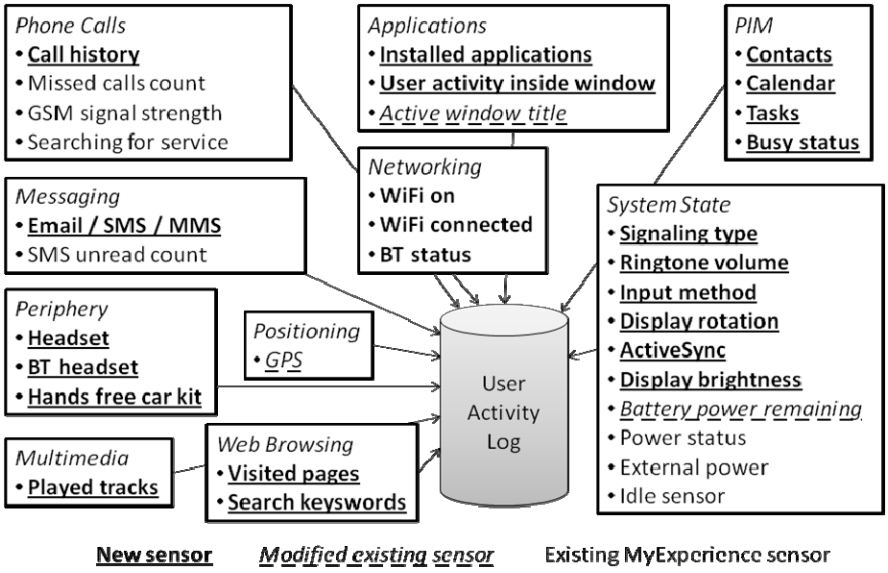

Fig. 2. Available sensors

MyExperience allows for configuring sensors via an XML file [8]. Note that the configuration not only controls which sensors to use for data recording, but MyExperience sensors also trigger actions such as starting an explicit user dialogue (Fig. 3, left). Since it is not viable to ask the end user to modify XML files on the mobile device, we have implemented an easy-to-use interface to activate and deactivate individual sensors (Fig. 3, right). Users may want to disable sensors for privacy reasons, and also to reduce power consumption or CPU load on the mobile device. The activity logger itself can be started and shut down manually by the user if necessary.

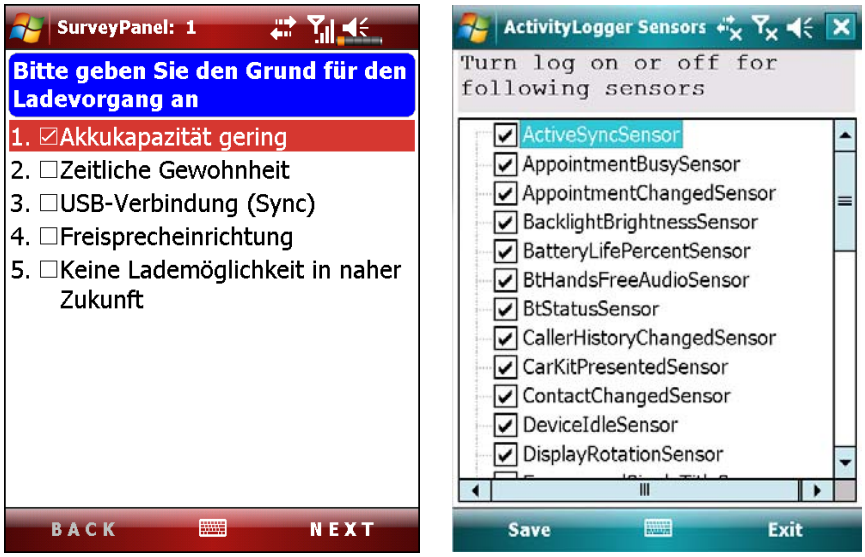

Fig. 3. Requesting explicit user feedback (left), and selecting sensors (right)

The implementation of sensors for implicit data acquisition can be summarized into the following categories:

- Application information such as visited web sites is usually stored in log files and local databases.
- Some sensors such as battery power status can be queried by using "SystemState" members of the .NET Compact Framework.
- Information about the location of local log files and some system information, such as display orientation and brightness, is available via the registry of the mobile device.

We will explain the available hardware and software sensors and issues concerning their implementation in more detail in the following subsections.

3.3. Sensors

3.3.1. Phone Calls

Making phone calls is one of the most important features of mobile devices. The fundamental parameters of a phone call are the phone numbers, the direction of the connection (outgoing, incoming, calls not accepted), the timestamp and the duration of the call. Furthermore, if the number of the other party can be found in the user's

address book, additional information like name and group membership (e.g. family, friend) can be determined. This information could be used to suggest a callee's phone number, for example, when a user accesses the phone function of her device at a certain day and time of the week.

The .NET Compact Framework offers a possibility to setup an event handler for incoming calls. However, a handle to log outgoing calls is not provided. Yet logging outgoing calls is important for mobile user modeling, because they are the direct result of a user action. Therefore, we implemented a sensor to log the stated information about all phone calls. This sensor uses a list of all calls the Windows Mobile operating system keeps in a file in the Embedded Database (EDB) format. Our framework uses this list to retrieve the call parameters, and conducts a reverse search in the user's address book to determine more information about the other party of a call if available.

We integrated sensors to count missed calls, for GSM signal strength and searching for service from the MyExperience project without modification.

3.3.2. Messaging and Personal Information Management (PIM)

Windows Mobile provides the Microsoft Office Outlook Mobile Tool for managing Emails and SMS messages. The program splits information about messages by different accounts. Access to the internal Outlook database is possible with a wrapper library "MAPIdotnet" in the "Messaging API" of the .NET Compact Framework. We used this API to retrieve information about incoming and outgoing messages. Our corresponding software sensor records one log entry for every Email/SMS/MMS message. Similar to phone calls, we store additional information of the sender or recipient of a message if the person can be found in the user's address book.

Outlook Mobile is also the default tool for personal information management (PIM) on a Windows Mobile device. Appointments (calendar), contact data or tasks (ToDo lists) are interesting categories of data for user modeling as well. We have implemented sensors to log changes a user makes in her PIM data. Basically, there are two options. The first one is to monitor the data in the local Outlook database "pim.vol". We can recognize changed entries by comparing the Outlook IDs before and after the usage of Outlook. We have implemented separate but similar sensors for calendar, contacts and tasks. These sensors are triggered when the system recognizes that Outlook is called up or shut down. The second option to log Outlook data is based on an event handler. In this case, the system is immediately modified when the user adds, modifies or deletes data in Outlook. Our sensor then generates a log entry that includes the ID of the corresponding data item.

3.3.3. Web Browsing

Analyzing the web browsing activities on the mobile device is an important part of mobile user modeling. We have created a MyExperience action to capture the usage of the Pocket Internet Explorer (PIE). It is not possible to directly query visited web sites in the .NET Compact Framework. However, the PIE manages information about

visited web sites, cookies and temporary internet files (cache) in three local files in different folders. The location of these files can be determined using the Windows Mobile registry. Access to these file is not permitted by the system if the PIE is running. Therefore, our sensor checks access to these files on start-up, and synchronizes the data with the activity log. The system keeps a timestamp of every accessed URL and visited web sites can hence be added to the activity log later on.

It is not only interesting for user modeling that a user has visited a web site, but also which keywords she has used for web searching. This information can be determined by analyzing the URL of web searches. For example, a query with Google leads to an URL similar to "http://www.google.com/search?q=activity+logging" in the log file. URLs to other search engine are comparable. We analyze and store the search keywords of about 20 search engines including Google, Yahoo and Bing, and also the query strings when accessing Wikipedia. Figure 5 (below) includes an example snapshot of recorded web browsing information.

3.3.4. Positioning

Obviously, one of most important differences between mobile and non-mobile systems is that the current user location is important in a mobile environment. Therefore it is important to log the user position in a mobile user modeling framework. There are a lot of work going on with regard to positioning systems, including approaches based on cell ID and WLAN access points. Since more and more mobile devices are equipped with a Global Position System (GPS) sensor, we decided to integrate GPS positioning in our framework.

The MyExperience project already includes a "GpsLatLongSensor" to trace GPS position. This sensor records GPS coordinates every one second. However, this leads to a lot of redundant GPS coordinates being stored in the user activity log which is not relevant for mobile user modeling. Therefore, we have extended this sensor with an option to configure a threshold. The threshold triggers when the parameterized distance to the last recorded location in meters is surpassed. Figure 4 shows an example configuration for our GPS logging sensor.

```
<sensors>
  <sensor name="GpsLatLongSensor"
      type="MyExperience.Sensors.GpsLatLongThresholdSensor">
    <property name="RecordStateChanges" value="true" />
    <property name="LogStateChanges" value="false" />
    <property name="ThresholdInMeters" value="500" />
    <property name="MinimumDopRequired" value="5" />
  </sensor>
</sensors>
```

Fig. 4. Configuration of the "GpsLatLongThresholdSensor" sensor

First tests with this sensor revealed problems with weak GPS signals. Especially when activating the GPS sensor, or leaving a building with no signal, the first log entries sometimes deviated from the actual position by several kilometers on our test devices. In addition, the system sometimes recorded "(0, 0)" coordinates with no signal. These phenomena are not a problem when using GPS for navigation, for example, because the system quickly calibrates itself and then provides correct coordinates. However, we aimed at avoiding these false values in our user activity logs. Thus, we implemented a solution based on the "dilution of precision" (DOP) parameter of GPS sensors. This value is determined by the GPS sensor itself and specifies the additional multiplicative effect of GPS satellite geometry on GPS precision. The lower the value, the more accurate the measurement. We obtained good results – i.e. inaccurate log entries were eliminated – with a minimum DOP value of 5 in our tests.

3.3.5. Networking and Peripheral Devices

State-of-the-art mobile devices usually support several technologies for wireless connectivity, including GSM, Wireless-LAN/Wifi and Bluetooth. A mobile system could utilize information about networking usage to automatically activate and deactivate connections based on past user behavior. We give a detailed example as a case study in our evaluation in Section 4. We have implemented different sensors to log when the user has turned on WiFi access, when the system is actually connected to a WiFi access point, and the Bluetooth connection status. Furthermore, our framework provides sensors to monitor peripheral devices such as a headset or Bluetooth hands free kits often used in cars.

3.3.6. Application Usage and Media Player

A mobile user modeling framework should be able to derive a possible correlation between application usage and sensor data. MyExperience offers a sensor to retrieve the title of the active window and thus determine the active application. However, it is possible that some applications are missed because this sensor queries the system periodically to determine this value. Therefore, we have modified this sensor using an event handler. In addition, our framework offers a sensor to log installed applications.

Logging user action inside an application is difficult in the Windows Mobile operating system, because this information is generally available inside the active process only. Therefore it is not possible to record the text a user enters on the virtual keyboard in a text processing program directly, for example. As an exception, the keystrokes on the hardware keys on a Windows Mobile device can be retrieved.

It is possible to build sensors for specific applications to be able to log more detailed information about application usage. As an example, we have implemented a sensor that logs the played tracks in the Windows Media Player. This information could be utilized later to provide context-aware media recommendations to the user.

3.3.7. System State

Finally, the last category of implemented sensors in our mobile user activity framework includes sensors that query the system state. Changes in the system state can be either triggered by user actions or an indirect result from usage of the device, for example battery power. Both are interesting for user modeling. Figure 2 lists the sensors we have implemented with regard to system state. An example is a sensor logging the input method a user selects. This information can be utilized to automatically select the appropriate input method based on previous user behavior. Windows Mobile devices with a touch screen usually offer a virtual keyboard and handwriting-recognition method such as Block Recognizer, Letter Recognizer or Transcriber. We implemented most of these system sensors by querying "SystemState" members in the "Microsoft.WindowsMobile.System" namespace of the .NET Compact Framework. The selected signaling type (vibration or ring) can be determined by querying a registry entry, in this case the variable "HKC\\ControlPanel\Sounds\RingTone0".

3.4. MyExperience Analyzer tool

The MyExperience framework stores all the information in a Microsoft SQL CE (Compact Edition) database on the mobile device [8]. The most important database table for our purposes is "SensorHistory" which stores the implicitly recorded data from the explained sensors. The MyExperience framework includes an "Analyzer" tool to manage and query the SQL CE database. We have extended this program with options to save and categorize queries. Queries are kept in an XML file, so it is possible to use them outside of the Analyzer tool.

Figure 5 depicts a screenshot of the extended Analyzer tool. On the left side, you can see the query library. This list corresponds to the implemented sensors in the categories explained above. On the right side of the window, an example query is shown. On top is the SQL CE query necessary to retrieve the Pocket Internet Explorer log entries.

Joining information from different sensors is possible but lead to rather complex SQL CE queries if done directly in the database. The Analyzer tool is intended to roughly check the collected data, not to interpret the gathered log data. For analysis and interpretation, the data can be exported from the database and further processed in data mining or other tools. For example, it is straight-forward to analyze where the user has performed certain actions. This is also included in the following evaluation section.

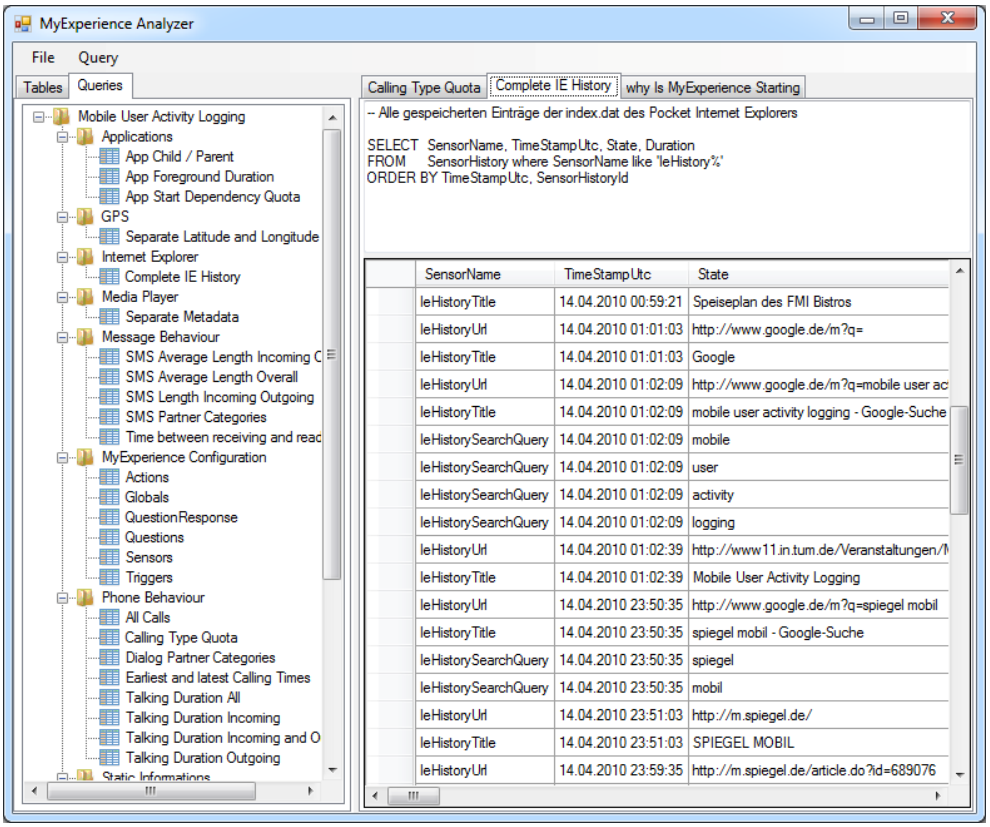

Fig. 5. Analyzer screenshot

4. Evaluation

In this section, we explain the evaluation of our approach. Note that we have focused on the collection of user data only at this time. Therefore, our approach and the evaluation do not cover the whole user modeling process (see introduction, Section 1), just the first step.

4.1. Experiences

We tested our sensors during implementation to make sure they perform accurately. Afterwards, we conducted a test run of the system lasting several weeks and included all sensors. During this time, 6748 log entries were recorded. In addition, we looked at scenarios to find out whether the recorded data can lead to meaningful data for user modeling. We describe one of these scenarios as a case study in chapter 4.2. We did

not include explicit user feedback (Fig. 3, left) in these tests, but this function could have been integrated easily.

Figure 6 depicts a visualization of parts of the logged data. For this visualization, the GPS position data was converted into the GPS Exchange Format (GPX) for Google Maps. The (blue) markers show locations where the user performed some activity on the mobile device. The figure depicts a typical scenario when a user travels from home to office during a work day. When looking at the data more closely, we were able to assess that the log files reproduced the user actions very well and in reasonable granularity for user modeling. Another example of the logged data is shown in the screenshot of the analyzer in Figure 5 (above). Thereby, the user was using her Pocket Internet Explorer to perform some web searches and the keywords of the searches were detected by the system.

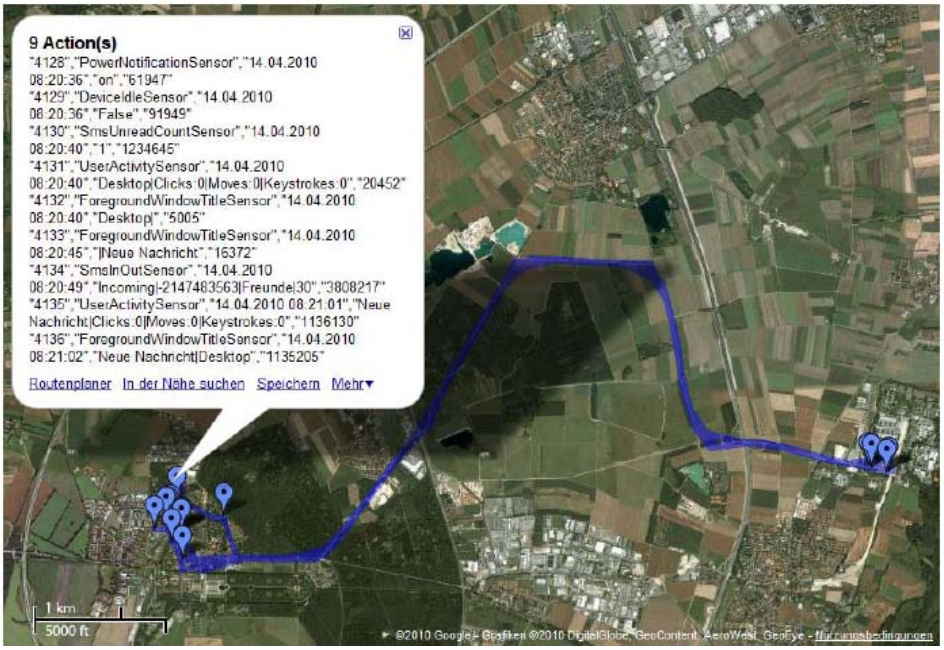

Fig. 6. Visualization of log data

Overall, our mobile logging framework performed well. There were a few program crashes in the prototype implementation but these occurred only very seldom. When the user was very active on her device and all sensors were enabled, the system performance degenerated somewhat. However, it is possible and assumed that not all sensors are active at all times. It is possible to deactivate sensors as explained above (Chapter 3.2). Overall, the logging did not obstruct the user experience significantly. Thus, our system complied with one of our main requirements: the implicit, non-distracting observation of user actions.

Apart from that, we have to note that, for an ongoing recording of sensor data, an active system status is required. Windows Mobile PDAs are usually configured to be hibernated when the user is inactive for some time. When this occurs, our logger is also stalled of course. However, with an inactive system, no meaningful user actions can be recorded anyways. If the user just turns off the display of her device, the recording of sensors such as battery power or GPS position continues. The battery power is shortened to a couple of hours at most without charging when all sensors are activated, but again the power consumption can be reduced by deactivating costly sensors such as the GPS module. It seems reasonable to define profiles with different sensors active (e.g. “indoor” with a disabled GPS sensor, or “light” with only a small subset of sensors active). Users would only have to choose among predefined profiles, not all sensors. But this profiling feature has not been implemented yet.

4.2. Case Study: WLAN Activation Based on User Position

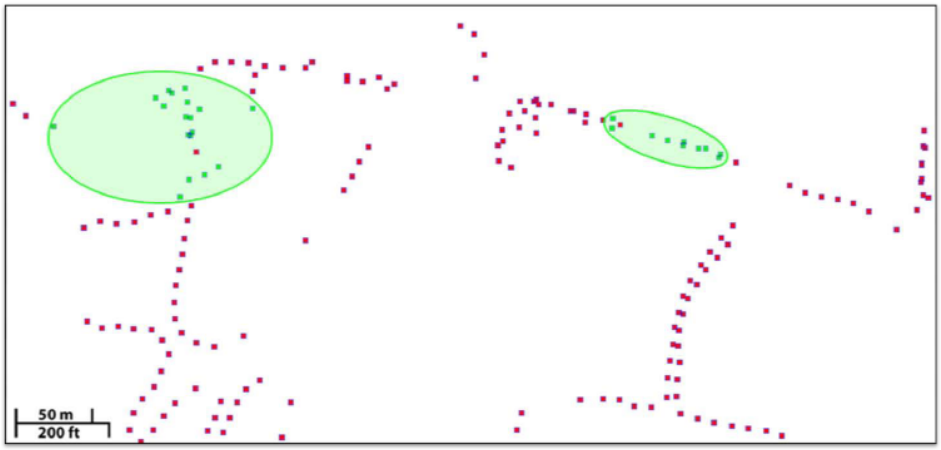

Fig. 7. WLAN activation based on position

In this scenario, we had a closer look on whether it is possible to identify locations where a user usually activates the WiFi/WLAN connection on her mobile device. The overall goal is that the system would then be able to automatically turn on WiFi when the user enters such a region. Thus, a combination of the “GpsLatLongThresholdSensor” with the “WiFiConnectedSensor” is investigated. In this test, the user moved her mobile device in an area with two WLAN access points. The GPS logging was set to store one log entry every 10 meters. The recorded position data was combined with the WiFiConnectedSensor data based on the timestamps of log entries. Figure 7 shows the graphical interpretation of the data. Dark (red) dots mark GPS positions with no WLAN activated, while the light (green) markings denote positions where the user has turned on WLAN. The (manually) highlighted areas indicate these geographic regions where the user usually activated her WLAN connection.

The recorded data corresponds very well with the actual WLAN access areas. We noticed that some of the points were slightly off when the user was moving fast. This behavior is due to slight delays when the system is observing and logging the deactivation of WLAN access. Overall, the recorded data seemed to be very useful for our purpose. Note again that the goal of this scenario was to evaluate whether the collected data can lead to meaningful results for user modeling. The scenario showed that a combination of sensors can be used to implement an adaptive function to automatically activate the WiFi/WLAN connection on the mobile device based on location. We have not investigated the actual data mining methods needed to identify such patterns so far. Yet our tests showed that our mobile user activity logger produced data in appropriate granularity for user modeling.

5. Conclusion and Outlook

The goal of this work is a unified approach for collecting data about user actions on mobile devices in a granularity appropriate for user modeling. To realize this first step of the user modeling process, we have designed and implemented a framework for mobile user activity logging on Windows Mobile PDAs based on the MyExperience project. We have extended this system with hardware and software sensors to monitor phone calls, messaging, peripheral devices, media players, GPS sensors, networking, personal information management, web browsing, system behavior and application usage. Our evaluation showed that it is possible to detect when, at which location and how a user uses an application or accesses certain information, for example.

Note that collecting data about user actions is more complicated on a mobile device than a desktop setting. This is due to restrictions in the available programming interfaces of the mobile platforms, in our case the Windows Mobile operating system, respective the .NET Compact Framework. We have explained some of the details of implementing the sensors in Section 3. The granularity or level of detail of the data collection is obviously dependent on the purpose of a subsequent user modeling task. We have aimed at selecting and designing sensors that lead to information which seems beneficial for learning user behavior in general. For example, our web browsing sensor records search keywords, but not single keystrokes a user may perform to fill in a web form. The framework can be used to implement new or modified sensors to fit special data collection purposes. In addition, properties of some sensors can be configured to adapt the data collection in more detail.

Future work includes integrating additional sensors. State-of-the-art mobile devices are more and more equipped with sophisticated sensors such as gravitation sensors or cameras that could be utilized for eye tracking. It is easy to integrate additional sensors in the MyExperience project and our framework. Portability and interoperability are also important issues. So far, our framework is tailored for the Microsoft Windows Mobile framework but similar tools can be implemented on other platforms like iPhone and Android. We are also investigating standards for interoperability of data collected on different platforms or logging frameworks. Existing relevant initiatives include the Attention Profiling Markup Language (APML) [9] and the Contextualized Attention Metadata framework (CAMf) [10].

One of the most important next steps of our work is also to investigate the analysis of the collected data using data mining and machine learning methods, and hence studying the second step of the user modeling process (see Section 1). It is also important to collect more substantial data sets in this regard.

Finally, our work focuses on observing one user so far. It may also be useful to take other users' logs into account and thus performing a "social" mobile user activity logging. The goal could be to identify situations where similar behaving users have performed certain actions, and personalize the mobile experience for the active user accordingly. In addition, our system also collects data about social interactions that can be utilized for analysis of social behavior.

References

1. Brusilovsky, P., Maybury, M.T.: From Adaptive Hypermedia to the Adaptive Web. Communications of the ACM, vol. 45, no, 5, pp. 30-33 (2002)
2. Subramanya, S.R., Yi, B.K.: Enhancing the User Experience in Mobile Phones. IEEE Computer, vol. 40, no. 12, pp. 114-117 (2007)
3. Froehlich, J., Chen, M., Consolvo, S., Harrison, B., Landay, J.,: MyExperience: A System for In situ Tracing and Capturing of User Feedback on Mobile Phones. In Proc. of MobiSys conf., San Juan, Puerto Rico (2007)
4. Chernov, S., Demartini, G., Herder, E., Kopycki, M., Nejdl, W.: Evaluating Personal Information Management Using an Activity Logs Enriched Desktop Dataset. In Proc. of 3rd Personal Information Management Workshop (PIM 2008), CHI conf., Florence, Italy (2008)
5. Zheng, Y., Zhang, L., Xie, X., Ma, W.: Mining Interesting Locations and Travel Sequences From GPS Trajectories. In Proc. of International Conference on World Wild Web (WWW 2009), Madrid, Spain, ACM Press, pp. 791-800 (2009)
6. Choudhury, T. et.al.: The Mobile Sensing Platform: An Embedded Activity Recognition System . IEEE Pervasive Computing, vol. 7, no. 2, pp. 32-41 (2008)
7. Kanjo, E., Bacon, J., Roberts, D., Landshoff, P.: MobSens: Making Smart Phones Smarter. IEEE Pervasive Computing, vol. 8, no. 4, pp. 50-57 (2009)
8. MyExperience project web site. http://myexperience.sourceforge.net/. Accessed June 2010.
9. APML web site. http://apml.areyoupayingattention.com/. Accessed June 2010.
10. CAMf web site. http://www.ariadne-eu.org/index.php?option=com_content&task=view&id=39&Itemid=55. Accessed June 2010.