A Webservice-based Context Data Service for the Android Platform

Dirk Bade, Roman Puszies

University of Hamburg
Department of Informatics
Distributed Systems and Information Systems
Vogt-Koelln-Strasse 30, 22527 Hamburg, Germany
[dirk.bade,roman.puszies]@informatik.uni-hamburg.de

Abstract: In recent years, a lot of attention has been attracted to sensors in mobile devices. Currently, the location of a user is the most commonly used kind of context information, but as more and more physical as well as virtual sensors are integrated into mobile devices and applications, new kinds of context-based services are conceivable. This in turn, leads to enhanced requirements for client-side applications and services, resulting in even more development effort and hence an increased time-to-market. Development frameworks for mobile devices try to counteract this trend by offering enhanced APIs, supporting the development of complex software components. A prominent example which has caused a lot of attention in recent years is the *Android* platform, which offers more higher-level programming concepts compared to other platforms for mobile devices, but unfortunately only primitive support for context mediation. In this paper, we therefore introduce a software component for easing the mediation of context data between mobile sensors and complex context-based applications and services. We analyze the requirements for mobile applications imposed by such complex services, present a generic high-level architecture of a software component meeting these requirements and highlight some implementation details of our prototypical implementation *SenseDroid* for the Android platform.

1 Introduction

Today, services aiming at mobile users need to take the users current context into account in order to provide the right information, to the right time at the right place. For this purpose, service provider require information about the user's current situation and her environment in order to adapt the service to the user's current needs. Providing the location is among the most important requirements for location- or more generally speaking context-based services, but there is more to context than location [SBG99]. Nowadays, mobile phones and other devices are equipped with a multitude of different sensors, whose observations could help adapting a service even better. Among these are sensors measuring attributes of the physical environment (e.g. location, temperature, current acceleration, luminance, noise, battery state, etc.) as well as virtual sensors providing e.g. calendar entries, a history of user actions or performance indicators like available memory, CPU load, etc.

Nearly all context data is collected on the mobile device itself and has to be explicitly made available for service providers (except location information, which might also be provided by the carrier). In this respect, two modes of context data transfer between the client device and the service provider must be distinguished:

- a simple request/response-schema in which the client initially pushes all context information required by the service provider as payload of its request, which enables the provider to directly answer the request as all necessary information has already been provided.

- a more complex schema, in which the client requests a service and the service in turn uses intermediary communication acts in order to enquiry information from the client device once demand arises.

While the first case is trivial and is widely used nowadays by location- and context-based services, the latter allows to provide more sophisticated services that make use of callbacks to pull relevant context data, in which the relevancy may depend on previously pulled data. This way, a service provider can adapt its services to the user's current situation even better and help to save bandwidth as only required information is transferred. But this poses some requirements on the client's ability to answer the provider's requests.

Up to date, no mobile platform natively meets these requirements and developers have to spend time and effort to integrate the required functionality in their context-based applications. Moreover, applications and services are generally tightly coupled, due to the lack of standardized interfaces for context mediation. This results in significant development effort to re-adapt the interfaces once one of the endpoints is exchanged. In this paper, we therefore introduce a self-contained context data service, which delegates local as well as remote access to sensor information through standardized interfaces, supporting arbitrary context communication acts taking place between a service and its clients. This way, application developers are unburdened from the need to collect and communicate context data themselves and can benefit from higher-level access to these tasks.

The context service is realized for the Android platform [Ope], an extensible open-source platform for (mobile) devices like phones, set-top boxes, tablets, etc., which has already gained a significant worldwide market share (approx. 17.7 percent [Gar10], two years after its rollout in 2008) and increasing attention from industry as well as academia. Besides this fast adoption, the platform also offers some technical benefits, in particular the possibility to access system internals by using high-level programming languages (Java), which therefore makes it ideally suited as a context delivery platform for context-based services.

The rest of this paper is organized as follows: Section 2 details two application scenarios and infers requirements for a context data service, whose design and implementation is briefly highlighted in Section 3 and Section 4 respectively. In Section 5 an overview of related work is given and finally Section 6 concludes and presents our prospects for future work.

2 Challenges

In the following, two application scenarios are introduced. Using these scenarios several requirements for sensor access and context data communication are inferred afterwards.

2.1 Restaurant Finder

Alice and her colleagues attend a conference taking place at a university. As the local cafeteria is said to serve low-quality food they decide to look elsewhere to grab something to eat. Time is short and hence Alice requests the service of a context-based restaurant finder. Using only the location as primary filter for restaurants nearby the university, the result set is too large for transmission and presentation. Therefore, the service calls back on Alice's device in order to find out about additional context data she likes to share. Luckily, her phone runs a profile sensor, which is capable of providing her preferences. With the information that Chinese is Alice's favorite dish the result set can be reduced to an acceptable size and the request can finally be answered. Further service-side enquiries for additional context data in case the result set is still too large can be imagined, e.g. information about the restaurants Alice ate at during the last month or the time left until her next appointment as scheduled in her calendar.

2.2 Mobile Consumer Marketing Research

In the Mobile Marketing Research scenario a market research company pays scouts on a "'per job"' basis. Bob is already registered as a mobile scout and once he has some spare time, takes a job to earn some money or gift vouchers. This time, Bob arrives early at the train station and finds himself being bored while waiting for the train to arrive. He has 30 minutes left, so he decides to take a scouting job and uses his mobile phone to connect to the market research online service. The service in turn requests his location and, using the mobile phone's calendar sensor, the time left to his next appointment - the train departure. 30 minutes is just enough time to scout a product in a small train station supermarket. Bob accepts this job and from now on, his location is monitored by the market service to find out if he really visits the supermarket. Standing in front of the product shelf, Bob is sent a questionnaire about the product to fill out. Depending on his answers the service might request additional context information from the phone's sensors, e.g. the current temperature or humidity, a product photo taken by the internal camera or the electronic product codes (EPCs) of other nearby products using the NFC-capability of the phone. After completing the job, Bob is gratified with a gift voucher to equip himself with food for the train ride.

2.3 Requirements Analysis

Both scenarios show the need for intermediary communication when invoking more complex context-based services. While in the first scenario the additional information is only used to filter out entries from the result set, the second scenario presents an approach to integrate a mobile device in a more complex business process, where individual tasks require sensing the context of a person or a nearby object respectively. Based upon these application scenarios we inferred some requirements concerning a context data service running on a mobile device to answers requests for context data. Its noteworthy, that in our vision both scenarios as well as any other scenario which solely relies on context data, no additional software is required to be installed on the device. Therefore, the following requirements represent a common minimal denominator to interact with a wide range of context-based services. We split the requirements analysis into two aspects: data collection and data communication, as these represent two completely different and independent aspects. Beginning with data collection, a context service shall meet the following requirements:

Sensors At first, a mobile device or a context service respectively, shall support different kinds of physical as well as virtual sensors. While physical sensors (e.g. location, temperature, acceleration) can typically be accessed by any application running on the device using specialized API-calls, virtual sensors are more difficult to access as these may be integrated into other applications running in different processes and hence address spaces.

Task-Ability Moreover, the sensors shall be *task-able*. A context data consumer shall have the possibility to send tasks, i.e. new configuration settings like update rate, accuracy or specific sensor properties, to a sensor in order to adapt the measurement process to its own needs.

Dynamic Sensor-Set There is a multitude of different context data types a device offers. For each type, a dedicated physical or virtual sensor is required. For privacy reasons and in order to save resources like CPU, memory and energy the user shall be able to (de-) activate sensors at runtime resulting in a dynamic sensor-set that is offered to consumers.

Access The sensors shall be accessible not only by remote services, but also by local applications running on the device itself.

1-n Relationship Sensors shall be able to provide measurements to more than one consumer at the same time, thereby obeying the possibly different tasks imposed by the consumers.

Concerning the aspect of context data communication, we identified the following requirements:

Initiation The communication shall always be initiated by the client or the mobile device respectively. This way, the user is always aware of the fact, that context data is going to be exchanged. On the other side, this circumvents problems like changing IP-addresses and NAT-routers between the IP-networks of mobile communication service providers and the Internet.

Protocols As could be seen in the market research scenario, complex communication protocols might take place once a remote service requests context information from a mobile device. In that scenario the device initially pushed its location data to the service, the service in turn subscribed to the device in order to monitor its location over a period of time and to sent a questionnaire to the Bob once he arrived at the supermarket. The server also pulled information from the device, namely a camera photo as well as the EPCs of surrounding products. Therefore, at least simple push- and pull-based protocols as well as a publish/subscribe-protocol have to be supported by the mobile device.

Interface Due to the heterogeneity in the area of mobile computing, the communication interfaces should be relatively ”technology-neutral”, i.e. be usable by any consumer independent of e.g. the programming language the consumer is realized with. This requirement does not hold for the interface used by local applications, as these are expected to be able to handle technology-dependent interfaces and might additionally benefit from a performance gain as technology-dependent interfaces can normally be realized more effectively.

3 Design

Based upon the requirements analysis, we identified a set of main building blocks for a context data service, that is capable of delegating sensor access for local as well as remote applications. As already mentioned, we chose the Android platform for realizing the context service (cp. Section 4). Some design decisions have therefore been specifically targeted to this environment and are not generally applicable to mobile platforms in general (e.g. the inter-process communication required for accessing local applications' virtual sensors). The high-level architecture of the service, presented in the following, already takes these design decisions into account.

3.1 Sensor Model

In order to query and task sensors we employ a very simple sensor model. Physical as well as virtual sensor are represented by property maps. For each property, the map basically specifies the informal semantics of a property, its data type, and whether it is writable (e.g. sensor attributes like update rate, etc.) or only readable (e.g. sensor perceptions). The main reason to prefer such a simple model over more sophisticated ones is the ease of access for service users.

3.2 High-level Architecture

The context service's architecture (cp. Fig. 1) basically consists of five components, whose main concerns are sensor access, sensor management and communication. In the following, each of the components is briefly detailed.

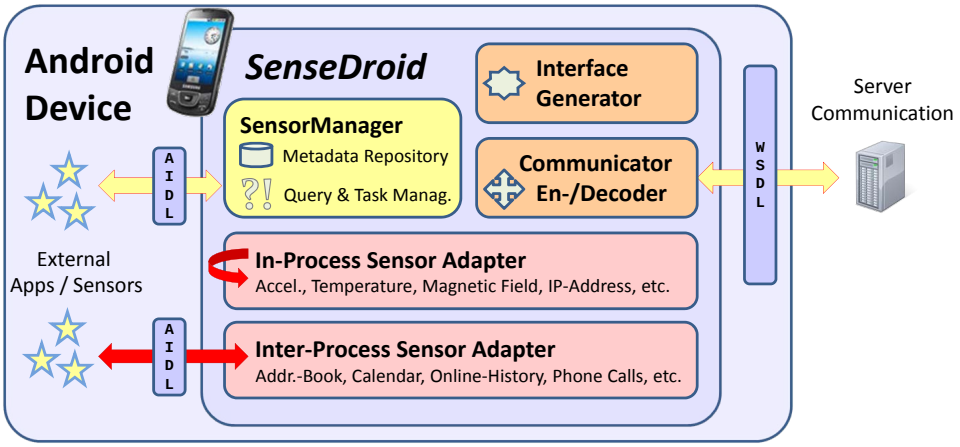

Figure 1: High-level Architecture

Sensor Adapter Sensor adapters are responsible for accessing virtual as well as physical sensors. In this respect, two different kinds of adapters must be distinguished: i) an *In-Process Adapter* is used for all sensors, which are - from a technical point of view - directly accessible from within the context sensor's process. This comprises all sensors accessible via API calls, e.g. location, temperature, IP address, etc. ii) In contrast, an *Inter-Process Adapter* needs to provide an interface for inter-process communication, as virtual sensors often reside in different processes or address spaces respectively, which prevents the traditional exchange of data using shared variables or method calls.

Sensor Manager This instance actually mediates queries, tasks and sensor perceptions between sensors and consuming applications. It keeps account of all sensors registered with the service and stores this information together with further meta data and the subscribed consumers in a persistent repository.

Interface Generator As sensors may be (de-)activated during runtime the communication interfaces have to be dynamically rebuilt in order to reflect the current range of available sensors and the operations they provide.

Communicator and En-/Decoder This component represents the gateway to the external world. On the one hand, it is responsible for managing communication channels and on the other hand it takes care of en-/decoding outgoing and incoming messages.

This architecture is rather slim-lined. But several extension points make it possible to extend the functionality in different ways as new requirements arise. For example, multiple different communication interfaces may coexist within the service and the same also holds for en-/decoders.

4 Implementation

The implementation of SenseDroid follows the architectural design presented in Section 3. We realized the prototypical implementation as an Android service, running in the background and

listening for incoming tasks, queries, subscriptions and context updates. The *In-Process Sensor Adapter* makes use of the Android API to access physical sensors as well as virtual sensors responsible for monitoring the IP address, addressbook or calendar. The *Inter-Process Sensor Adapter* offers an interface based on the *Android Interface Definition Language* (AIDL) to communicate with virtual sensors residing in other address spaces (i.e. applications). Further details about the implementation can be found on the SenseDroid homepage[1].

One important, and still open concern is the implementation of the webservice interface, which is dynamically generated on the basis of the meta data repository once sensors are (de-)activated. The discussion whether to explicitly publish all attribute-operations for all sensors or to provide generic, parameterized get-/ and set-operations along with access to all meta data is currently ongoing and might lead to changes in future releases.

5 Related Work

Context data collection, representation, distribution and usage has been discussed in numerous publications over the years. The advent of more powerful mobile devices, that are equipped with multiple sensors bore several projects that not only focused on theories and methodologies but actually provided implementations to practically demonstrate the feasibility of concepts.

The *ContextPhone* [ROPT05] for example is an extension for the Symbian OS, incorporating several virtual sensors, which observe the phone's current context. This context information is used by other local applications or communicated to other ContextPhones. To the best of our knowledge, the context data is not intended to be provided for other external applications like context-based services and therefore only proprietary communication interfaces are offered.

The work by [ZKL09] addresses mobile devices as active participants in business processes. For this purpose, a J2ME-enabled mobile device implements a workflow engine and is able to execute mobile workflows, which migrate through the network in search of an appropriate execution environment, which is determined based on its current context. This work mainly focuses on the ability of workflows to migrate, rather than on context data mediation, but both aspects are also realized with webservice-based interfaces.

Mobile devices offering context data through webservice interfaces are also used within the *MyCampus* infrastructure [SSG04]. In this project, services running on mobile PDAs are semantically augmented so that they can be automatically discovered and used by other services residing in the infrastructure. This however, requires additional efforts from sensor and application developers and is not suitable for a broader audience. Further research efforts in this area are surveyed by [UBPU08].

Tasking of sensors for the sake of adapting the measurement process to one's own need is tackled by the *Sensor Web Enablement* initiative [BPRD08], which standardizes, among several service roles for sensor data mediation, also a modeling language for sensors: *SensorML*. This language also contains schemes for the tasking of sensors, which we plan to adopt in the future.

[1] http://vsis-www.informatik.uni-hamburg.de/projects/sensedroid

6 Summary and Future Work

In this paper, we addressed the need for a context data service running on mobile devices as a precious resource for context data. Two application scenarios demonstrated our vision of a slim, easy-to-use service component that eases access to context-based services and allows to seamlessly integrate mobile devices into more complex business processes, thereby relieving application developers from low-level details such as sensor access and communication issues. We also briefly highlighted some implementation details of our prototypical implementation for the Android platform called *SenseDroid*[2], which is available under the LGPL.

Our prospects for future work include the integration of *SenseDroid* as a context delivery platform into more complex middleware architectures like *JESPA* [BL09]. In doing so, new application scenarios will be realized, verifying the completeness of our requirements analysis and the context service's feature set. Moreover, security is an important issue, that we need to take special care of. Finally, as mobile devices become more powerful, new extensions like on-device context aggregation will be subject for further research.

References

[BL09] Dirk Bade and Winfried Lamersdorf. An Agent-based Event Processing Middleware for Sensor Networks and RFID Systems. *Computer Journal, Spec. Iss., Agent Technologies f. Sensor Networks*, 2009.

[BPRD08] Mike Botts, George Percivall, Carl Reed, and John Davidson. *OGC Sensor Web Enablement: Overview and High Level Architecture*, volume 4540/2008 of *Lecture Notes in Computer Science*, pages 175–190. Springer Berlin / Heidelberg, 2008.

[Gar10] Gartner Inc. Gartner Says Android to Become No. 2 Worldwide Mobile Operating System in 2010 and Challenge Symbian for No. 1 Position by 2014. Press Release, September 2010. http://www.gartner.com/it/page.jsp?id=1434613.

[Ope] Open Handset Alliance. Android Overview. http://www.openhandsetalliance.com/android_overview.html. visited 03.10.2010.

[ROPT05] Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE Pervasive Computing*, 4:51–59, 2005.

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans Gellersen. There is more to context than location. *Computers and Graphics*, 23:893–901, 1999.

[SSG04] M. Sheshagiri, N. M. Sadeh, and F. Gandon. Using semantic web services for context-aware mobile applications. In *Second International Conference on Mobile Systems (MobiSys 2004), Applications, and Services - Workshop on Context Awareness*, 2004.

[UBPU08] Aitor Urbieta, Guillermo Barrutieta, Jorge Parra, and Aitor Uribarren. A survey of dynamic service composition approaches for ambient systems. In *SOMITAS '08: Pro. of the 2008 Ambi-Sys workshop on Software Organisation and MonIToring of Ambient Systems*, pages 1–8, 2008.

[ZKL09] Sonja Zaplata, Christian P. Kunze, and Winfried Lamersdorf. Context-based Cooperation in Mobile Business Environments. *Business & Information Systems Engineering*, 1:301–314, 2009.

[2] http://vsis-www.informatik.uni-hamburg.de/projects/sensedroid