CAPIM: A Platform for Context-Aware Computing

Ciprian Dobre*
*University POLITEHNICA of Bucharest, Romania
E-mail: ciprian.dobre@cs.pub.ro

Abstract

Today smartphones integrate many sensors and provide large computing capacities. They enable the shift towards massive quantities of real-time information becoming access push rather than demand pull on a global case. In this we describe CAPIM, a platform to support such a paradigm. It integrates services to monitor and a context for adapt with the user's context using sensors and capabilities of smartphones, together with online social data. It integrates context-aware services that are dynamically configurable and use the user's location, identity, preferences, profile, and relations with individuals, as well as capabilities of the mobile devices to manifest themselves in many different ways and re-invent themselves over and over again. Such services aggregate and semantically organize the context data. They react based on dynamically defined context-oriented workflows, and the platform includes an execution engine that supports context-aware actions for orientation, information, and recommendation. We describe the architectural decisions of our platform, together with implementations details, and present several case study scenarios which show its potential to handle a variety of context-aware situations.

Key Words: context-awareness, mobile computing, smartphones, pervasive adaptation and computation carried by users, social adaptation of services.

1. Introduction

Today smartphones are becoming commodity hardware. They are seen everywhere, as more people realize that having more sensing and computing capabilities in every-day situations is attractive for many reasons. Smartphones are in fact already used to optimize (e.g. by helping organizing tasks, contacts, etc.) and assist (e.g. with navigation, find information more quickly, access online data, etc.) users with their everyday activities. Their success is also the basis for a shift towards developing mobile applications that are capable to recognize and pro-actively react to user's own environment. Such context-aware mobile applications can help people better interact between themselves and with their surrounding environments. This is the basis for a paradigm where the context is actively used by applications designed to take smarter and automated decisions: mute the phone when user is in meeting, show relevant information for the user's current location, assist the user find its way around a foreign city, or automatically recommend events based on the user's (possible learned) profile and interests.

CAPIM (Context-Aware Platform using Integrated Mobile services) is a platform designed to support the construction of such next-generation applications. It integrates services designed to collect context data (location, user's profile and characteristics, as well as the environment). These smart services are dynamically loaded by mobile clients, and make use of the sensing capabilities provided by modern smartphones, possibly augmented with external sensors. The data is collected and aggregated into context instance. This is also possible augmented with external and inferred data about possible situations, relations, or other events.

In addition, the platform includes a workflow engine designed to continuously evaluate the context and take automatically decisions or actions based on customized rules and particular context events. We describe how such rules are constructed further within this paper. We also present CAPIM's visualization layer that allows intuitive and easy interaction for the user with the platform and its running services, but also with the user's own environment.

The rest of the paper is structured as follows. Section 2 presents related work. In Section 3 we present the architecture of the proposed context-aware platform, and Section 4 presents the context model. Section 5 presents several evaluation scenarios. In Section 6 we conclude and present future work.

2. Related Work

Several platforms for pervasive and context-aware systems to support rich contextual features were built in the past few years. Several of these systems are openly available. However, no such systems are available for devices that are inexpensive, available off-the-shelf, and widely accepted by users in their everyday life, such as smart phones, the first real-world pervasive platform.

MobiPADS [3] is a middleware for mobile environments. It consists of Mobilets, entities providing particular services that are able to migrate between MobiPADS environments. Each Mobilet consists of a

slave and a master. The slave resides on a server, while the master resides on a mobile device. Each pair cooperates to provide a specific service. MobiPADS is concerned with internal context of the mobile devices, which is used to adapt to changes in the computational environment. Thus, context types include: processing power, memory, storage, network devices, battery etc.

CARISMA [4] provides adaptable services for different applications. Each application has passive and active profiles. The passive one define actions the middleware should take when specific context events occurs, such as shutting down if battery is low. The active information defines relations between services used by the application and the policies that should be applied to deliver those services. Different environmental conditions may also be specified, which determine how a service is delivered.

CARMEN [5] transparently handles resources in wireless settings assuming temporary disconnects. If a user moves to another environment the proxy also migrates using wired connections. Each mobile user has a proxy which provides access to resources needed by the user. When migrating, the proxy makes sure that resources are also available in the new environment. This can happen by: moving the resources with the agent, copying the resources, using remote references, or re-binding to new resources which provide similar services.

These and similar other middlewares support differently pervasive and mobile computing based on context information. They all provide some method of adapting to changes in the context, and methods for collecting context, but otherwise use different entities and have different focus. We present a more complete and complex context model that integrates a wider spectrum of information, ranging from location to user's profile and social capabilities. The middleware allows collecting context information from a wide-area of data sources, aggregation including providing semantic relations, and an engine that is able to mimic the behavior of various context-aware applications.

3. Architecture Design

CAPIM's architecture consists of several vertical layers (see Figure 1). Each layer provides a specific function: 1) collecting context information, 2) storing and aggregation of context information, 3) construction of context-aware execution rules, and 4) visualization and user interaction. All layers are composed of several components, making the infrastructure suitable for experimenting with a wide range of context-aware methods, techniques, algorithms or technologies. It can be used to construct context-aware applications using a service-oriented composition approach: load the core container, instruct it to load the necessary context-gathering services, deploy a corresponding context-aware business workflow and call the actions to be executed when context is met. Several such scenarios are presented in the next Section.

First the user installs on his/her smartphone the platform container. It is the execution root framework on which all layers are built. For example, the monitoring services are dynamically discovered, downloaded as needed, loaded and executed inside this container. So, for collecting context information, the **first layer** includes sets of monitoring services (collecting and first-stage storing on the local mobile device) for the context data.

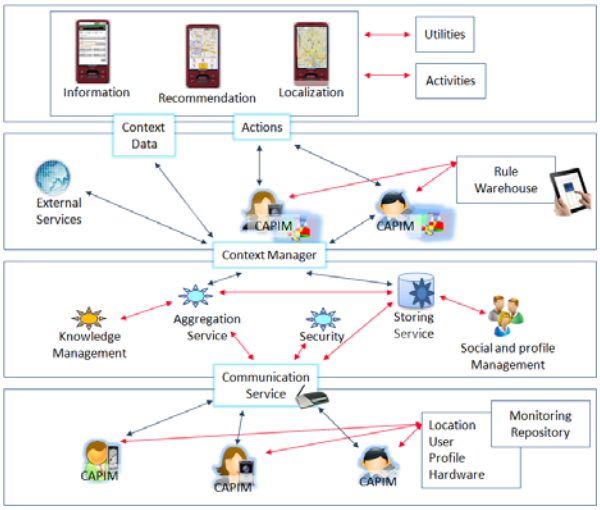

Fig. 1. The proposed architecture.

Each monitoring service is packed in a digitally signed monitoring module. These modules are downloadable from remote repositories, resembling application stores. The monitoring services can be developed/maintained by third party organizations (for example, a manufacturer might construct a module to collect data from its own sensor, therefore integrating its data within the user's context).

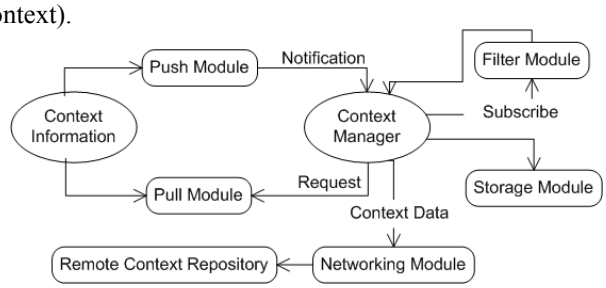

Fig. 2. Flow of monitoring information.

Each monitoring service is executed inside a separate container. This allows separation of concerns (no service needs to know what other modules are deployed) and fault isolation.

The monitoring flow is under the control of a Context Manager (Figure 2), orchestrating the flow of information between the monitoring services. Depending on the

function supported, the monitoring services are grouped in several categories. The Push and Pull monitoring services are directly responsible for collecting context information. They collect context information generally directly from sensors. The Push service reacts to changes of the context, which in turn triggers notifications to be sent to the Context Manager. The Pull service is periodically or on-request interrogated for the current values of the monitoring parameters.

The context information is sent to Filter, Storage and Networking services. The Filter service subscribes to specific context information. The Context Manager forwards the data of interest to the Filter service, which in turns can produce new context information (possible from multiple data sources). Such a construction allows for first-stage aggregation of context information.

The Storage service can store data locally for better serving the context-execution rules. Finally, the Networking service is responsible for sending the collected context information remotely to aggregation services (the Remote Context Repository component located in the next layer). Here we can experiment with different network protocols and methods of sending data, whilst balancing between costs and energy-consumption.

Each monitoring service is also responsible for a particular type of monitoring information. Thus, these services fall into different categories: location, user, profile, hardware. This is further described in the following Section.

The **second layer** deals with the aggregation and storing of context data. The components at this layer are running in a server environment, mainly because the aggregation involves collecting data from multiple mobile sources. Also it involves higher computational capabilities that are available on the user's smartphone without interfering with his/her own activities. The components are distributed, and we envision a scheme where several such servers collect data based on a localization approach.

At this layer the information is received from several context sources. It is further organized based on concepts from a predefined model. At his layer the data is organized according to the proposed Context Model. For example, the data from several sensors (GSM, WiFi, Bluetooth) is aggregated into current Location, and the user can experiment with various location algorithms. The user's characteristics are organized based on a FOAF and semantic technologies [6]. We therefore are able to aggregate data into models describing actual relations between users, inferring information about their interests and activities. In an academic environment this allows defining rules specific to users interested in particular research area, or belonging to particular classes.

This layer also provides an abstraction layer which can be used by all applications to access context information. The domain described by the model acts as a contract between the middleware system and consumer applications.

The information and services offered by the contextualization services are consumed by two sorts of applications. Autonomous applications can use the services directly to access context information. They control entirely the way they react to context changes.

In addition, we define a **third layer**, which uses context Rule actions. Changes in the context may trigger different actions on the mobile phone according to a predefined rule set. The rules are expressed in an XML-based format and are stored in a remote repository. The user is therefore able to dynamically load and execute on the local mobile phone specific rules, depending on his/her own preferences. An example of such a rule is presented in Figure 3, which notifies the interested and available user about an event in the field of Distributed Systems. The main rule consists of two standard rules united by the logical operator AND. The first rule retrieves context information regarding user agenda or university timetable.

```
<rules-config>
    <rule-definitions>
        <rule-def name="DistributedSystemEventNotification"
action="category.EVENT_NOTIFICATION">
            <rule name="userIsFree" />
            <operator name="AND" />
            <rule name="userHasInterest" />
        </rule-def>
    </rule-definitions>
    <rule-implementations>
        <rule-impl name="userIsFree" class="rules.StringFieldEquals">
            <property name="argField"
value="CURRENT_ACTIVITY"/>
            <property name="target" value="free"/>
            </rule-impl>
    </rule-implementations>
    <rule-implementations>
        <rule-impl name="userHasInterest"
class="rules.StringFieldContainedInList">
            <property name="argField" value="INTERESTS"/>
            <property name="target" value="Distributed Systems"/>
        </rule-impl>
    </rule-implementations>
</rules-config>
```

Fig. 3. Example of a context rule.

Finally, the **fourth layer** is responsible with the applications, expressed as rules and actions, which can be used for orientation, information and recommendation purposes. At this layer there are local utilities that can help with context-triggered actions. There are also the applications that use the context data to improve response to stimulus (an interior or exterior request). An application can react to changes in the current context and take specific actions depending on some predefined rules. For this, conditions are evaluated period as the data is retrieved. Third party applications and services can use the API provided by the context-aware services. They can

use functions for obtaining particular context data, using filters, or can subscribe for context data. They can also declare new execution rules for users to install on their mobile devices.

4. The Context Model

We model a generic *context*, seen as all information that can be used to characterize the situation of an entity [1]. Entity is any person, place, or object that is considered relevant to interaction between a user and an application including the user and application themselves. In accordance, the context is the collection of information used in some particular form.

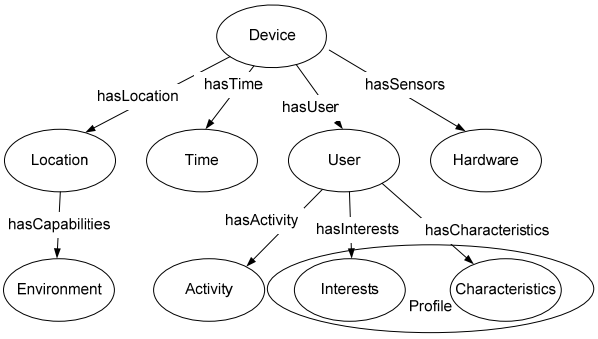

Fig. 4. The proposed context model.

The context model includes external data (relative to the environment of the mobile device executing the application, such as location, proximity) or internal information (to the device, such available disk space, battery, capabilities, etc.). The proposed context model aggregates this information into a unique set of data. The context is build based on all detectable and relevant attributes of the mobile wireless device, the application's user, the device's surrounding environment, and the interaction between mobile devices (creating an ad-hoc social interaction map) (Figure 4).

The hierarchical context model has several layers. On the first layer is the device, grouping together location, time, the user's identity, and the information gathered from various hardware sensors. The device object also provides static information about the device, such as its identifier, operating system and platform, etc.

Location is obtained from several sources. For out-door locality we use the GPS or GSM capabilities of the device. For in-door location we combine information received from several sensors, such as GSM cells, WiFi access points, and hardware devices capable of recognizing bluetooth pairing. The platform also allows experimenting with various in-door locality algorithms and solutions. In this case first the user constructs a module (if one is not already available) for collecting information from sensors. It then aggregates the information into a recognizable form of location data (e.g., the user is in front of a predefined room).

The user's identity is made available from the certificates installed on the mobile smartphone. The identity information is used for discovering relevant services. It can also be used for situation-awareness where the application recognizes the user, its location and take an action to automatically open the door (as described in a subsequent Section).

If the user's identity is found, it is augmented with additional information, such as the user's profile and activities. The user's activities are discovered from his/her agenda, or from the user's academic schedule (if the user is a student, based on his certificate the schedule is discovered by interrogating the university's data management system).

The profile context includes information related to the user's research interests, academic interests, or social interests. For the research interests a special service collects and aggregates data from scientific research databases and provides a set of features including automatic collection of information, guided and focused retrieval of researcher profiles, aggregation and storage of structured data in time, aggregated and personalized view of collected information.

The user's profile is provided in either a static form (for example, based on the certificate the user's current academic profile can be easily extracted from the university's digital record database), or is inferred from social networks. For that the application uses as data sources several social networks: facebook, linkedin [2]. These sources provide dynamic information about user's interests for example. But they also provide information about social relations between users. So, instead of asking users to insert their social preferences again, we learn them from the users' social networks and devise new connections based on the supplementary context information. This allows making queries to the system asking for the whereabouts of the user's current friends, representing users with current interests situated in the immediate proximity, or finding friends that can serve some specific events.

The context also includes system information. For example, special designed collecting modules can use the mobile smartphone's sensors (for battery level, light intensity, accelerometer, etc.). In addition the hardware context includes information gathered from external sensors (from sensors in the environment).

Our vision is to use the context information as part of the processes in which users are involved. The context can support the development of smart applications capable to adapt based on the data relevant to the user's location, identity, profile, activities, or his/her environment (light, noise, speed, wireless networking capabilities, *etc*.). We propose the use of a context model

that includes these parameters. Based on this model we propose *building smart and social environments capable to adapt to context using mainly the sensing and processing capabilities of users' mobile smartphones.*

In this sense the context model could support an academic environment in which users (students, teachers, etc.) may be endowed with a portable device which can react to changes in context by adapting the interface to the user's abilities (increase the luminosity when user is in a dark room, but not if a presentation is in progress) and profile (academic stuff are presented with a different set of services than students), increase the precision of information retrieval (use the context information relevant for the user's current action), or make the user interaction implicit (assume its interest based on his/her profile).

4.1. A semantic-based model

As a case study, we present the semantic-based context model. The data is aggregated in CAPIM using this semantic model, but other models are also supported. For example, data is also collected as time series, for long-term and near real-time processing guarantees.

The semantic model provides a vocabulary to represent knowledge about context and to describe specific situations. The Context Ontology defines a common vocabulary to manage and share context data. The advantage of such an approach is sharing a common understanding of information to users, devices and services, because the ontology includes machine-interpretable definitions of basic concepts and relations.

The aggregation and semantic services are running on server-side (second level in Figure 1) because the semantic aggregation involves collecting and aggregating together data from multiple sources. All mobile devices send context information to the aggregation service, where it is further managed and semantically organized. The aggregation service is also responsible to infer the stored data and send aggregated information back to uses or applications (Fig. 5).

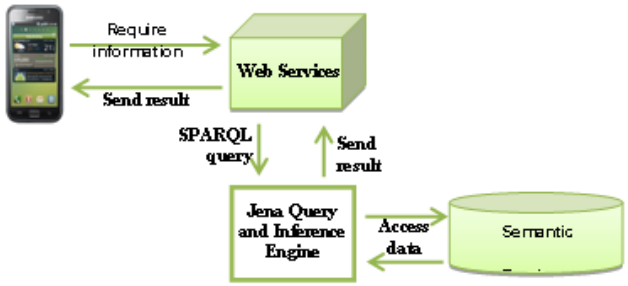

Fig. 5. The flow of query the semantic service.

The aggregated semantic data is kept in a semantic database. Our repository implementation uses the Jena Sematic Web Toolkit ([8]). The framework provides functions to add, remove, query even to infer data on generic RDF models.

The context ontology should capture all context information and model the basic concepts of person, interests and activities, describing the relationships between these entities. Considering as an example the pervasive computing which can be divided in a collection of sub-domains (e.g. home domain, school domain), we composed our ontology using domain-specific ontologies. Therefore, and considering its specific characteristics, CAPIM user's characteristics are organized based on **FOAF** [9] ontology (Figure 6). In this way we can describe user's activities and his relations to other people and objects. To model a paper or a book we use **PUBL** [10] ontology, storing and linking in this way information such as authors, publishing company or the release date. To describe events, dates or locations we use the **ICAL** and **GEO** [11] / **WAIL** [12] ontologies.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:org="http://www.w3.org/ns/org#">

    <foaf:Person rdf:about="andreea.starparu">
        <org:memberOf rdf:resource="Gr341C3"/>
        <foaf:name>Andreea Starparu</foaf:name>
        <foaf:nick>andreea.starparu</foaf:nick>
        <foaf:interest>Semantic_Web</foaf:interest>
        <foaf:interest>Distributed Systems</foaf:interest>
        <rdfs:subClassOf rdf:resource="prezentare_licenta"/>
        <wail:location>
            <geo:Point>
                <geo:lat>47.235</geo:lat>
                <geo:long>25.581</geo:long>
            </geo:Point>
        </wail:location>
    </foaf:Person>
    <ical:vevent rdf:about="prezentare_licenta"/>
        <ical:summary>thesis presentation</ical:summary>
        <ical:dtstart rdf:datatype="xsd:data">2011-07-11</ical:dtstart>
        <ical:dtend rdf:datatype="xsd:data">2011-07-15</ical:dtend>
        <ical:location>
            <geo:Point>
                <geo:lat>47.235</geo:lat>
                <geo:long>25.581</geo:long>
            </geo:Point>
        </ical:location>
    </ical:vevent>
</rdf:RDF>
```

Fig. 6. Example of FOAF description.

The main benefit of using domain-specific ontologies is that we can dynamically plug and unplug them from our model when the environment has changed. We prefer to create the CAPIM ontology based on others already implemented ontology because in this way the redundancy can be avoided and our semantic stored data can be easier linked with other information on the web.

Using the context ontology in a CAPIM academic scenario, for example, we can query and infer the stored data finding out new useful information easier. To illustrate our modeling concept we can describe a typical scenario: to socialize, in a break, first year computer science student Tom wants to discuss about Semantic Web with his interested mates. For this he just needs to use CAPIM service. It will interrogate the aggregation service, which will send to device the required data. With a relational model, the service should have to iterate through all CAPIM users, to find their locations and their interests. This semantic model has all this data linked, so the result is obtained faster without being affect its validity.

5. Evaluation and Prior Work

A possible application of the proposed platform and context services is an automated support for people in an university, who may be endowed with a portable device which reacts to changes of context by (a) providing different information contents based on the different interests/profiles of the visitor (student or professor, having scientific interests in automatic systems or computer science, etc), and on the room he/she is currently in; (b) learning, from the previous choices formed by the visitor, what information s/he is going to be interested in the next; (c) providing the visitor with appropriate services – to see the user's university records only if appropriate credentials are provided, to use the university's intranet if the user is enrolled as stuff; (d) deriving location information from sensors which monitor the user environment; (e) provide active features within the various areas of the university, which alerts people with hints and stimuli on what is going on in each particular ambient.

```
<?xml version="1.0" encoding="UTF-8"?>
<rules-config>
    <rule-definitions>
        <rule-def name="showRestaurantSuggestion"
        action="category.PLACE_SUGGESTION"
        parameter="restaurants">
        <rule name="isLunchTime" />
    </rule-def>
</rule-definitions>
<rule-implementations>
    <rule-impl name="isLunchTime" class="rules.IntFieldBetween">
        <property name="argField" value="TIME"/>
        <property name="targetStart"  value="13"/>
        <property name="targetEnd" value="14"/>
    </rule-impl>
</rule-implementations>
</rules-config>
```

Fig. 7. Another context-based rule example.

The proposed context-aware platform can be used for the experimental evaluation of many solutions. Users can evaluate methods for gathering context information, for aggregating data using semantics, ontologies. Such approaches were demonstrated in [7].

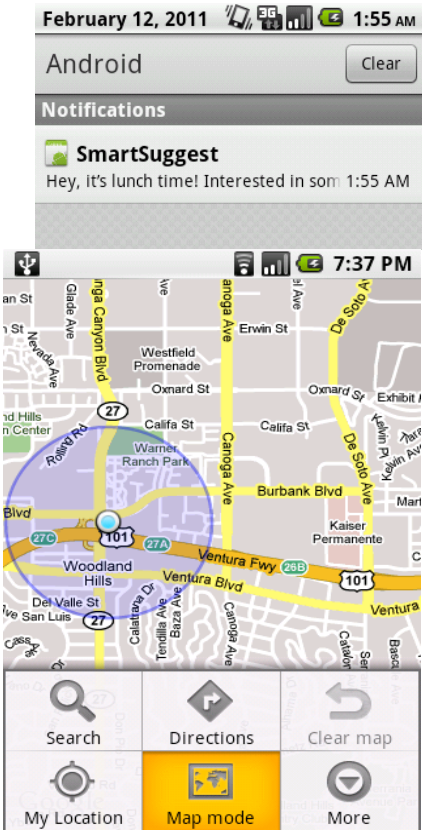

Fig. 8. Expanded notification (up), followed by suggestion of nearby restaurants (down).

In addition, the platform allows experimenting with complementary context-aware solutions. Consider for example a security component designed to offer a session establishment mechanism, along with session verification processes. Services might use it to verify the identity/authorization of the user currently being the possession of the smartphone. A session can be established through HTTPS using certificate authentication. The solution can, for example, allow users to unlock doors within a building without the requirement of using a physical key or any other replacements (smartcards, swype cards, etc.). All that is required is a smartphone present in the proximity of the door and a valid user X.509 certificate installed within the phone.

Another example is based on the rule described in Figure 7. In this example the context is evaluated. When it is lunch time (anywhere between 13 and 14 hour), the rules triggers an action which, based on the user's current location and using Internet services, finds all restaurants nearby. A notification is then brought up. If the user is interested he can access more details about the suggested nearby restaurants. In another situation, the application

observes that the user is in a free time interval according to his/her agenda and place (location), and also that the weather is sunny (using weather Internet services). According to the user's settings it can suggest parks nearby, or other similar outdoor activities close to the user's current location.

An example of an execution of the rule is presented in Figure 8. As a result of the execution, the user is presented with a notification and restaurants suggestions nearby current location.

6. Conclusions

Smartphones are today becoming commodities. Considering that even today more than half a billiard people have at least one smartphone, the previous affirmation is not so far-fetched. The advances in mobile technologies allowed people to have in their pockets, wherever they go, powerful computing devices, which can be of great help in their activities. Besides portability, these gadgets present another great feature: they have the necessary hardware capabilities to sense the environment. These advantages can be used to make mobile devices and the applications they host to be aware of the context they work in.

CAPIM is a platform designed to support the shift towards massive quantities of real-time information becoming access push rather than demand pull on a global case. It integrates services to monitor and a context for adapt with the user's context using sensors and capabilities of smartphones. It integrates context-aware services that are dynamically configurable and use the user's location, identity, preferences, profile, and relations with individuals, as well as capabilities of the mobile devices to manifest themselves in many different ways and re-invent themselves over and over again. Such services aggregate and semantically organize the context data. They react based on dynamically defined context-oriented workflows, and the platform includes an execution engine that supports context-aware actions for orientation, information, and recommendation.

A pilot implementation of the contextualization platform has proven the great advantages it provides in terms of simplicity and flexibility.

Acknowledgments

The research presented in this paper is supported by national project: "TRANSYS – Models and Techniques for Traffic Optimizing in Urban Environments", Contract No. 4/28.07.2010, Project CNCSIS-PN-II-RU-PD ID: 238. The work has been co-funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

CAPIM's official site and source code repository are available at http://cipsm.hpc.pub.ro/capim.

References

[1] Dey, A.K., "Understanding and Using Context". *Personal Ubiquitous Comput.*, 5(1), January 2001, 4-7.

[2] Skeels, M.M., and J. Grudin, "When social networks cross boundaries: a case study of workplace use of facebook and linkedin", in *Proc. of the ACM 2009 Intl. Conf. on Supporting group work* (GROUP '09), ACM, NY, USA, 2009, pp. 95-104.

[3] Chan, A.T.S., and S.-N. Chuang, "MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing", *IEEE Trans. Softw. Eng.*, 29 (12), 2003, pp. 1072-1085.

[4] Capra, L., W. Emmerich, and C. Mascolo, "CARISMA: Context-Aware Reflective Middleware System for Mobile Applications", *IEEE Trans. Softw. Eng.*, 29 (10), 2003, pp. 929-945.

[5] Bellavista, P., A. Corradi, R. Montanari, and C. Stefanelli, "Context-aware Middleware for Resource Management in the Wireless Internet", *IEEE Trans. Softw. Eng.*, 29 (12), 2003, pp. 1086-1099.

[6] Banford, J., A. McDiarmid, and J. Irvine, "FOAF: improving detected social network accuracy", in *Proc. of the 12th ACM Intl. Conf. on Ubiquitous Computing*, New York, NY, USA, 2010, pp. 393-394.

[7] Pop, F., M. Andreica, C. Dobre, R. Rughinis, A. Moldoveanu, C. Boiangiu, "eUPB: Towards an Integrated e-Service Platform in Large Scale Distributed Environments", in *Proc. of 18th Int. Conf. on Control Systems and Comp. Science (CSCS18)*, Bucharest, Romania, 2011, pp. 509-521.

[8] McBride, B., "Jena: A Semantic Web Toolkit", *IEEE Internet Computing*, 2002, pp. 55-59.

[9] FOAF, official website, last accessed June 07, 2011, from http://xmlns.com/foaf/spec/.

[10] PUBL, official website, last accessed June 07, 2011, from http://ebiquity.umbc.edu/ontology/publication.owl.

[11] GEO, official website, last accessed June 07, 2011, from http://www.geonames.org/ontology/documentation. html.

[12] WAIL, official website, last accessed June 07, 2011, from http://www.eyrie.org/~zednenem/2002/wail/.