# Activity-oriented Context Adaptation in Mobile Applications

Jan D. S. Wischweh
University of Hamburg
Hamburg, Germany
`mail@wischweh.de`

Dirk Bade
University of Hamburg
Hamburg, Germany
`bade@informatik.uni-hamburg.de`

**Abstract**

Although usability of mobile devices increases steadily, use of mobile applications is still inconvenient. Adapting application behavior and functionality to the user's current needs is a promising approach to compensate for limited input capabilities. Despite great effort in research, smart adaptable applications are still rare. With our approach, we build upon existing works and extend them with the notion of *activity context*. Activities are one of the most basic elements of context and are well suited to determine the relevance of context entities in a given situation. Such information can be used to realize more intelligent suggestion mechanisms for input elements in mobile applications. The feasibility of our approach has been proven by a prototype implementation of our *Activity Awareness Architecture* for the Android platform providing activity context for mobile applications and a context-aware calendar on top of it demonstrating the usefulness of activity context.

## 1 Introduction

Throughout the last decade, the use of mobile phones increased steadily. In particular, smartphones are gaining more and more interest [14] as these offer, beside a wide range of communication capabilities, a platform for a multitude of applications, ranging from entertainment to serious business appliances. Along with the growing market of mobile applications the need for user-friendly interaction attracts more attention, because users tend to get accustomed to accessing data and services whenever and wherever they are.

To ease interaction with mobile devices manufacturers increased screen sizes, made them touchable, and attached small sensors (accelerometers, gyroscopes, etc.). By doing so, the devices are enabled to exploit context to adapt application behavior to the current situation, e.g. adjust screen orientation, provide location-based services or filter ambient noise. Research in this area mainly focused on aspects about the physical environment (e.g. [12]), environmental conditions (e.g. [37]), available resources (e.g. [29]) and the user's social context (e.g. [28]) so far. Information about the user himself, his activities as well as causal and temporal relations between context entities are hardly exploited as a source for adaptation yet. Thereby, knowledge about the user's past, present and future activities is not only suited to ease input by presenting context-based suggestions, but may also be used to help automating recurring activities (i.e. tasks) to establish relations between people, places and objects and to infer the user's focus of attention and his cognitive load.

In this paper we propose to use *activity context*, which is inferred by monitoring the user's activities while he is interacting with a mobile device. This kind of context relates to goal-oriented tasks the user performs and is therefore ideally suited for adapting application behavior.

Section 2 of this paper details an application scenario and infers conceptual requirements for our work. Section 3 introduces the idea of activity context and relates it to context in general. In Section 4 a context model is proposed, which is used as a basis for our *Activity Awareness Architecture*, presented in Section 5. We conclude this paper by providing some details of the middleware and demo application we developed to show the feasibility of our approach in Section 6 and by giving our prospects for future work in Section 7.

## 2 Application Scenario

Alice and Bob want to conduct a meeting. For this purpose, Alice calls Bob to make an arrangement. Afterwards, she opens the calendar application on her mobile phone to create a new appointment. To ease the input the application makes suggestions for each of the input fields. Hence, Bob as well as other people related to Bob are suggested as participants. The calendar also suggests meeting locations like Bob's office or places Alice and Bob met before. This way, the need for textual input is reduced to two simple selections. Moreover, other applications, e.g. a navigation app, can make use of the provided information and once the time of the meeting approaches the navigation app, considering Alice's current location and the traffic situation, informs her right on time to prepare to leave.

### 2.1 Scenario Analysis

*Usability* can be seen as a measure of simplicity to reach a specific goal using certain technology [10]. Concerning software artifacts, it is undoubted that the usage context greatly influences the handling of applications and tools [10]. This holds even more, when considering mobile devices, as their usage context is highly dynamic. To overcome usability problems, an application should be able to adapt its functionality to the current context. Three areas of adaptation can be distinguished: adapting the (i) interaction, (ii) content and (iii) presentation [22]. In our approach, we do not propose adaptation mechanisms, but we want to provide application-level information about the user's (activity) context, so that applications can use this to carry out some adaptation logic. Analyzing the presented scenario, some important aspects that need to be considered when using activity context can be identified:

**Temporal Relations** In addition to a user's current activity, also past and future activities are important context information. Past activities constitute the *persistent context* (e.g. places where Alice and Bob met before), whereas future activities are a result of *prediction*.

**1$^{st}$- and 2$^{nd}$-order Relations** In the given scenario the calendar suggests Bob to be a participant based on the fact that Alice just phoned Bob. This is a 1$^{st}$-order relation. The suggestion to meet at Bob's office is instead based on a 2$^{nd}$-order relation. Theoretically, relations are not restricted to just two orders, but while the set of related entities grows exponentially with increasing order the relevance probability decreases drastically.

**Restriction of Entity Types** For every application, only certain types of entities are relevant. The calendar app in the given example only deals with persons and locations linked with appointments. A navigation app is interested in locations only. The software in use therefore has to restrict the entities to consider based on their types.

**Cooperation of Tools** The scenario shows, how different applications may cooperate using context information. The more applications create and use context information, the higher its value for further usage. The calendar app in the scenario makes use of three different context sources and represents a new context source for the navigation app. Thereby, it is not required that applications know each other, but they should instead be loosely coupled.

**Bidirectional Flow of Information** As can be seen, the calendar app not only consumes, but also creates new context information which can be used by other applications (e.g. the navigation app). Therefore, the flow of information between applications, or applications and some mediating instance, needs to be bidirectional.

Now, that the idea of using information about a user's activities to adapt application behavior in order to enhance usability has been introduced, a formal view on activities is presented in the next section.

## 3 Foundations of Activity Context Adaptation

Schilit at al. coined the term *context aware computing* for mobile computer systems, that are able to automatically adapt themselves to the current situation [35]. Various examples for such adaptation exist [9, 29, 7, 28]. The examples range from transparent adaptation to active context-aware behavior where the exploited aspects of the situation range from location to available resources and the social situation.

Because of this broad spectrum, context is quite an abstract term, thus multitude of definitions exist. An often cited definition by Dey and Abowd [11] states, that "*context is any information that can be used to characterize the situation of an entity. An entity is a person, place or object [...]*". Several works (e.g. [7, 25, 17]) criticized this definition due its lack of behavioral factors, e.g. activities. Therefore, we follow a modification of Zimmermann et al. [43], stating that: "*[...] Elements for the description of this context information fall into five categories: individuality, activity, location, time, and relations.*". Activities are not seen as context entities, but rather represent relations between such entities [33]. These relations can be based on joint participation as well as temporal or spatial proximity and can be used to determine the relevancy of entities for a given context [13, 15]. Moreover, [13] indicated that context and activities inevitably belong together as context is primarily created by activities.

In computer science different perspectives on *activities* exist. For example, *human engineering* experts observe and analyze human activities while handling software artifacts in order to understand and optimize their interactions [26]. In contrast, autonomous *software agents* pursue their own goals and in doing so execute plans and actions to reach them [5]. In the area of *user modeling* a formal, domain-dependent model of users, their goals and actions is created in order to adapt a system's behavior to the users' needs [22].

Our understanding of activities is based on the *activity theory* which tries to comprehend the course of activities by incorporating external influences as well as the actors themselves [23]. This scientific/philosophical way of thinking, stemming from the area of psychology, is based on several basic principles that allow us to establish an understanding about (i) the different ways to automatically monitor activities by a software system and (ii) what parts are out of the scope of monitoring. In the context of our work one of the most important principles in the activity theory is the object-orientation which states that objects always carry socio-cultural knowledge of what can be done with them and how it can be done. Another principle states that activities are mediated through physical and non-physical tools. For example, language influences the way we think about problems. Last but not least, activities are hierarchically composed of actions which in turn are composed of operations. A corresponding hierarchy of motives, goals and conditions exists linking behavioral levels with levels of purpose. Activities and motives are on the top level, they are conscious and long-lasting. Operations and conditions are at the bottom level. Like actions and goals they are short-term, but unlike them they are unconscious.

As activities are carried out over a longer period of time and are, depending on situations, alternated with other activities the activity theory leads us to the conclusion that it will not be possible to completely assess a user's activities and his motives can only be presumed indirectly at best. Furthermore, we can only observe user actions which are carried out using physical (software-)tools. But activity theory also provides us with a starting point for doing so, as in the light of this theory it seems practical to incorporate means for observing actions directly into the (software-)tools which are used to carry out tasks. This marks a difference to traditional approaches to context-awareness where context is often observed from an outside perspective using some kind of sensors. It also marks a difference to user modeling, as this approach strives to be domain-independent and tries to avoid speculation on a user's motives.

We will therefore present a context model for activity context which holds observable information about actions and afterwards a middleware architecture which interacts with arbitrary software tools to gather information, to infer higher-level activity context information and to provide this for applications as a basis to adapt their behavior to the user's current activity.

| | Key/Value | Markup | OO | Dataflow | Relational | Logic | Ontology |
|---|---|---|---|---|---|---|---|
| (C1) Lightweightness | ++ | + | / | − | − | − | −− |
| (C2) Access Efficiency | / | − | / | / | ++ | / | / |
| (C3) Aggregation Support | −− | −− | ++ | ++ | / | + | + |
| (C4) Management of Relations | − | − | / | −− | ++ | + | / |
| (C5) Natural Projection | − | − | ++ | − | + | / | + |
| (C6) Data Distribution | − | − | − | ++ | / | − | + |
| (C7) Unambiguousness | − | ++ | − | − | − | − | ++ |
| (C8) Quality Compensation | −− | −− | / | + | − | − | / |
| (C9) Inference of Facts | −− | −− | −− | −− | −− | ++ | ++ |

Table 1: Evaluation of Context Models [42]

## 4 Context Models and Data Schemes

In literature the term *context model* is often used in different ways. Some authors understand a context model as the set of information, their representation as well as processing rules for a certain context-aware application or domain. For example, in an application which adapts to current weather conditions the fact that temperature is represented in degree Celsius, humidity as relative air humidity in percent as well as a rule that states at which thresholds the weather is considered muggy could make up a context model. Typically, authors who describe concrete context-aware systems [18] use the term in this way. Authors who seek to compare different approaches of modeling context [39, 16] use the term context model to describe the generic underlying data structures and available operations. These different meanings correspond to the notions of a *database schema* and *data model* in the terminology of database systems.

Throughout this paper we will use the term context model in the latter sense as we understand a context model as a set of methods and formal languages used to create, represent and interpret a context data schema. We will use this understanding to evaluate several existing context models based on a set of requirements that are presented in the following.

### 4.1 Evaluation of Context Models

A multitude of context models can be found in literature (cp. surveys by Chen/Kotz [6], Strang/Linnhoff-Popien [39] and Hartman/Austaller [16]). Thereby, different categories of models can be distinguished: key/value models, models based upon markup-languages, graphical models, object-oriented models, logical models, ontology-based models and dataflow models.

All of these classes make use of different methods and methodologies in order to model context. For the purpose of modeling activity context we conducted a requirements analysis based on a set of application scenarios as well as a literature survey in order to evaluate existing models. In the following, a set of criteria is presented that need to be considered when it comes to implementing context awareness on modern smartphones, providing a basis for the following evaluation of context models.

(C1) The infrastructure for realizing a model shall pose as few and low requirements as possible on soft- and hardware. (C2) Access to data shall be efficient. (C3) To ease access to high-level context data for developers, mechanisms for aggregating high-level data from lower-level data are vital. (C4) The model shall allow to easily manage relations between context entities. (C5) Real world concepts shall be naturally projected onto a context model, whereat the mapping shall be simple, direct, immediate and comprehensible. Although not used in the scenarios we addressed, future context-aware systems will likely be distributed. Therefore, (C6) support for distribution of context data is required. (C7) The context data shall be unambiguously interpretable even in heterogeneous systems. (C8) Mechanisms for dealing with incomplete, contradictory or uncertain data should also be considered. And finally, (C9) the ability to infer new facts from acquired data or preexisting world knowledge is desirable.

A summary of the strengths and weaknesses of different models can be seen in Table 1. Simple models like key/value- or markup models are lightweight and relatively efficient, but they lack support for more sophisticated demands. Whereas more complex approaches like logic-based models or ontologies offer good support for reasoning and data distribution at the price of being more heavyweight. As activities are relational in nature (cf. Section 3) and access to a potential large quantity of historical data is required a relational model was chosen to model activity context data. However, the context architectures discussed in Section 5 augment this model with some dataflow-oriented mechanisms. Especially when it comes to terms of distributing data and data sources as well as aggregating them this approach is useful to compensate weaknesses of the relational model.

## 4.2 A Generic Context Data Schema for Activity Context

Activities are the core of our data schema and our concept has to provide means for representing activities, entities and their relationships. Due to the fact that activities always occur in time and space and can form indirect relationships through spatial or temporal relations such data must be represented in the schema (including information about future events). As our approach should not be limited to a certain domain, activities and entities are modeled in a generic and abstract sense. This results in an open schema, extensible for all kinds of activities, entities, relations and application domains.

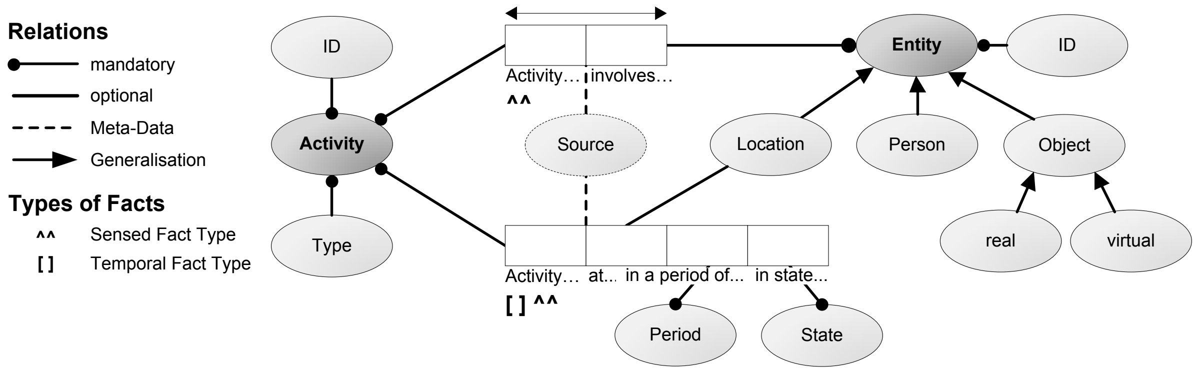

Figure 1: Core Components of the Context Data Schema

A graphical representation of the developed context data schema in the *Context Modeling Language* (CML) [18] can be seen in Figure 1. Activities form direct relations to entities and carry information about the place and time in which they took place. Places are also modeled as entities. Activities are enhanced with meta-data about how the data was gathered (as reported by software tools, measured by sensors, inferred from existing knowledge, other indirect means, or planned to be executed in the future), so the quality and reliability of information can be judged. This is especially useful for dealing with future activities as these are not certain to be carried out. Also, the lifecycle of an activity is modeled, whose abstract states are *ongoing*, *paused*, *successfully finished*, *unsuccessfully finished* or *finished with unknown result*. This information is crucial to infer the goals and needs of a user as a successful operation has other consequences for the goals of a user than an unsuccessful one. Now that we introduced the context data schema the next section will briefly survey existing middleware architectures for context awareness.

## 5 Architecture and Middleware for Context Awareness

Context data is relatively complex, the retrieval might be cumbersome and its management is not trivial. At the same time, a lot of applications could benefit from using context data and could even contribute context data. The latter is especially true for activity context. To address the difficulties in dealing with context data several approaches for developing middleware architectures for context awareness exist. The general goal of such software systems is to mediate context data between context sources and context consumers (i.e. applications). Typically, such systems are responsible for communicating with sensors, processing context data, supporting context-dependent decisions and for providing a communication infrastructure between the several components of a context-aware system [19].

By analyzing a set of application scenarios as well as existing literature, we found several requirements which apply to any middleware supporting context awareness for mobile devices. Ideally, the middleware should leverage a context model (R1) by being able to hold and manage instances of context data schemes. It should tolerate failures of components (R2), e.g. malfunction of sensors. A generic approach should be flexible and extensible (R3) and would benefit from a separation of basic facts and conclusions (R4) as this not only supports the flexibility, but also would make resources and inference processes more manageable. Compactness and suitability for limited hardware resources (R5), mechanisms for respecting privacy and security needs (R6) and for distributing the constituting components (R7) are other common desires.

Our perspective on activity context reveals additional requirements: In scenarios where activity context is used (cf. Section 2) context-aware applications not only consume context data, but the usage of software tools also directly provides context data. Therefore, the middleware should support activities (R8) by having direct means of reporting them. The more software tools use and provide activity context the greater the potential benefit for the user. As a consequence, such a middleware should be easy to use for application developers (R9). Looking at the scenario analysis presented in Section 2 and the context data schema developed in the previous section it becomes obvious that several traits of data and data structures need to be considered in the design of the middleware, such as: state and event-oriented data (R10.1), relational views (R10.2), time and sequence-oriented pattern analysis (R10.3) as well as hierarchical structures (R10.4). The support for temporal information should be taken into special consideration as there is a need to deal with events expected in the future (R11.1) as well as for recording time series to persist historical data (R11.2).

Besides the well known *Context Toolkit* [34], several other software architectures supporting context awareness have been presented (e.g. [41, 19, 24, 2]). These can be grouped into *service-oriented*, *agent-oriented* as well as *object-oriented* approaches [42] and are briefly surveyed in the following.

**Service-oriented Approaches** Service-oriented approaches aim at simplifying the use of hardware and network infrastructures for context-aware systems by providing a set of services. Typical services are communication services [27, 30, 31], services for dealing with geodata and positioning information [27, 30, 31] and services for local aggregation and storage of context data [27, 21, 31]. For systems which are meant to be run in a dynamic, fully distributed ubiquitous computing scenario some means for service discovery are usually provided [27, 31]. A lot of service-oriented approaches consider the challenges of mobile computing already in their design, several are targeted at smartphones which is why they often shine having a small footprint. The idea of building context-aware services by composing different services is somehow similar to the philosophy of the *Context Toolkit*. However, often these approaches are not based on an explicit context model and the different services lack unity. For instance, the *ContextPhone* platform is a service-oriented approach which provides services to monitor some aspects of user behavior (like messaging or telephony), but is not based on an explicit context model. To gather different kinds of context information, different services have to be queried [30].

**Agent-oriented Approaches** These kinds of approaches use autonomous, cooperative software agents to gather context information. Moreover, the agents are often able to exhibit context-adaptive behavior in order to achieve their goals. They often rely on well-defined context models based on ontologies, such as the *Web Ontology Language* [4], or custom extensions of them. Hence, most of these systems directly support reasoning and inference, but they do not necessarily support pattern detection and analysis. Examples for such approaches are the early work of Schilit [36], Spreitz and Theimer [38] or more recently the *Context Broker Architecture* by Chen et al. [8]. But, hence most existing systems are developed for ubiquitous computing rather than mobile computing they are rather heavy-weight.

**Object-oriented Approaches** In object-oriented approaches context entities are modeled as objects. A supporting middleware provides means for communication between objects, persisting and marshalling and most important observing them as well as notifying other components and entities upon any changes. Examples following this approach are the *Java Context Awareness Framework* [3] and the *DEMAC Context Service* [40]. Object-oriented approaches are very flexible and only make few restrictions. However, as they need to instantiate objects for all context entities they are not very efficient in large-scale scenarios. Moreover, they typically follow the principle of encapsulation and are therefore not well suited for reasoning about context entities, which involves some kind of data mining or pattern analysis. Finally, they can implement a context model, but are not necessarily based on one.

Two recent approaches to support context-aware systems deserve a more detailed explanation as some of their ideas influenced the design of our own middleware to a greater extend.

**PACE Middleware** The *Pervasive Autonomic Context-Aware Environments* (PACE) Project [19] developed a middleware to support context data schemes based on a relational context model, along with a formal graphical *Context Modeling Language* (CML) [18] to develop them. The middleware incorporates context repositories holding instances of such data schemes and provides means for managing and discovering them. Repositories support the logging of historical data if needed. Additionally, repositories can be distributed and the middleware manages communication by supporting querying as well as notification mechanisms using content-based message relaying between the loosely coupled components.

**Gaia OS** *Gaia OS* was developed as an operating system for context-aware ubiquitous computing applications [32]. It could be described as a service-oriented approach featuring five central services: (i) event management, (ii) context data storage, (iii) a service for detecting the presence of people, hardware, applications and other services, (iv) a service to access properties of such entities, and (v) a context file system. Unlike other service-oriented approaches it features an explicit, logic-based context model. In this model context is represented as four-digit predicate and this structure is closely linked to the mechanisms of the other services. How this works can be illustrated by the example of the *Context File System (CFS)* [20]. By storing a file in the directory `/location:/RM2401/situation:/meeting` it would be associated with the context "meeting in room RM2401". Listing the content of `/situation:/meeting` would list all files associated with meetings. Additionally, the special keyword `current` always expands to the current context. Gaia OS is a good example of leveraging a context model and integrating it with the mechanism of the framework while allowing transparent access by applications. Even applications not having implemented any kind of context awareness could benefit from the CFS and become context-aware (to some degree) without any further development work.

Table 2 gives an overview of our evaluation. Some approaches use activity context to some degree, but none supports generic activities. Furthermore, bidirectional communication between a context-aware

| | **Context Toolkit** | **Service-oriented** | **Agent-oriented** | **Object-oriented** | **PACE** | **Gaia OS** |
|---|---|---|---|---|---|---|
| (R1) Context model | key/value | – | ontology | (object-oriented) | relational | logic |
| (R2) Fault Tolerance | – | partial | partial | partial | partial | partial |
| (R3) Extensibility | $\surd$ | partial | $\surd$ | $\surd$ | $\surd$ | partial |
| (R4) Separation | $\surd$ | $\surd$ | $\surd$ | $\surd$ | $\surd$ | $\surd$ |
| (R5) Lightweight | partial | some | – | – | partial | partial |
| (R6) Privacy | – | some | $\surd$ | some | $\surd$ | – |
| (R7) Distribution | $\surd$ | $\surd$ | $\surd$ | $\surd$ | $\surd$ | partial |
| (R8) Activities | – | – | – | – | – | – |
| (R9) Ease of Use | partial | partial | partial | partial | partial | partial |
| (R10) Range of Data Aspects | partial | partial | – | – | – | – |
| (R11) Temporal Information | partial | partial | partial | partial | partial | – |

Table 2: Evaluation of Architectures for Context Awareness [42]

application and the enabling middleware in a way that applications not only consume context data, but also provide data is missing in general (R8). Support for temporal information is also generally limited, but some approaches at least provide support for recording and querying historical data (R11.1). Others feature ad-hoc mechanism to query calendar data, but do so without a well-defined model and generic support for future events (R11.2).

## 6 The Activity Awareness Architecture

As existing architectures of context-aware middleware systems do not meet all of the requirements to deal with activity context, we will present our *Activity Awareness Architecture* and describe some of the most important architectural and functional aspects as well as an example for the Android platform.

**Support for Activities** To support observing and reporting ongoing activities, the middleware is based on a minimalistic model of activities (cf. Section 4.2). The middleware offers a simple application programming interface (API) to create activities, report lifecycle state changes and assign related entities to them. Such API-calls will automatically trigger the creation and dispatching of events which will inform other endpoints of the system about the ongoing activity.

**Modularisation** The basic building blocks of our middleware are loosely coupled modules of two types: *EventSources* generating events and queryable *Repositories* with mixtures of them being possible. An example for an event source would be a sensor which monitors outgoing and incoming phone calls and reports them using the *Activity API*. Repositories are active data containers that are able to process and store data. Multiple repositories may coexist for different tasks or domains. They acquire data by listening for events or querying other sources and are also able to actively engage in event processing by conducting e.g. pattern analysis or infer high-level context from primitive events. To retrieve information afterwards repositories expose simple id-based interfaces for queries, offer path-based interfaces for hierarchical data or SQL-based interfaces for relational data. One example for such a repository is our central module which holds an instance of the context data schema (cf. Section 4.2). Another example is a repository which listens for ongoing activities and provides heuristic-based conclusions which entities are currently relevant for the user.

**Communication Infrastructure** The Activity Awareness Architecture uses an event-based publish/ subscribe mechanism as a basic communication infrastructure [1]. Modules and applications declare in which kind of events they are interested by specifying *Filters*. Filters can be based on event types, entity types, ids associated with events or a combination thereof. These filters are registered and managed by our middleware which uses them as a basis for content-based relaying of events. This structure allows for a loose coupling of modules and enables horizontal or vertical data fragmentation which is the basis for distribution later on.

**Entities and context-dependent, dynamic Naming Schemes** A special kind of query which repositories may support are context-aware dynamic path-based queries. Assuming that each entity can be identified by its type and its id, it is possible to use a simple path syntax like `/Person/id/5` to query for a certain person or `/Person/id/5/related/Place` to query all places related with the given person. Much like the *Context File System* (cf. Section 5) an entity or a list of entities can be addressed using such dynamic context-based path schemes. But unlike the CFS this mechanism is not limited to files, but can be used for all kinds of entities. The possibility to resolve such path-schemes by using multiple repositories and not being bound to a single central repository is an even more important difference.

**Support for Historical Data and Future Events** To deal with future events, like planned activities derived from calendar entries, our event system supports special kinds of events. *InstantEvents* are events taking place at the time they are dispatched and are exactly what traditional events are like. *ScheduledEvents* wrap other events and indicate that something is expected to happen in the future. As future events are not certain to happen they can hold additional meta-information to deal with this uncertainty. If a future event is canceled, for instance because a user deletes a calendar entry, this can be indicated by broadcasting a `CanceledEvent`. In order to query future as well as past events, special *TimelineRepositories* are used. Such repositories hold timelines for events which are managed on the basis of *InstantEvents*, *ScheduledEvents* and *CanceledEvents*. Other modules may not only query, but can also register themselves to be notified some time before future events occur.

## 6.1 Implementation of a Context-aware Calendar

We developed a basic implementation of the middleware and a context-aware calendar application for the *Android* platform to prove the feasibility. One of the main reasons for choosing Android, was its application framework, allowing to develop complex applications as a set of specialized, loosely coupled screens. As this concept is close to our understanding of activities it can be easily exploited to monitor the user's activities by injecting proxies between these modules which record what is going on. A similar mechanism can also be used to monitor phone calls.

The different middleware components that enable our context-aware calendar and the data flow between these are depicted in Figure 2. A call sensor monitors incoming and outgoing calls and reports them along with information about the participants using our Activity API. A location sensor monitors the user's whereabouts, tries to translate the position into symbolic location data and creates a `LocationEvent`. All this data is gathered in the *ActivityRepository* which implements the data schema for activity context (cf. Section 4.2). The ActivityRepository in turn is monitored by the *RelevanceRepository* which uses a simple heuristic to estimate which entities are relevant for the user in the near future. The heuristic is based upon the frequency of engaging entities, 1st- and 2nd-order relations between entities, and the temporal proximity of activities in which these entities take part. It is implemented using a bayesian network.

To further ease the use of the Activity Awareness Architecture we developed a generic GUI widget which suggests relevant entities depending on the current context. A path schema can be used as parameter to specify what kind of suggestion should be made. In our demo application we used it to suggest people and places for a calendar entry. Once a calendar entry is created our application will report this using the Activity API. A screenshot of the GUI and the GUI widget suggesting people to assign with the entry (red bounding box) can be seen in Figure 3.

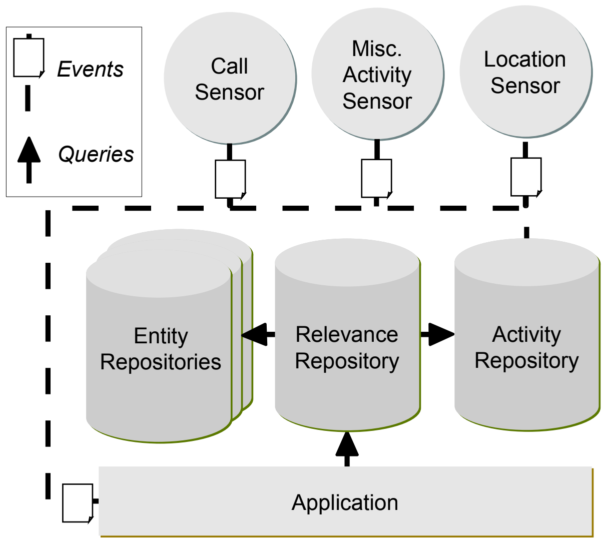

Figure 2: Middleware Modules and Data Flow

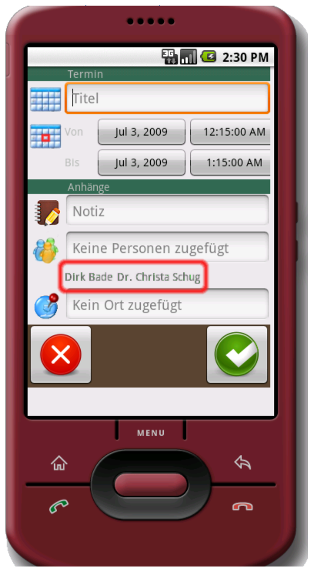

Figure 3: Context-based Calendar App

## 7 Conclusion

Activities are one of the most basic elements of context and represent goal-directed actions of a user. They are well suited to determine the relevance of context entities in a given situation by establishing relations between them which in turn can be used by applications to adapt their behavior to the user's current needs. In order to do so, a well-defined activity context model as well as a mediating middleware is required. In this paper, we therefore conducted an extensive requirements analysis, evaluated several existing context models and context awareness middleware architectures and concluded that none of the existing approaches is suited for dealing with activity context. Because of this, we presented a generic, domain-independent context data schema which is able to represent activities, the entities taking part as well as relations between these entities. Based upon this data schema, we introduced the *Activity Awareness Architecture*, a middleware for mediating activity context between context sources and context consumers and demonstrated the feasibility of our concepts by presenting an activity-aware calendar application for the Android platform.

As the idea of using activity context in mobile applications is relatively new our prospects for future work include a qualitative evaluation of our heuristic for suggesting related entities in a given context, an analysis in what way our middleware can help to conduct studies about user behavior, to what extent complex event processing techniques can be used to realize notifications about future-directed complex activities and how the activity context data can be distributed among several devices, thereby obeying privacy and security considerations.

In a future where ubiquitous computing will increasingly become reality more and more tools of our daily life will emerge into software-enhanced tools. Thus, the basis for sensing user activities will constantly grow, increasing the potential knowledge of activity context as well as the opportunities for making use of it. Therefore, activity context is a natural field for ubiquitous computing research.

## References

[1] Erwin Aitenbichler. Event-based and publish/subscribe communication. In Max Mühlhauser and Iryna Gurevych, editors, *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 152–171. Idea Group Publishing, December 2007.

[2] Matthias Baldauf and Schahram Dustdar. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4), 2004.

[3] Jakob E Bardram. The java context awareness framework (jcaf) - a service infrastructure and programming framework for context-aware applications. *in Pervasive Computing. Munich*, pages 98—115, 2005.

[4] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl reference. W3C Recommendation, February 2004. visited: 5.2.2009.

[5] Lars Braubach. *Architekturen und Methoden zur Entwicklung verteilter agentenorientierter Softwaresysteme*. PhD thesis, University of Hamburg, 2007.

[6] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, 2000.

[7] Harry Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, University of Maryland, Baltimore County, December 2004.

[8] Harry Chen, Tim Finin, and A. Joshi. Using owl in a pervasive computing broker. In *Proceedings of the Workshop on Ontologies in Agent Systems (OAS)*, July 2003.

[9] Keith Cheverst, Keith Mitchell, and Nigel Davies. Design of an object model for a context sensitive tourist guide. *Computers and Graphics*, Vol. 23:24—25, 1999.

[10] Constantinos K Coursaris and Dan J Kim. A qualitative review of empirical mobile usability studies. In *Proceedings of the Twelfth American Conference on Information Systems (AMCIS)*, 2006.

[11] Anind Dey and Gregory Abowd. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Hendheld and Ubiquitous Computing*, pages 304–307, London, UK, 1999. Springer-Verlag.

[12] Alan Dix, Tom Rodden, Nigel Davies, Jonathan Trevor, Adrian Friday, and Kevin Palfreyman. Exploiting space and location as a design framework for interactive mobile systems. *ACM Trans. Comput.-Hum. Interact.*, 7(3):285—321, 2000.

[13] Paul Dourish. What we talk about when we talk about context. *Personal and UbiComp*, 8(1):19–30, 2004.

[14] Gartner, Inc. . Market Share: Mobile Communication Devices by Region and Country, 2Q11. `http://www.gartner.com/it/page.jsp?id=1764714`, August 2011. visited: August 11th, 2011.

[15] Stephen Greene and Jason Finnegan. Usability of mobile devices and intelligently adapting to a user's needs. In *Proceedings of the 1st international symposium on Information and communication technologies*, pages 175–180, Dublin, Ireland, 2003. Trinity College Dublin.

[16] Melanie Hartmann and Gerhard Austaller. Context models and context awareness. In Max Mühlhauser and Iryna Gurevych, editors, *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 235–256. Idea Group Publishing, December 2007.

[17] Karen Henricksen. *A framework for context-aware pervasive computing applications*. PhD thesis, University of Queensland, September 2003.

[18] Karen Henricksen and Jadwiga Indulska. Developing context-aware pervasive computing applications: models and approach. *Journal of Pervasive and Mobile Computing*, 2(1):37–64, 2005.

[19] Karen Henricksen, Jadwiga Indulska, and Ted Mcfadden. Middleware for distributed context-aware systems. *International Symposium on Distributed Objects and Applications (DOA)*, 3760:846—863, 2005.

[20] Christopher K. Hess and Roy H. Campbell. A context file system for ubiquitous computing environments. Technical Report UIUCDCS-R-2002-2285 UILU-ENG-2002-1729, Department of Computer Science, University of illinois, Urbana, Illinois, July 2002.

[21] Thomas Hofer, Wieland Schwinger, Mario Pichler, Gerhard Leonhartsberger, Josef Altmann, and Werner Retschitzegger. Context-awareness on mobile devices - the hydrogen approach. In *Proc. of the 36th Annual Hawaii Int. Conf. on System Sciences (HICSS'03)*, page 292.1. IEEE Computer Society, 2003.

[22] Matthias Jöst. Adapting to the user. In Max Mühlhauser and Iryna Gurevych, editors, *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 282–295. Idea Group Publishing, December 2007.

[23] Manasawee Kaenampornpan and Bath Ba Ay. An integrated context model: Bringing activity to context. *Workshop on Advanced Context Modelling, Reasoning and Management (UbiComp2004)*, 2004.

[24] Kristian Ellebæk Kjær. A survey of context-aware middleware. In *Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering*, pages 148–155, Innsbruck, Austria, 2007. ACTA Press.

[25] Anders Kofod-Petersen and Jörg Cassens. Using activity theory to model context awareness. *Modeling and Retrieval of Context: Second International Workshop, MRC 2005, Revised Selected Papers. Volume 3946 of LNCS (LNAI)*, pages 1—17, 2006.

[26] Kari Kuutti. Activity theory as a potential framework for human-computer interaction research. In *Context and Consciousness: Activity Theory and Human-Computer Interaction*, pages 17–44. MIT Press, 1996.

[27] Hui Lei, Daby M. Sow, I. I. John S. Davis, Guruduth Banavar, and Maria R. Ebling. The design and applications of a context service. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(4):45–55, 2002.

[28] Antti Oulasvirta, Mika Raento, and Sauli Tiitta. Contextcontacts: re-designing smartphone's contact book to support mobile awareness and collaboration. In *IEEE Photon. Technol. Lett*, page 167—174. Portal, 2005.

[29] Thai-Lai Pham, Georg Schneider, Stuart Goose, and Arturo Pizano. Composite device computing environment: A framework for situated interaction using small screen devices. *Personal UbiComp*, 5:25—28, 2001.

[30] Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. Contextphone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51—59, 2005.

[31] Jukka Riekki, Davidyuk, Oleg, Forstadius, Jari, Sun, Junzhao, and Sauvola, Jaakko. Enabling context-aware services for mobile users. In *Proceedings of IADIS Virtual Multi Conference on Computer Science and Information Systems*, pages 360–369, April 2005.

[32] Manuel Román, Christopher Hess, Renato Cerqueira, Roy H Campbell, and Klara Nahrstedt. Gaia: A middleware infrastructure to enable active spaces. *IEEE Pervasive Computing*, 1:74—83, 2002.

[33] Caspar Ryan and Atish Gonsalves. The effect of context and application type on mobile usability: an empirical study. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, pages 115–124, Newcastle, Australia, 2005. Australian Computer Society, Inc.

[34] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. 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*, pages 434–441, Pittsburgh, Pennsylvania, United States, 1999. ACM.

[35] Bill N Schilit, Norman Adams, and Roy Want. Context-aware computing applications. *In Proceedings of the Workshop on Mobile Computing Systems and Applications*, pages 85—90, 1994.

[36] William Noah Schilit. *A System Architecture for Context-aware Mobile Computing*. PhD thesis, Columbia University, 1995.

[37] Albrecht Schmidt. Implicit human computer interaction through context. *Personal and Ubiquitous Computing*, 4(2):191–199, June 2000.

[38] Mike Spreitzer and Marvin Theimer. Providing location information in a ubiquitous computing environment (panel session). *SIGOPS Oper. Syst. Rev.*, 27(5):270–283, 1993.

[39] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[40] Mirwais Turjalei. *Integration von Context-Awareness in eine Middleware für mobile Systeme*. Diplomarbeit, University of Hamburg, June 2006.

[41] Terry Winograd. Architectures for context. *Hum.-Comput. Interact.*, 16(2):401–419, 2001.

[42] Jan D. S. Wischweh. Aktivitätsorientierte Kontextadaption in mobilen Anwendungen. Master's thesis, University of Hamburg, July 2009.

[43] Andreas Zimmermann, Andreas Lorenz, and Reinhard Oppermann. An operational definition of context. In *Modeling and Using Context*, volume 4635 of *Lecture Notes in Artificial Intelligence*, pages 558–571. 2007.