GENERATING HIGH LEVEL CONTEXT FROM SENSOR DATA FOR MOBILE APPLICATIONS

Wolfgang Woerndl¹, Christian Schueller², Thomas Rottach¹²
¹ Technische Universität München, Institut für Informatik
Boltzmannstr. 3, 85748 Garching, Germany
{woerndl,rottach}@in.tum.de
² UnternehmerTUM GmbH
Lichtenbergstr. 8, 85748 Garching, Germany
schueller@unternehmertum.de

ABSTRACT
Context-awareness is an important element for developing mobile services. Often, context is available as sensor data but not in a form that can be exploited in (mobile) applications. In this paper, we present an approach to generate high level context from sensor data. First, we describe a process model and illustrate this model by a case study with an acceleration sensor. Then we explain the design and implementation of software components that are integrated in a framework supporting the development of mobile applications. Thus, we demonstrate the evolution of sensor data from raw data streams to aggregated data and to interpreted information.

KEYWORDS
Context, context-awareness, sensor data, mobile applications, learning classifiers

1. INTRODUCTION
For the development of marketable mobile applications, it is necessary to receive customer feedback by prototypes promptly. For the rapid prototyping, we created a framework “play.tools”, which makes it possible to develop mobile applications within a short time and test these applications regarding their technical feasibility, economy and acceptability at customer side (Woerndl, Schueller and Wojtech, 2007). The framework makes base software components available for end user applications, e.g. encapsulating the binding of Bluetooth devices, and communication with a database-supported server system.

In addition, it is important to adapt mobile applications to the current context of users. For example, it may be useful to track and record user location to reproduce a journey using GPS technology. In addition, there exist sensors other than GPS receivers to gain contextual information. Therefore, we propose a general process model to acquire context information with the goal of reusability of high level context for different applications. Thereby, it is possible to integrate different sensors. The approach is implemented in our play.tools framework and can be applied in different application scenarios.

The rest of the paper is organized as follows. In the next section we provide some background information on the notion of context and context-aware, mobile applications and also review related work. In section 3 we introduce our process model for the generation of high level context. In section 4 we explain the design, implementation and evaluation of our system that consists of two separate applications, the ContextLogger and the ContextAnalyzer. Finally, we conclude with a summary.

2. BACKGROUND
2.1 Context-sensitive mobile applications

All areas of life of humans become more and more affected by the presence of computers. In order that these computers are usable for humans, applications are needed that intelligently support the user dependent on its situation, without overstrain or divert him thereby. In the mobile domain, this is particularly important, because the interaction between humans and machines is made more difficult by small displays, input capabilities, shortness of resources (e.g. network bandwidth) and dynamic environments.

The entire process of the development of mobile applications is affected by the characteristics specified above. If the marketing of mobile applications is to have success in the long term, the manufacturers of mobile solutions must help to steer against this problems. The standardization of specifications represents an important step in the correct direction. But this is not the crucial point as above-mentioned. More important is to adapted applications to the needs of the users. For example the application adapts the GUI to the situation of a customer, as only needed functions are visible in one moment.

2.2 Context definition

A lot of work has been done regarding context and context-aware applications. However, it is not always clear, what context really means. In our work we follow the context definition of (Dey, Abowd and Salber, 2001): "Context is any information that can be used to characterize the situation of entities (i.e. whether a person, place or subject) that are considered relevant to the interaction between a user and an application, including the user and the application themselves". That means, context is very dynamic and transient. In the mobile domain, context attributes include location, movement, lighting condition or current availability of network bandwidth of mobile devices. Context can also have static components like pre-defined user preferences, we call these static parts (user) profiles. In this work, we are only dealing with the dynamic context that can be gathered in the mobile domain using sensors.

2.3 Related work

In this section, we outline related work from the area of context modeling and acquisition. Thereby, our attention is particularly towards the concepts which are used for the generation of high level context from sensor data. (Chen and Kotz, 2000) state that there are essentially three different possibilities for the collection of high level context:

- Image processing (is not really applicable for mobile applications because of limited resources)
- Use of user inputs, as for example calendar entries, in order to find out what the user does at a certain time
- Artificial intelligence methods, in order to get complex context by the combination of simpler low level sensor data

They argue that the processing of low level sensor data should be separated from the applications that are using it, which also motivates our work. However, they do not give concrete examples for the generation of high level context that could be used in our domain. As an alternative to just use GPS data as context, (Gellersen, Schmidt and Beigl, 2002) propose to combine and integrate different sensors. Their process is similar to ours (see chapter 3.1), including acquisition of raw data, pre-processing and analysis.

In order to classify context, (Korpipää et.al., 2003) use naïve Bayes networks. The procedure of the authors is motivated by the digital signal processing of audio data to infer the current activities of users, but they do not point out a generalized approach. (Himberg, Flanagan and Mäntyjärvi, 2003) show that data mining techniques are suitable for identifying high level context. In a first step, they collect information about the user (logs, calendar entries etc.), the physical environment (temperature, voice level, acceleration etc.) and the location of the user. The approach uses statistical methods to aggregate and cluster the data in subsequent steps. Mayrhofer (2004) describes a multi-phase software architecture for context recognition and prediction. The approach is more tailored towards the forecast of future contexts.
3. A PROCESS MODEL FOR THE GENERATION OF CONTEXT

In the last section, some related work was discussed. The overview of the literature shows that the different systems substantially differ in the used algorithms and in the application domain. However, most approaches are based on a comparable process for the generation of high level context, even if they specialize in different ways. In this section, we propose a refined process model consisting of the following four tasks:

1. Acquisition of raw data
2. Preprocessing: aggregation and feature extraction
3. Interpreting the data
4. Utilization in applications

Note that this is not a one-time process but a continuous activity to generate context. The first step is to acquire raw data from sensors for location, acceleration, temperature, sound/voice, imaging etc. For each sensor, a different software component has to be developed. Thereby, it is important to consider measurement ranges and error rates to get useful results. Thus, the acquisition of raw data allows for a perception of the physical environment and establishes the foundation for the other tasks. The second step is preprocessing of the (raw) data. This step varies significantly in the existing literature, but some kind of preprocessing is done in about every approach. One reason is that the sheer amount of data usually is huge after acquiring raw data, and cannot be handled directly, especially in our domain of mobile devices. Therefore, aggregation and feature extraction has to be applied. One example is to use “sliding windows” to extract features from data streams. Thereby, an algorithm examines only a subset of the whole data set at one time, and calculates characteristics such as mean values or standard deviation. The extraction window is then moved along the data stream. The size and overlap of the windows have an influence on the necessary memory and processing requirements, which is an issue on mobile devices.

The third step in our process model is interpretation which realizes the move from lower-level data to context information that is useful in applications. The interpretation runs in several steps, dependent on the used methods. Often data mining algorithms are applied such as Symbol Clustering Maps (SCM) (Himberg, Flanagan, and Mäntyjärvi, 2003) or supervised or unsupervised learning algorithms. After interpreting the data, the fourth step is to utilize the generated high level context in applications, or to visualize results for users.

We have experimented with several sensors including GPS receivers and Bosch SensorTec sensors, for example the acceleration sensor SMB 365 (Bosch SensorTec, 2007). This sensor records data along the three axes. The ultimate goal is to recognize user activities such as “standing”, “running” or “biking”. For the interpretation of this raw data stream, we use the mentioned sliding window approach. This means that every data window that is aggregated corresponds to about 5 seconds of an activity. The result of the preprocessing step for the acceleration sensors can be seen in Fig. 3, top (see chapter 4.4). Interpreting the acceleration sensor (step 3) means to figure out the actual activity of the user. In our case, we use the supervised learning algorithm J48 that Bao and Intille (2004) are describing. That means users have to enter some training data (see Figure 2, right, for the user interface). The algorithm builds a model of the (pre-defined) activities such as running, sitting etc. using a decision tree. The decision tree contains rules such as “xAxisMean > 20.054: SITTING”.

4. SYSTEM DESIGN AND IMPLEMENTATION

In this section we will show how the conceptual ideas of the last chapter can be realized in concrete software architecture. Thereby, we must map the process of generating high level context to a set of software components.

4.1 Architecture

There were several functional and non functional requirements we had to consider at the designing phase of our system. Besides the implementation of the explained process, the system should be easy to use and extend and integrate well in the client-server concept of our software development framework “play.tools”.
Since we were mainly interested how the generation of high level context could work for us, we decided to use mobile devices only for the collection of rich sensor data. The interpretation of the data was performed on a desktop PC running our interpreter components. Since we are using J2ME, one can easily port those components to the favored end user device or run them on the server provided by the play.tools framework.

Fig. 1 shows the layers of our architecture. There is a layer for each step of the generation process for high level context (cf. Section 3). In addition, there are some auxiliary layers that are necessary for controlling the sensors, transmission of sensor data (export/import) and visualization of generated high level context. The layers are grouped into two applications: “ContextLogger” for gathering context data and writing it to log files and “ContextAnalyzer” for processing the log files and generating high-level context. In the following sections we will discuss those two applications in more detail.

4.2 Gathering context with ContextLogger

The ContextLogger is a mobile application for the collection of rich sensor data from several sensors. It is possible to control the logging of sensor data over an easy to use user interface (Fig. 2). Besides the pure logging functionality, ContextLogger offers the opportunity to annotate the log data with meta information regarding the user’s current activity. That is very important if one wants to use a supervised learning approach for the interpretation of the data. Note, that we do not force anybody to specify that meta information and if you only want to use an unsupervised learning approach you might skip that step. For testing purposes and to have some data for our case study we integrated a GPS device and an acceleration sensor into the ContextLogger application.

Figure 1. Architecture

Figure 2. Main menu (left), managing different sensors (middle), annotating an activity (right)
With ContextLogger, we offer a tool for the generation of real life sensor data. The application allows easy integration of additional sensors and helps to understand the process of generating high level context. Since it is an MIDP 2.0 application it can run under a variety of current mobile phones allowing cheap large scale data sampling with multiple users.

4.3 Aggregating and interpreting context with ContextAnalyzer

After the collection of sensor data with the application ContextLogger, we come to the step of feature extraction from the sensor data stream (preprocessing). Thereby the architecture dedicates a component for each sensor, which is responsible for extracting features from a single sensor data source. The target of the feature extraction step is to increase the interpretability for the following interpretation by reducing the amount of data and giving emphasis to important aspects inside the data (removing redundancy and computing characteristics). Note that the annotations (the meta information specified by the user) are treated the same as the other sensor data sources.

We come now to the interpretation layer. This layer basically includes a set of interpretation components which get their input data from the components inside the feature extraction layer and other interpretation components. Each interpretation component is performing an abstraction step by interpreting an aggregating data. This is where the magic happens and high level context is generated. Using the Weka toolkit for the implementation of our case study we could show the validity of the process of generating high level context (Witten and Frank, 2005). The following three tasks are implemented inside the interpretation layer:

- Learning classifiers
- Load and save classifier-models (XML format)
- Classify new data using the learned models (validation)

Since pure numbers are hard to understand for humans, visualization of data and interpretation results has an important role in our architecture. This helped us to make an easy interpretation of the results possible and feasible for human beings. All this is implemented in the ContextAnalyzer application. With this application, we introduce a platform for the development of algorithms for the generation of high level context. Users can import log files and design, implement and test their algorithms without being slowed down by the

Figure 3. Visualizing results acquired with the acceleration sensor

Since pure numbers are hard to understand for humans, visualization of data and interpretation results has an important role in our architecture. This helped us to make an easy interpretation of the results possible and feasible for human beings. All this is implemented in the ContextAnalyzer application. With this application, we introduce a platform for the development of algorithms for the generation of high level context. Users can import log files and design, implement and test their algorithms without being slowed down by the
constraints of mobile devices at an early stage. This is an important basis for porting the ideas to mobile devices, since it allows playing with real data and helps to get a better understanding of the problem domain.

4.4 Evaluation

We have implemented the ContextLogger and ContextAnalyzer applications and experimented with real world data. Figure 3 shows the visualization of the acceleration sensor. The aggregated values of the 3 axis are plotted and in the bottom third of the screen, the inferred and actual user activity are displayed. We have not done a formal study regarding the accuracy of the results yet, but our tests indicate that the algorithm is performing reasonably well and can be used to develop an application which monitors, store and analyzes user activities. Some problems existed when activities are somewhat related. For example, it is difficult to distinguish between “sitting” and “(sitting in a) car”. Therefore at timestamp 950, the algorithm first inferred that the user is driving, even though that was not the case.

5. CONCLUSION

In this paper, we have explained the generation of high level context from sensor data in the domain of mobile devices although our method may be useful for other application areas as well. We have first defined a process that is grounded on existing approaches in the literature and designed and implemented respective software components. The design and implementation of our system is very extensible, so it is easily possible to integrate additional sensors and new learning classifiers, for example. The evaluation shows that our approach can produce meaningful result which can be utilized in mobile applications. Apart from the acceleration sensor, we have experimented with other sensors, most notably GPS receivers to retrieve location data.

The ContextLogger and ContextAnalyzer are integrated in the play.tools framework to support developers of end user applications, as explained in the introduction. This allows for reuse of context information and sensors in different applications. We are currently working on the 2nd iteration of our framework and also are integrating more sensors. In addition, we are currently developing mobile end user applications to test our overall approach in practice.

REFERENCES