Context-aware Apps with the Zonezz Platform

Jörg Roth
Univ. of Appl. Sciences Nuremberg
Kesslerplatz 12
D 90489 Nuremberg, Germany
+49 911-5880 1169
Joerg.Roth@Ohm-hochschule.de

ABSTRACT
Current smart phones can easily detect the geographic location, but very often applications are more interested in the *meaning* of the location for the user. In this paper we present the *Zonezz* platform that identifies meaningful locations such as 'home' or 'work'. It provides an easy to understand context model and fully runs on a mobile device without the need for a central service. Other apps can use this platform to create context-awareness. We show the benefits of this platform with the help of a context-aware calendar tool.

Categories and Subject Descriptors
D.2.2 [**Design Tools and Techniques**]*: Software Libraries*

General Terms
Algorithms, Design, Human Factors

Keywords
Context-awareness, location-awareness, mobile application

1. INTRODUCTION
The idea of context-awareness is nearly as old as mobile devices. If a mobile application knows about the current context (especially location and time), it can provide more specific information that is suitable for the current situation. This is especially important as mobile devices have certain limitations that affect the usability for mobile users.

Even though the theory of contexts is investigated in-depth in the last decade, the corresponding concepts often did not widely reach end-users. This was because for a long time typical end-user devices were not able to capture required context information. This changed when smart phones such as the iPhone or Android phones get more and more available. These devices usually are able to capture context information, especially related to positioning. In addition, they provide powerful development environments. With market platforms we can reach a broader user community without remarkable deployment costs.

A device can get the user's current geographic location with the help of GPS or fingerprinting. But as a major problem, an application cannot detect the *meaning* of the current location for the user. Typical contexts related to locations are, e.g., 'home', 'work', or 'shopping', but these locations are different for different users.

In this paper, we present the *Zonezz* platform that provides a mapping of physical positions to context-related locations. Zonezz is fully implemented for the Android platform. Mobile applications (in the following called *apps*) can use this platform to react on contexts and context changes. Some examples:

- A calendar app may show business entries more obtrusively when the user stays in the company. Or, it may hide a shopping list until the user enters a shopping site.
- A context-based reminder could trigger an alarm, if the user enters a special location. It could e.g. remind: 'check the heating' not until the user comes home.
- Mobile devices could switch between communication channels at different sites and can, e.g., use the business SIM card and business email account at work.
- In social media, check-in services could inform friends, when a user is at home.
- Mobile devices could switch between device home screens, e.g. one that only shows business apps at work, and one that includes games and MP3s at home.

It is reasonable to shift administration and detection of contexts to a central platform on a mobile device rather than separately inside each app. Other apps can use such a platform with the help of powerful communication and triggering mechanisms between software components that are available in modern smart phone operating systems.

2. RELATED WORK
Many projects in the area of context- or location-awareness focus on positioning issues (e.g. [12]). Indoors it still is a problem to get a precise position that identifies, e.g. a special room. But for larger scales current smart phones provide easy to use mechanisms that even work indoors when GPS fails. Therefore, many recent publications consider positioning as available building block and either focus on the processing of positions or investigate new sensors that provide additional information. In [11], e.g., apps can distinguish types of movement (such as 'walking', 'car' and 'train') with the help of the accelerator sensor that is available in most smart phones. The movement type can be viewed as additional information to define a context.

Other projects try to identify symbolic (or often called *semantic*) locations from physical positions. In [18] three major contexts are

MobiHeld'11, Oct. 23, 2011, Cascais, Portugal.

distinguished: 'home', 'work' and 'on the move'. [1] presents an approach to geometrically cluster GPS locations of multiple users to identify interesting locations. In [2] visited locations are stored in a kind of diary. This can by classified later by a Bayesian network that also takes into account the time of visit. A similar approach follows [10], but it first maps physical locations to semantic locations with the help of a central geo database. Also indoor locations are considered: in [9] small indoor regions, so called *activity zones* (e.g. the 'lunge chair') are classified. Activity zones may have complex geometric shapes and trigger context rules that can perform further activities.

Further approaches concentrate on the usage of context information for certain applications. [5] takes into account the current position to optimize queries to public transportation schedules. The benefit of contexts for mobile collaboration is explored in [6]. [8] uses the location as means for control access to certain resources.

Projects such as *CybreMinder* [4], *Nexus* [7] or [17] provide a general platform to capture and distribute context information. Usually they base on a centralized infrastructure that collects information and executes the rules to trigger events. An exception is *ContextPhone* [13] that primarily provides a system to plug-in context components on a mobile device. If a platform relies on central services, privacy issues become more important.

Even though conceptual interesting, some of the approaches above provide too many degrees of freedom to define contexts and rules. A typical user wants to understand such a system in some degree. If we want to avoid security issues, it is reasonable to keep the entire computation on the mobile device.

3. THE ZONEZZ PLATFORM

3.1 Goals and Requirements

Zonezz is designed according to the following goals:

- The entire context computation should be performed on the mobile device. No central service or central database should be required.
- The concept should be understandable and manageable by typical users. Too complex context configurations or rules should be avoided.
- The user should be strongly supported by the platform to set up the configuration, but should be able to overwrite suggestions if desired.

We do not want to distinguish fine-grained contexts (such as 'meeting in room xy') that require a very precise indoor positioning system. Typically we want to distinguish 2 to 4 contexts of which 'work' and 'home' usually are the most important.

Our approach is not designed for security related applications, e.g. for access control or payment. Position measurements can easily be manipulated inside the phone and the resulting contexts are thus not verifiable. Even tough there exist complex mechanisms to generate manipulation-secure positions [3], they are not incorporated into our platform.

All context-relevant data are stored on the mobile device. As the whole computation is also performed on the smart phone, we have not to deal with privacy issues related to the network or central services. No information from our platform is sent via a wireless network. But, if attackers get access to the mobile device itself (e.g. steal it or get a root login), they may be able read private context-related data of our platform. These attacks, however, would be a serious problem for nearly all mobile applications.

3.2 Modeling Zones

Our model is influenced by former research (especially [14]) but is strongly simplified as the entire execution should take place on the mobile device.

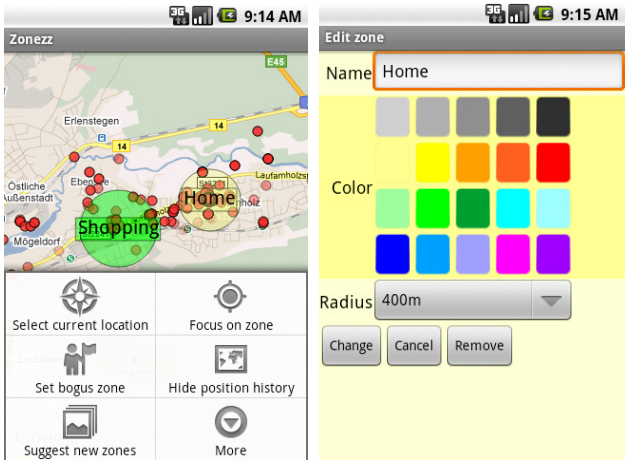

Figure 1. Definition of zones

We use a simple geometric model to define different contexts called *zones*. Zones are two-dimensional circular regions on the Earth's surface with arbitrary center and radius. The model is easy to understand by the mobile user and allows the execution of efficient algorithms. To define a new zone, a user points on a map and specifies name, color and radius (fig. 1). The color is only used for presentation purposes. For the radius, there exist discrete values 50m, 100m, 200m,…, 50km.

One could argue that circles usually do not precisely represent context borders. But:

- More complex areas such as polygons are difficult to enter and to administrate on a mobile device.
- For circular areas we can realize efficient algorithms to suggest new zones (section 3.4).
- Position measurements are Gaussian distributed around a real position. Areas that include all position measurements of a certain area tend to circles anyway, even though the original areas are not circles. Thus, the actual benefit for more complex geometries would be small.

Every time a measured position is inside such a circle, the corresponding zone is considered as part of the current context. As zones may overlap, the context is defined by a set of *hit zones*. We expect the hit zone with the smallest radius as more specific for the context, thus the set of hit zones is ordered ascending by their radius.

3.3 The Architecture

Fig. 2 shows the Zonezz architecture. The platform contains the *Zonezz App* and *Core Services* with a total size of 71 kB. The app is mainly used to set up the zone database (*Zone DB*), which is modeled with SQLite and contains the table of zones. New zones can be suggested by a *Suggestion System*. The Core Services allow two types of usage:

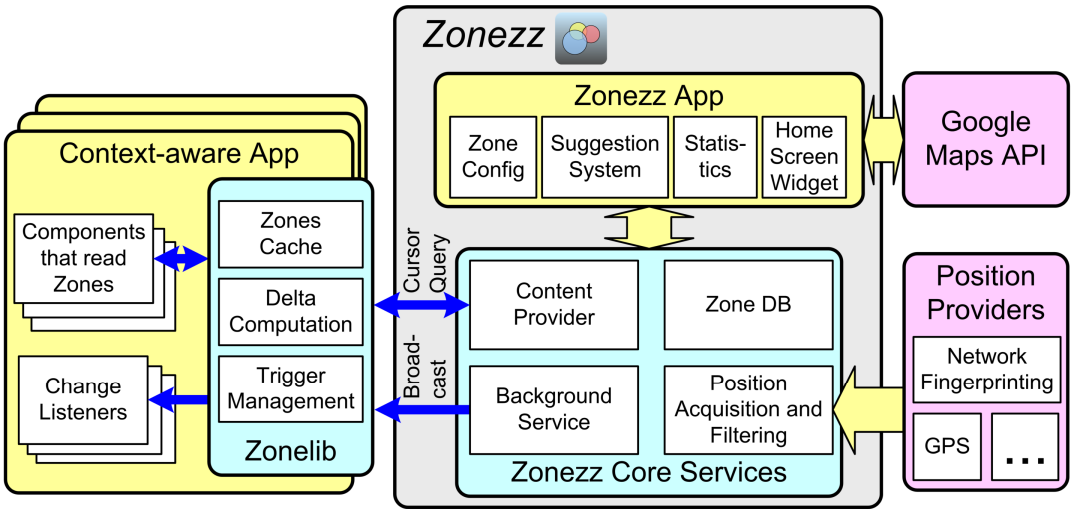

Figure 2. The Zonezz architecture

- Other apps can use the platform to query all defined zones and the ordered list of hit zones that represent the context. For this usage, Zonezz provides a query interface similar to databases.
- Other apps can asynchronously listen to changes of hit zones. I.e., they receive an event whenever the user enters or leaves a zone.

Not every app must use both types.

The Android structure to query data across apps is the *Content Provider*. Similar to an SQLite database the Zonezz content provider supports cursor-based queries on the tables *all zones*, *hit zone* and *state* (see below).

The Zonezz *Background Service* permanently computes the current set of hit zones. For every change it sends a system-wide broadcast using Android's *Broadcast Receiver* model. Any app may register for these events to asynchronously react to changes.

Background services on mobile phones continuously run, even when the device is in stand-by mode. Thus, it is important that the service does not consume too much CPU power as other apps may be affected and battery will be drained. This is an additional argument for the simple zone model. In the current version, the background service wakes up every 2 minutes to check if the location moved to other zones. Between the checks, the phone can switch to low power mode. The check as such does not significantly affect battery lifetime. Note that on typical smart phones a lot of background services periodically work.

To compute the list of hit zones, the recent position is required. Current Android devices offer two providers: GPS satellite navigation and a 'network' provider that is based on signal strength fingerprinting. As a main decision, a user has to specify, if GPS should be turned on (causing higher battery consumption). The core component *Position Acquisition and Filtering* tries to fulfill the user's and system's requirements according precision, age of measurements and battery consumption. It balances these factors with the help of a set of pre-defined rules.

An empty hit zone list can be result of two effects: either no position measurement is available or a position is available but outside any zone. An application maybe has to distinguish these cases. Thus, the Zonezz content provider and broadcasting distribute a *state* (table 1).

Table 1. Service states

State	Description
`RUNNING`	Everything OK and running.
`OLD POSITION`	The last position fix is too old. The app may use the list of hit zones but should be aware that it may be outdated.
`NO POSITION`	No position is available. The hit zone list is thus empty.
`NO PROVIDER`	The user deactivates all positioning mechanisms or the device has no such component.
`NOT RUNNING`	The Zonezz background service is not running anymore – this should not happen (and never happened so far).

To simplify the access of other apps to the Zonezz services, they can use the *Zonelib*. This library shields the communication, provides caching and manages the registration of listeners. Zonelib has a very small size of only 7 kB (jar file) and can simply be included into any Android app project. Table 2 shows the most important library calls.

Table 2. Important Zonelib calls

Main methods
`ArrayList<Zone> getAllZones()`
`ArrayList<Zone> getHitZones()`
`int getState()`
`void addZoneListener(ZoneListener listener)`
ZoneListener methods
`void allZonesChanged(ArrayList<Zone> zones)`
`void hitZonesChanged(int state, ArrayList<Zone> zones, ArrayList<Zone> addedZones, ArrayList<Zone> removedZones)`

3.4 Suggesting Zones

Zonezz contains a *Suggestion System* that makes suggestions for new zones based on the position history. The idea is to identify clusters of measurements that are close together. If the user wants to use the Suggestion System, she or he agrees that the Zonezz platform logs visited locations for a certain time.

An ideal Suggestion System would compute new zones without the help of the user. However, we have to face two problems:

- Two effects are not distinguishable by a fully automatic system. First: a position may deviate due to measurements errors (e.g. some 100 meters for the 'network' provider); second, a user may really move away from a zone center. Thus, it often is not clear if measurements belong to a location.
- If multiple frequently visited locations are close together, it is not clear, if this is one zone or multiple zones. This is because contexts are not only geometrically defined but have a certain meaning.

We look at the example in fig. 3: an appropriate zone may be shown on the top/left, if we assume that the zone only covers the building below. But if all three buildings belong to the same company, the zone in the top/right could be more appropriate.

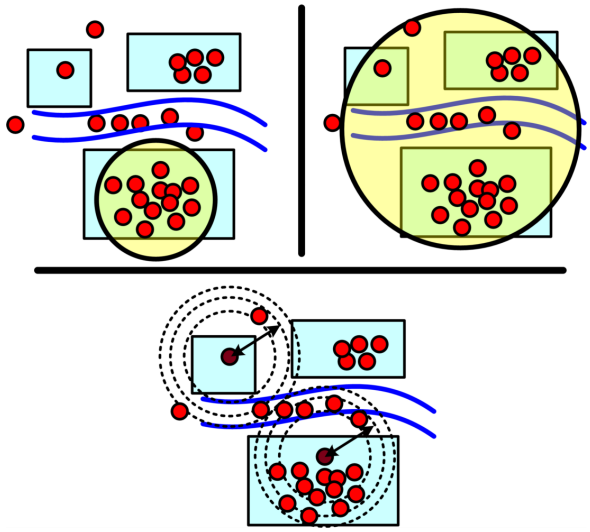

Figure 3. Suggesting zones

Therefore, we need user cooperation. Before the system identifies zones, the user defines

- the time range of position measurements that should be analyzed (e.g. last day, last week), and
- the minimum time a user has to reside at a location to consider it as a new zone (e.g. 4 hours/day).

A user who, e.g., requests a suggestion for a new zone 'work' could demand a minimum of 7 hours per day in the last three days.

The idea of the suggestion algorithm is as follows (fig. 3 bottom):

- Each measurement in the specified time range is considered as center of a potential zone. We iterate through all measurements and all possible radius steps. Note that our radiuses have discrete values. For every iteration we count the number of other measurements that are inside the circle.
- We filter out iterations that do not fulfill the minimal residence time defined by the user.
- For each iteration we compute a rank that reflects the number of measurements (more is better) and the radius (smaller is better). We filter out zones below a certain rank.
- Remaining iterations are mapped to zone suggestions. As suggested center we use the mean of measurements inside the circle and *not* the measurement that originally defined the circle.
- We order the remaining zones by their rank (best first). Each zone inside the list is presented on a map. The user is asked one after the other, whether the suggestion should be stored as new zone or whether the next suggestion should be presented (fig. 4).

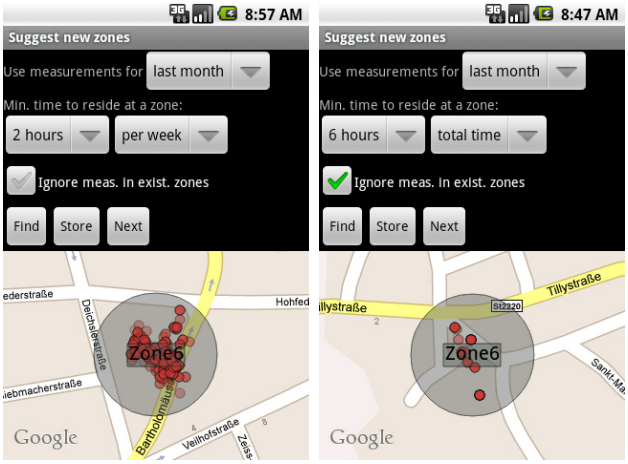

Figure 4. Screens of the Suggestion System

We could use very different formulas to rank the potential zones. We made the best experiences with

$$rank = \frac{m}{\sqrt{r}}$$

for m measurements inside the circle of radius r.

Some words about performance. Zonezz logs a position measurement every 6 minutes. If a suggestion should base on one month, this means to check 7200 records. For n records a plain implementation requires $O(n^2)$ steps to compute all ranks. If we used spatial indexing mechanism (such as [16]), we can reduce the number of checks to $O(n \log n)$. Zonezz requires 21 seconds to create suggestions based on one month of measurements on a Milestone device (550 MHz).

3.5 Further Functions

The Zonezz platform offers some further functions:

- The user can put a widget onto the Android home screen (fig. 5 left). This is useful to view the current zone and positioning state.

Figure 5. Additional Zonezz functions

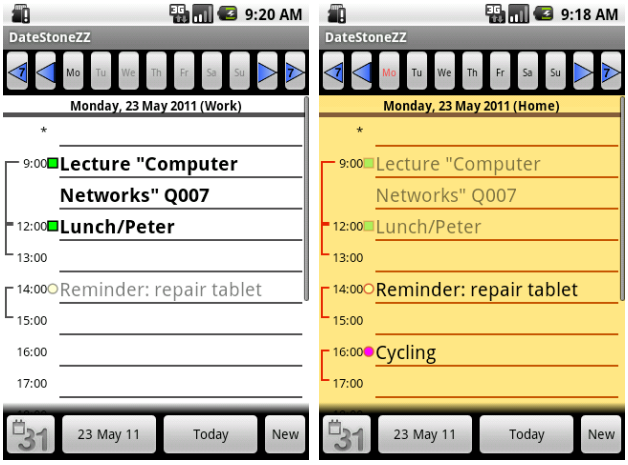

Figure 6. Different presentations dependent on the current zone

- A position statistics (fig. 5 right) provides information, how long the user resides at which zone.
- The user can set a 'bogus zone' for a certain time. This means, the measured positions are ignored and a pre-selected zone is returned as hit zone.

Bogus zones can be useful if, e.g., real positioning completely fails in a certain situation, or if the user wants to simulate a certain context.

4. THE CALENDAR CASE

DateStone Calendar [15] is a typical calendar app similar to many other apps that allow a user to organize daily dates and tasks. Our goal was to make it context-aware. We identified three mechanisms that can be affected by the current context:

- The color style is changed according to the current zone. There can be e.g., a different 'work' and 'home' color style. The look and feel of app usage is heavily influenced by colors, thus it is useful to have different color styles for different contexts. There exist predefined styles such as 'black/white', 'water', 'summer' that define all screen colors.
- Date entries that are more related to the current context are displayed more obtrusively (e.g. bold) whereas other entries are presented unobtrusive or even hidden.
- Date alarms may be delayed until the user enters a certain zone. E.g. an alarm to remind something special to buy may be delayed until the user enters the 'shopping' zone.

Typical calendar apps already provide the concept of user-defined categories. For each date entry a user can define a category such as 'job', 'to buy' or 'hobby'. In non-context-aware calendars, these categories are primarily used as display filter and affect some colors in overviews.

As a great benefit, categories are already strongly related to contexts. Thus, we can easily map these categories to our zones. A user can express the importance of a category for a certain zone with the help of text styles such as *hidden*, *transparent*, or *bold*. Fig. 6 shows the same day sheet displayed in different zones.

In the 'work' zone (left), the work entries are bold, others are transparent or even hidden. In the 'home' zone (right), home entries use a standard font, whereas work entries are transparent. The user can configure the presentation of entries with the help of rules, e.g.

```
if inside Work:
    Color Style: Black/White
    Category Job: Bold
    Category Home: Transparent
    Category Hobby: Hide
    Category Shopping: Show
```

A zone rule is 'fired' if the user resides *inside* or *outside* a certain zone. The ordering of rules is important as the first fired rule defines the presentation. The user can configure the rules with the help of a convenient graphical front-end that also checks the rule integrity.

Finally, we can define for each category, whether reminder alarms should be activated at the given time, or be delayed until the user enters a certain other zone. With these rules, reminder alerts are not triggered in the wrong context what may distract the user.

The resulting flow of data is presented in fig. 7. The major challenge to extend the original calendar app was to develop the *Zone Rule Editor* and *Zone Rule Execution* (only few hundreds line of code), whereas the major context functions are provided by Zonezz and accessed through the Zonelib.

5. CONCLUSIONS

The Zonezz platform allows other apps to query for the current location context or can listen to context changes. A context-aware calendar app shows how an app could react on different contexts.

In the future we want to additionally integrate movement contexts into the concept. Besides our zones, contexts such as 'train-driving', 'walking' etc. could be derived from the accelerator sensor and further define the user's current situation.

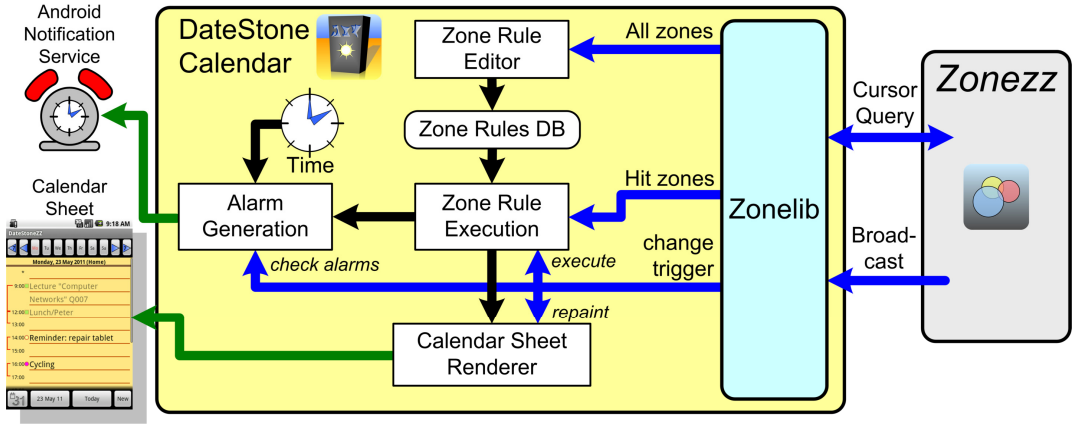

Figure 7. Data flow in the context-aware calendar

6. REFERENCES

[1] Ashbrook D., Starner T. 2003. Using GPS to Learn Significant Locations and Predict Movement across Multiple Users. Personal and Ubiquitous Computing archive Vol. 7 No. 5, Oct. 2003

[2] Bicocchi N., Castelli G., Mamei M., Rosi A., Zambonelli F. 2008. Supporting location-aware services for mobile users with the whereabouts diary. 1st intern. conf. on Mobile Wireless Middleware, Operating Systems, and Applications

[3] Decker M. 2009. Prevention of Location-Spoofing – A Survey on Different Methods to Prevent the Manipulation of Locating-Technologies. Proc. of intern. conf. on e-Business (ICE-B 09), Milan, Italy, July 7-10 2009, 109-114

[4] Dey A., Abowd G. 2000. CybreMinder: A Context-Aware System for Supporting Reminders, Proc. of the Handheld and Ubiquitous Computing 2000, Bristol, UK, Springer LNCS 1927, 172-186

[5] Ferris B., Watkins K., Borning A. 2010. OneBusAway: Location-aware tools for improving public transit usability. IEEE Pervasive Computing, January-March 2010, Vol. 9 No. 1, 13-19

[6] Häkkilä, J.; Mäntyjärvi, J.2005. Collaboration in Context-Aware Mobile Phone Applications. Proc. of the 38th Annual Hawaii intern. conf. on System Sciences, 2005. HICSS '05

[7] Häussermann K., Hubig C., Levi P., Leymann F., Simoneit O., Wieland M., Zweigle O. 2010. Understanding and designing situation-aware mobile and ubiquitous computing systems. Proc. of intern. Conf. on Mobile, Ubiquitous and Pervasive Computing, March 2010, 329-339

[8] Jafarian J., Amini M. 2009. CAMAC: A Context-Aware Mandatory Access Control Model. ISC Intern. Journal of Information Security January 2009, Vol. 1, No. 1, 35-54

[9] Koile K., Tollmar K., Demirdjian D., Shrobe H., Darrell T. 2003. Activity Zones for Context-Aware Computing. UbiComp 2003: Ubiquitous Computing, Springer LNCS 2003, Vol. 2864/2003, 90-106

[10] Liu J., Wolfson O., Yin H. 2006. Extracting Semantic Location from Outdoor Positioning Systems. 7th intern. conf. on Mobile Data Management

[11] Nick T., Coersmeier E., Geldmacher J., Götze J. 2010. Classifying Means of Transportation Using Mobile Sensor Data. IEEE World Congress on Comp. Intelligence, Barcelona, Spain, July 2010

[12] Papliatseyeu A., Mayora O. 2009. Mobile Habits: Inferring and Predicting User Activities with a Location-Aware Smartphone. 3rd symp. of Ubiquitous Computing and Ambient Intelligence 2008, Advances in Soft Computing, 2009, Vol. 51, 343-352

[13] Raento M., Oulasvirta A., Petit R., Toivonen H. 2005. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. IEEE Pervasive Computing, April-June 2005, Vol. 4 No. 2, 51-59

[14] Roth, J. 2008. A Probabilistic Approach for Context Reasoning. Use In Context, GI Informatik 2008, Munich, Germany, Sept. 12. 2008, Proceedings 134, Vol. 2, 802-807

[15] Roth, J. 2011. DateStone Calendar Manual. wireless-earth, http://android.wireless-earth.org/datestone.html

[16] Roth, J. 2011. Moving Geo Databases to Smart Phones – An Approach for Offline Location-based Applications. Innovative Internet Computing Systems (I2CS), Berlin Germany, June 15-17 2011

[17] van Sinderen, M. J., van Halteren, A. T., Wegdam, M., Meeuwissen, H. B., Eertink, E. H. 2006. Supporting context-aware mobile applications: an infrastructure approach. IEEE Communications Magazine, Sept. 2006, Vol. 44, No. 9, 96-104

[18] Verkasalo H. 2009. Contextual patterns in mobile service usage. Personal and Ubiquitous Computing. Vol. 13, No. 5, 331-342