Using a Cloud-based Event Service for Managing Context Information in Mobile and Ubiquitous Systems

Waldir R. Pires Junior[1], **Antonio A. F. Loureiro**[1], **Ricardo A. R. Oliveira**[2]

[1]Department of Computer Science – Federal University of Minas Gerais
Belo Horizonte – MG – Brazil

[2]Department of Computer Science – Federal University of Ouro Preto
Ouro Preto – MG – Brazil.

{wpjr,loureiro}@dcc.ufmg.br, rabelo@iceb.ufop.br

Abstract. *In mobile and ubiquitous computing systems, profile and context information from mobile users constantly change over a period of time. It is also desired for local and remote services to effortlessly access this information for adaptation of activities and event notification for mobile users. This work proposes an event-based system for managing context information in ubiquitous services and applications. This approach uses an event service to manage the events representing local and remote changes in the environment, allowing relevant information to be shared amongst interested services and mobile users. In our tests, an event service (tourist guide scenario) proved useful in disseminating changes in profile and context information between entities in a simulated ubiquitous environment.*

Resumo. *Em sistemas computacionais móveis e ubiquos, informações de perfil e contexto de usuário móveis podem sofrer mudanças constantes durante um determinado período de tempo. É desejável também que serviços locais e remotos possam acessar de uma forma simples e unificada estas informações a fim de proverem recursos tais como adaptação de atividades e notificação de eventos para usuários móveis. Este trabalho propõe um sistema baseado em eventos para o gerenciamento de mudanças de perfil e contexto locais e remotos, permitindo assim o compartilhamento de informações entre serviços e usuários móveis de interesse. Nos testes realizados (guia turístico), o servidor de eventos proposto mostrou-se útil em disseminar as mudanças de informações de perfil e contexto entre entidades em um ambiente ubíquo simulado.*

1. Introduction

Ubiquitous computing defines a new computational model of human-machine interaction in which information processing is integrated to daily objects and activities for the user. In this new paradigm, a user can activate and participate in several simultaneous and unconscious tasks and activities, in some cases not even being aware of the devices present in the surrounding environment. This introduces the necessity of context-aware computing, which proposes the capability of devices to sense changes in the environment and in the user's behavior. Context is defined [Dey 2001] as being "any information that can be used to characterize the situation of entities", such as the location, time of day, people, devices and services nearby, and user activities. Ubiquitous computing makes use of this information collected from the environment for context definition and service adaptation in real time. This adaptation is defined [Rossi and Tari 2006] as the capacity of a system or middleware of modifying its behavior in response to changes in the environmental context. In this way, mobile applications as well as remote services can utilize the information collected and present at the context for providing services and content for the mobile user and for applications and services present at the mobile device.

Ubiquitous environments, in general, contain applications and network elements that are both heterogeneous and distributed in fashion. They are heterogeneous in the sense of performing different types of tasks for the user, requiring different sets of hardware and software components. They are also distributed in such a way that services may be composed of more than one computing element. For instance, a network element may depend on other devices for obtaining information about the environment and executing tasks for the user. As a result, a distributed communication model is required to seamlessly integrate these elements in the environment.

In the following, we briefly describe a tourist guide application that conveys the ideas discussed in this work. Suppose a tourist begins his/her day at the hotel and uses a mobile phone or PDA to search for tourist attractions and confirming the purchase of electronic tickets. While the tourist guides (people and companies) are contacted by the guiding system, the tourist can confirm the weather forecast and elaborate a route around the city. The route information is sent to the system, which offers existing services in the path such as shopping options, restaurants, among other activities.

After the tourist has begun his/her trip, the tourist guide application, based on previous defined interests and the necessary tickets purchased early in the morning, presents some leisure options that interest him/her the most. Caring about good services, the tourist remembers to leave a favorable comment to this guide in the service quality evaluation system. While visiting each attraction point, the tourist receives at his/her mobile device an electronic flyer describing the attraction and, according to his/her interests, some propaganda of products on sale at each location.

At any moment, the tourist may change his/her path. The guiding system running on the mobile device detects this change in the location and updates the information related to the services offered in this new path. In case the tourist is lost, the user can request some directions from the map service provided by the system, allowing him/her to return to the previously planned route and the attractions defined previously, or follow onto a new attraction of interest.

1.1. Problem and Contribution

Depending on the scenarios involved, user profile and context information may suffer constant changes over a period of time. Take for instance, changes of weather and traffic information, user or device location and state. Tourist guide applications are a good scenario example, since they contain virtually all the characteristics mentioned above. Activities initially selected by the user may become unfeasible due to traffic and/or weather conditions at that region or the user may simply not feel well or up to performing that activity due to his/her condition. The mobile device may also suffer difficulties or limitations at a certain moment, such as energy constraints, connectivity state and transmission costs.

This variability in profile and context information at context-aware ubiquitous applications promotes a significant challenge in managing these changes in a distributed mobile computing environment. This challenge includes the need to detect, collect, process, publish, subscribe and consume occurring events in the system. These changes occur in the environment (e.g., physical or logical) where the device is located, and the events generated must incorporate information relative to these changes. Physical context can be defined as the information relative to the outside environment, such as the user location, traffic and weather states, among others. Logical context relates to the conditions concerning the user and the mobile device, such as mood, time availability, battery level, connectivity state, among others. Interested event consumers use this information for reaction to changes detected, for example, by the means of adaptation and notification.

The main contributions of this work are:

- **Management of local/remote profile and context information from the user**: This work provides the provisioning of profile and context information from the user, application and the environment to other entities (e.g., other mobile users and Web-based services) in a mobile/ubiquitous system.
- **Fast provisioning of profile and context information from the user to other entities**: This work allows ubiquitous applications and services to distribute, access and share profile and context information amongst other entities in a decoupled and distributed manner by using event objects and notification messages sent over the wireless network.
- **Usage of a cloud-based infrastructure for the provisioning of ubiquitous applications and services**: This work makes use of cloud computing model based on some features such as the virtualization of hardware and software resources at the server side, making these features accessible in a seamless manner in the form of services on the Internet.

This paper is organized as follows. In Section 2, we briefly describe the related work. In Section 3, we present the event-based model proposed in this work, its architecture and the event processing workflow. In Section 4, we present the case study chosen for evaluating the proposed service called the DroidGuide. In Section 5, we discuss the conclusions and future work.

2. Related Work

Context-aware computing in ubiquitous environments and event-based distributed systems define the main areas of research of our work. The event-based communication model defined in Meier and Cahill [Meier and Cahill 2002] describes a useful paradigm in providing the interconnection between elements that comprise applications for ubiquitous environments in an asynchronous manner. This model allows the association between application components (producers and consumers) and events that are generated in ubiquitous system by the means of notification messages. It enables elements (devices, components and applications) to react to state changes of other elements, providing interested parties with notification messages based on these changes.

Muhl et al. [Muhl et al. 2006] define an event as being an occurrence of interest that can be observed by a computer element (e.g., PC, sensor, or any other device). One can define an event in a more global aspect as being a change of state where a system entity is responsible for creating an event instance representing the changes in which other elements will respond, react and/or adapt by the means of imposed rules at the service and application levels. In our work, changes in the environment are represented by event objects that may be consumed by client peers and Web services.

Caporuscio and Inverardi [Caporuscio and Inverardi 2005] present a design of an event-based system, which allows applications to detect events at a certain region and evaluate their relevance and uncertainty taking into account the applications' main context. This allows the application to adapt to certain environmental conditions and reach its purposes. A specification is presented which includes the design of a prototype based on a publish/subscribe middleware. Other event-based systems proposed include Sacramento et al. [Sacramento et al. 2004] who proposed a middleware architecture for the development of context-aware services and applications for wireless network environments. Carzaniga et al. [Carzaniga et al. 2001] also propose four architectural solutions for distributed event-based systems: client/server, acyclical P2P (Peer-to-Peer), redundant P2P and hybrid. Those proposals can be used in the construction of context-aware ubiquitous systems for the dissemination of context information amongst various client peers.

Despite the solutions above, we chose to utilize a client/server architecture provided by the cloud-based computing model for several reasons. Some justifications in using this model for the virtualization of services and resources over the Internet include support for popular Web-based application layer protocols such as HTTP and XMPP, service mobility (e.g., device and location independent), flexibility (e.g., speed in redefining computing resources and scalability) and the possibility for application designers to focus on other application issues rather than in service and platform infrastructure.

Several use case scenarios may benefit from the usage of ubiquitous applications and services. These scenarios in general contain functional requirements such as the need for collecting, processing and sharing profile and context information from the user/application, adaptation, and non-functional requirements such as security, usability and mobility. The adaptation occurs according to the user's intentions and interests, to the graphical interface and to available services for the mobile user while security involves privacy, authentication, authorization and anonymity. From all those scenarios, we chose to construct and evaluate one: an electronic tourist guide.

3. The Event-based Service

The service proposed in this work is based on the event-based model described in [Muhl et al. 2006]. Our model comprises of nodes in three different categories: client, server, and service. Client nodes represent the mobile users containing devices capable of accessing services over a wireless network. The server node contains an interface for accessing available information-based remote services. These services are represented by service nodes that can receive events from the client and server nodes and respond back to the server when required. Client nodes can access the available Web services and other client peers by using the existing server node.

An overview of the related scenario can be seen in Figure 1. In this overview, one can note the presence of mobile and fixed users accessing and providing information to services available over the Web. The information shared is represented by event objects created, published and consumed by each entity in the system. In order for entities to be capable of publishing and consuming these events to all interested parties, an event-based server receives and dispatches events to consumers according to subscriptions to elements such as interested services, activities and topics. In this way, only events that match the entities interests are delivered to each client. This event-based service offers subscription and event management services related to profile and context information collected from participating entities in the system.

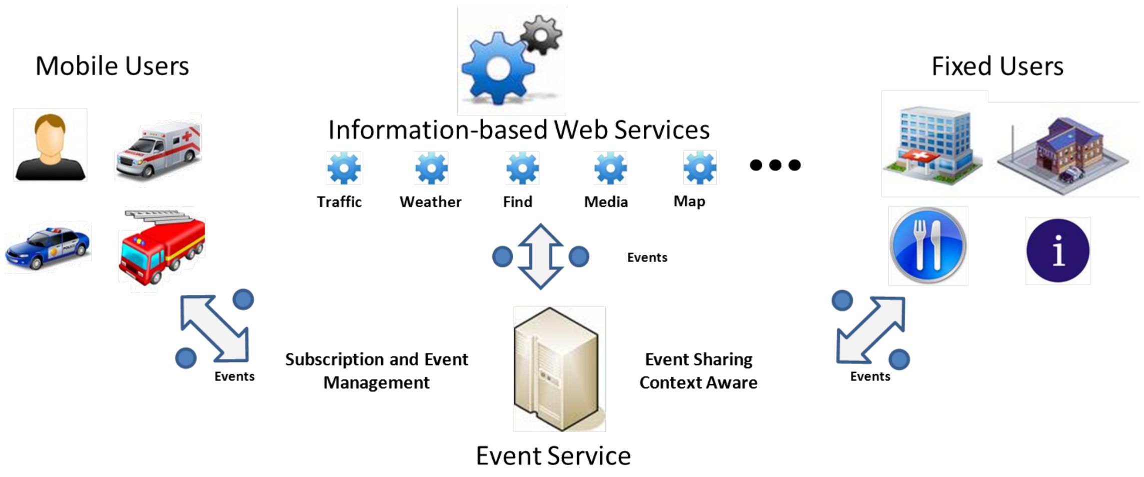

Mobile Users
Information-based Web Services
Fixed Users
Traffic
Weather
Find
Media
Map
Events
Events
Subscription and Event Management
Event Sharing Context Aware
Events
Event Service

Figure 1. An overview of the proposed service.

The proposed service in our work contains several responsibilities in the ubiquitous computing environment. First, the service must periodically detect and collect changes in profile and context information from the mobile user. This is done in order to detect explicit and implicit changes in the environment, such as changes in location, interests and conditions, among others. In order to maximize the communication between client and server peers, the event processor located at the client plays an important role by creating event objects whenever there are local changes in profile and context information. In this way, the processor only reports the collected changes to the remote event service located at the server.

Secondly, our event service allows access to profile and context information by Web-based services in the form of event subscriptions based on interest topics. Suppose, for instance, that the mobile user wishes to change his logical context condition, such as mood or hunger state. If the mobile user feels agitated or hungry, he/she updates this condition at the mobile application in execution while in transit or inside a tourist activity. With this logical context update, the event processor detects the changes and sends them to the event server for further processing. The event server then publishes this information to services interested in the user's mood and hunger states. Based on the information received, nearby services and activities can be offered to the user, allowing him/her to rest or eat in locations nearby his/her position. By defining his/her preferences, the mobile user informs the event server the desire to share his/her profile and context information, allowing these services to monitor and react to changes informed by the user.

3.1. System Architecture

The event-based service proposed in this work is composed of two main components: the event server residing at a remote Web server and the event processor located at the client side. Both components are responsible for generating event objects representing changes in the user profile and context information. The event server is responsible for handling events generated at the server side and also process events detected at the client side by the event processor. In turn, the event processor manages events detected at the client side and receives notification messages from the event server. Web-based services can take advantage of the relevant local and remote context information collected by the event processor/service, by receiving and processing these events. As a result, the Event Service Module provides subscription, filtering and event notification services to mobile applications and services that are both local and remote in relation to the mobile user.

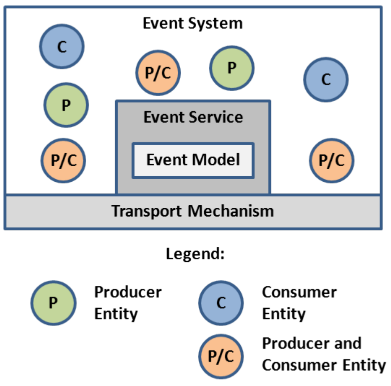
Different from the proposed architectures described in Carzaniga et al. [Carzaniga et al. 2001], our system uses a cloud-based solution where client peers connect to a unique server. This was done to simplify the development process and infrastructure provisioning at the server side. Cloud computing is a style of computing in which dynamically scalable and often virtualized resources are provided in the form of a service on the Internet [Miller] . As a result, applications and infrastructure reside not locally, but at a remote location accessible over the Internet. Therefore, users do not require having the knowledge of, expertise in, or control over the technology infrastructure in the ”cloud” supporting them.

3.2. Event System Classification

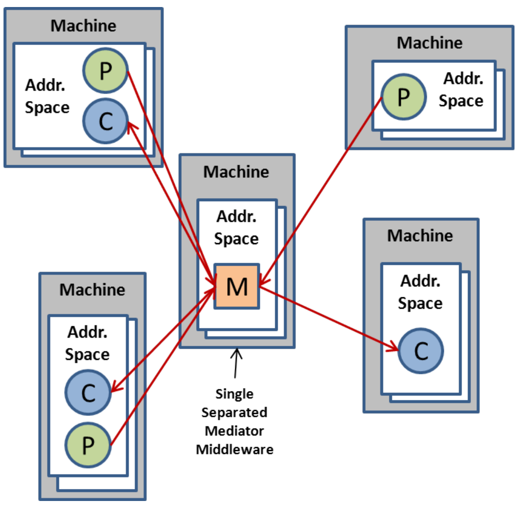
A taxonomy has been proposed [Meier and Cahill 2002] for classification of event-based systems in several characteristics, such as the event model used, event service characteristics and functional/non-functional features available. Other event-based systems have been used in the classification, such as the CORBA notification service model[1], SIENA [Carzaniga et al. 2001], SECO [Haahr et al. 2000] and Hermes [Pietzuch and Bacon 2002]. Figure 2 describes the event service organization and interaction model as defined in the taxonomy for the service proposed in this work.

According to this taxonomy, the event service proposed in this work can be classified as follows. It implements a single mediator with a separated middleware, single centralized intermediate event model. It uses a single mediator between existing entities in the system, with this mediator being centralized and separated physically (different machines) and logically (different address space) from producers and consumers. In respect to the event propagation models defined in the taxonomy, our event system uses periodic pull, with typed events based on application specific attributes and without event hierarchy. The event service is an intermediate between mobile collaborative entities that are both producers and consumers. Events are delivered on a best-effort basis with no support for priority. Regarding failure handling, our system is characterized as a partial system failure for entities, functional partial system failure or total system failure for the event service middleware and partial or total system failure for the network. Other non-functional requirements such as security and ordering were omitted in our work. Tables 1 and 2 present the functional and non-functional features of the event service proposed in this work.

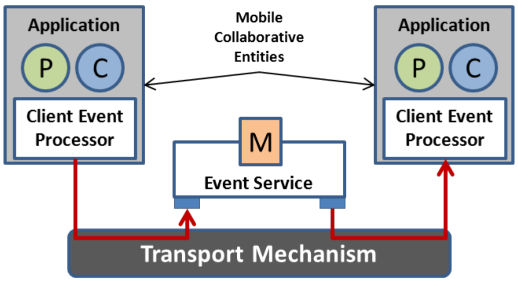
[1]http://www.omg.org/cgi-bin/doc?formal/04-10-13.pdf



(a) *Event system overview.*



(b) *Organization*



(c) *Interaction model*

Figure 2. The organization and interaction model of the system proposed according to the taxonomy [Meier and Cahill 2002].

4. Case Study: DroidGuide

This section presents the first case scenario with the prototype implementation of the Tourist Guide scenario in the form of an emulated mobile application. This aims to evaluate the capacity of the mobile device to detect and collect profile and context information and allow user activity selection based on the profile information defined by the tourist.

The mobile application proposed provides several features to the mobile tourist user. First, it displays a map containing all the activities available for him/her to consume, as shown in Figure 3(a). Mobile users define their tourist profile data in order for the system to propose activities and provide information regarding interest topics, as shown in Figure 3(b). Mobile users also define their condition or state by entering their logical context, as shown in Figure 3(c). Once profile and context information is defined, the application may begin offering activities as well as notify mobile users regarding client and server based events.

The software platforms used in our prototype application were Google Android[2] and Google Web AppEngine[3] for the client and server, respectively. Communication was achieved by using HTTP request messages being sent from client to server while

[2]http://code.google.com/android/
[3]http://code.google.com/appengine/

Characteristic	Service Proposed	Description
Event model	Single mediator	One event service mediator between entities.
Event service organisation	separated single middleware	Event service middleware in different address space.
Event service interaction model and location	Single centralized intermediate	Entities interact with a single mediator.
Event propagation model	Periodic pull	Request-response communication.
Event type	Generic	Events are generic throughout the system.
Communication	Unicast, multicast and broadcast	Entities communicate with each other using published events.
Expressive power	Application specific attributes	Attributes defined for the event define its epression.
Type hierarchy support	No	Support for inheritance between events.
Event implementation	Object and String	Events are represented as objects at the serve side and Strings at the client side.
Event evaluation time	Propagation	Events are evaluated at propagation time.
Mobility	Collaborative entity	Mobile entities collaborate across the system space.

Table 1. Classification of the event service proposed in this work according to the taxonomy.

Characteristic	Service Proposed	Description
Support for composite events	Yes	Programatic support.
Event delivery	Best effort	No deadlines associated with events.
Event priority support	No	Priorities for events at the event queue.
Store occupancy	Configurable	Support for memory (hash table structure) or DBMS persistence.
Reliability	Reliable connection and (temporarily) and persistent	Usage of a connection-oriented protocol (HTTP/TCP/IP)
Security support	No	Authentication, authorization and criptography.
Entity failure	Partial system failure	Partial failure at the entity.
Middleware failure	Functional partial system failure or total system failure	Depending on the failure, it can be partial (recovery) or total (no recovery).
Network failure	Partial or total system failure	Depending on the entities at the communication, being partial (between entities) or total (between entities and middleware).

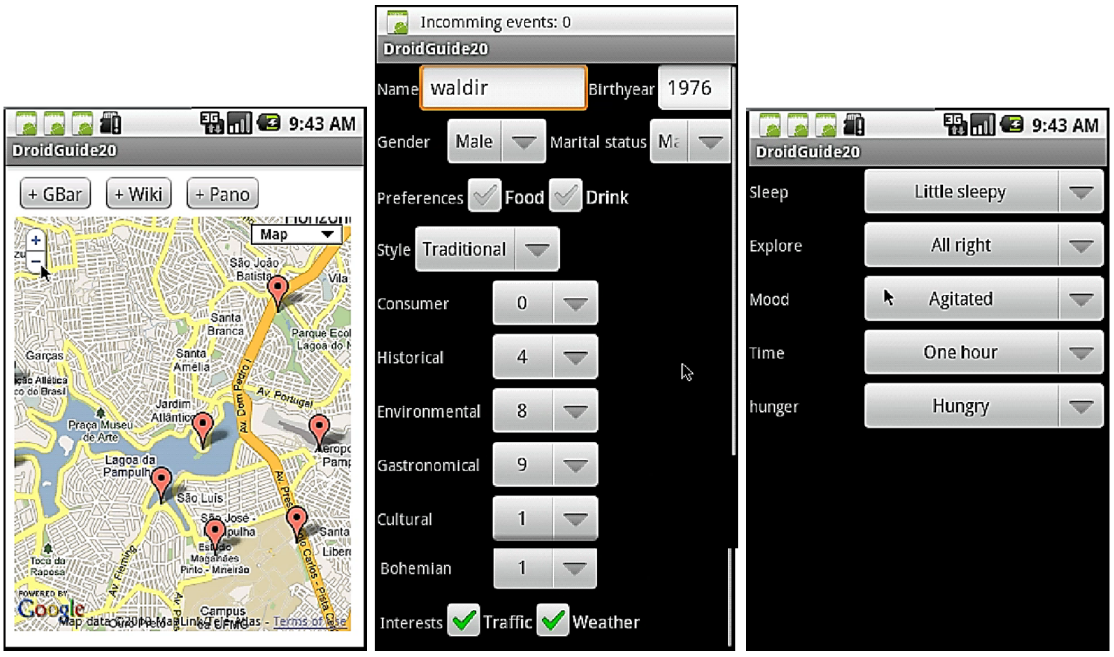
Table 2. Non-functional feature support according to the taxonomy.

the server communicates with clients via XML documents over the HTTP response messages. In our work, we used a pull-based [Muhl et al. 2006] client-server communication approach by sending periodic request messages from the client to the server. This enables the server to seamlessly send notification messages to client peers when required as well as the client peers to send data to the event service.

To simplify the implementation and integration of the many responsibilities that comprise the application, the DroidGuide guiding system was logically divided into modules with well defined responsibilities and relationships among them. As shown in Figure 4, these components are:

- At the mobile device: the profile and context management service (PCM), the client event processor (CEP) and the communication module (CM);
- At the server: the event processing service (EPS), the subscription manager (SM), the event container (EC) and the information-based remote-services container (IBWS).

From all modules, two of them should be noted due to their importance: the Client Event Processor (CEP) and the Communication Module (CM). The first one manages the



DroidGuide20

Name	waldir
Birthyear	1976
Gender	Male
Marital status	M
Preferences	✓ Food ✓ Drink
Style	Traditional
Consumer	0
Historical	4
Environmental	8
Gastronomical	9
Cultural	1
Bohemian	1
Interests	✓ Traffic ✓ Weather

DroidGuide20

Sleep	Little sleepy
Explore	All right
Mood	Agitated
Time	One hour
hunger	Hungry

(a) Map with attractions. (b) User profile data. (c) User context data.

Figure 3. The Tourist Guide prototype developed on the Google Android Platform 2.0.

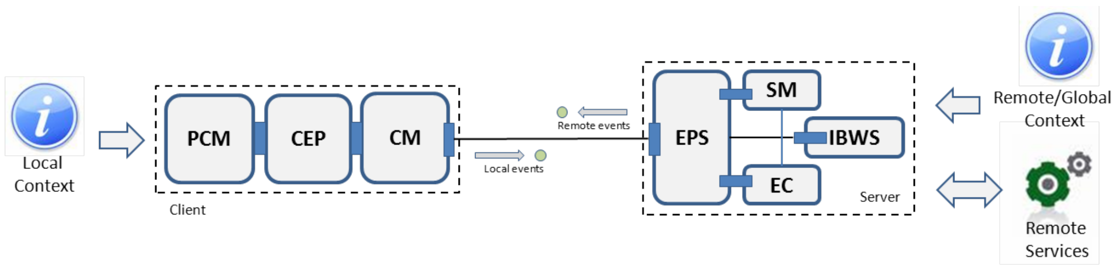


Figure 4. The Event-based system deployed into the DroidGuide application divided into modules.

changes in profile and context information by storing client events to be sent to the event server. The Communication Module manages the communication of all modules that make up the application, which include sending requests, receiving and processing XML messages from the server. The Event Processor uses a two-phase pull-based approach: (a) it sends HTTP GET requests from client to server, and (b) receives XML over HTTP response messages from the server to client. The communication module handles XML parsing of incoming data messages from the server, converting them to generic property-based objects. In our prototype, DOM (Document Object Model) parsing was used to convert XML messages to Java objects.

Located at the client side, the profile and context management service (PCM) is responsible for monitoring and collecting data regarding changes in profile and context information at the mobile device. This service submits the collected changes to the client event processor (CEP), which prepares the collected information for the communication module (CM). The communication module sends the information to the event processing service (EPS) located at the server. Once the information arrives at the server, the event processing service creates event objects representing them, stores them at the event container (EC) and sends them to the corresponding consumers according to the existing subscriptions managed by the subscription manager (SM).

Once these consumers (e.g., remote services) receive the incoming events, they may also generate new events due either to the events received or changes in remote profile and context information. In this case, they also generate events that are sent to the event processing service. After storing the new events at the event container, the service requests the subscription manager for subscriptions related to the new events and generates notifications to be sent to the client. At the HTTP response message, the service sends the related notification messages to the client for further processing by the communication module and client event processor. This incoming message is presented to the mobile user, for example, in the form of a popup message alerting him of the event.

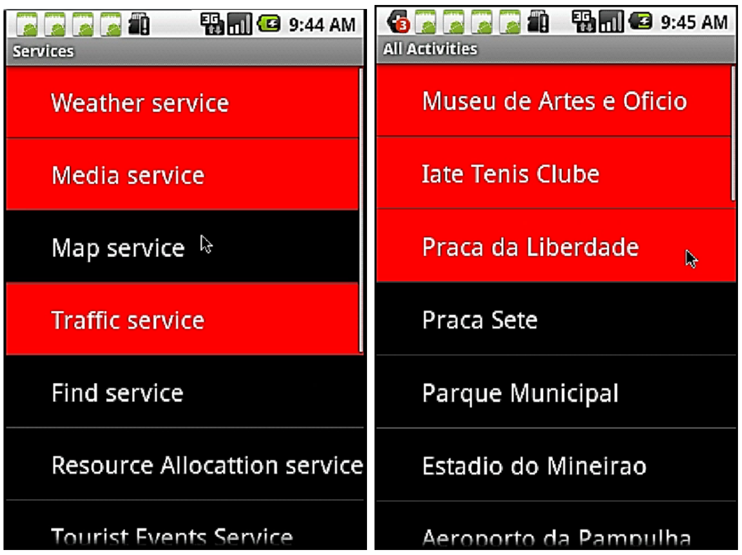
4.1. The Execution Flow

DroidGuide begins by requesting the tourist to login with his/her username and password. After the login process, the user is asked to define some attributes in his/her profile and context data, as shown in Figures 3(b) and 3(c). This can be later updated, even if in the future the tourist logs in again at another device containing the application. After the tourist defines his/her profile and context data, the event service takes action by monitoring local/individual or remote/global changes in this data. In the next step, the user is able to select available information-based remote services available at the remote data server for notification message provisioning. In our simulated scenario, two remote services were provided: traffic and weather.

After the tourist selects the interested services, the application begins selecting available tourist activities that can best suite his/her interests defined in the profile data, taking into account the six styles or interests already defined by the user: consumer, historical, environmental, gastronomical, bohemian, and cultural. The tourist grades each of these styles with a score from zero (not interested) to ten (totally interested). In general, every tourist activity available at the location contains a grade based on the same styles presented to the tourist. For instance, a restaurant activity has a high grade in the gastronomical style while a very low grade in environmental. Another activity, however, can have high grades in two or more styles such as in activities that relate to both historical and cultural styles. An example of the activity suggestion feature can be seen in Figure 5(b).

As proposed in the system, the tourist should also be capable of selecting other activities not initially presented to him/her by the system. The service capable of activity selection can be seen in Figure 5(b). For instance, there can be activities in which his/her friends have performed that were highly recommended to him/her and not initially suggested by the system. This is possible due to the publish/subscribe feature also available at the proposed event service. In our prototype, activity selection is treated in the same manner as in subscription to services and topics. Once the tourist selects the desired activities, he/she may begin visiting these activities.

During the execution process, the service can deliver notification messages to interested parties in case there are changes detected in the user's profile and context data.



(a) Remote services. (b) Tourist activities.

Figure 5. Subscription elements available to the mobile user.

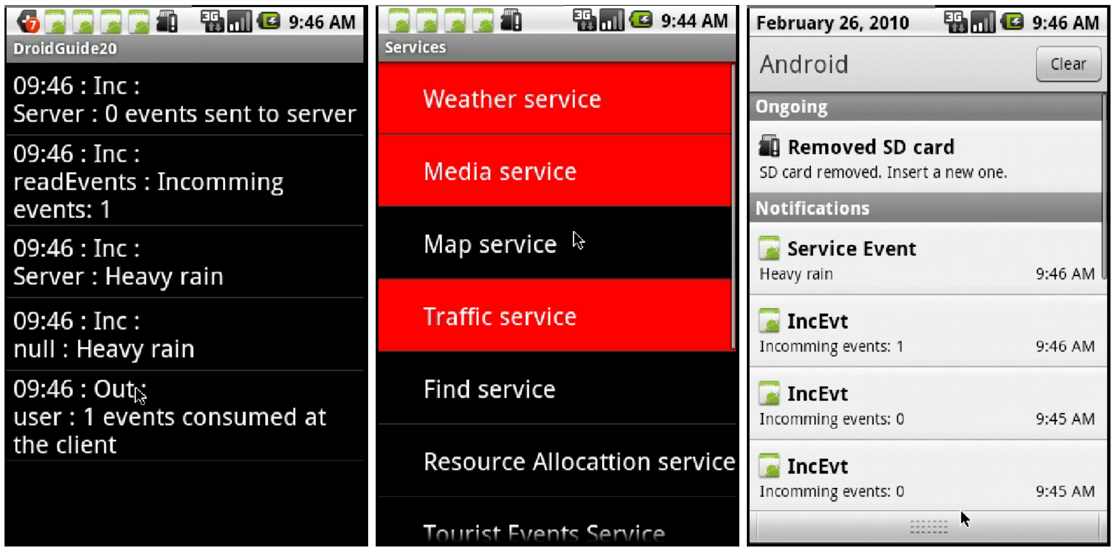
An example of a list of subscribed services can be seen in Figure 6(b). For instance, a remote restaurant finder service is interested in knowing when the tourist is hungry. When this occurs, the user updates his/her logical context data present at the mobile device. This change causes the event processor to send data to the server, allowing the creation of event objects representing the changes. The event service then shares these objects to interested/subscribed services. Once the information-based service receives the event object, it understands that a notification message must be sent to the user, informing him/her of a possible restaurant nearby. In the end, the user receives the notification informing him/her of the attraction nearby that can satisfy his/her hunger.

In our simulation, we created a weather event object at the server that was directly related to one of the tourist's interest topics and subscribed information-based remote services. Due to the subscription to the weather service, this service will notify the user with any changes of profile and context information occurring at the server – in our case a change in weather condition.

Figure 6 shows the creation of a server-side event and its notification at the client application. Figure 6(a) shows the notification screen presenting the incomming notification from the server. Figure 6(b) shows the list of remote services subscribed by the mobile user. Figure 6(c) shows the notification tab presenting the incomming notification to the user.

5. Conclusion and Future Work

In this work, we presented an event-based service for the management of local and remote profile and context information for client peers, as well as for information-based Web services. The event service was capable of capturing changes in profile and context data from both the client and the server sides. The events have been collected in the device and sent to the server by the communication module. According to the interests defined by the user in his/her profile, tourist activities were selected accordingly. Subscriptions


9:46 AM
DroidGuide20
09:46 : Inc :
Server : 0 events sent to server
09:46 : Inc :
readEvents : Incomming events: 1
09:46 : Inc :
Server : Heavy rain
09:46 : Inc :
null : Heavy rain
09:46 : Out :
user : 1 events consumed at the client

9:44 AM
Services
Weather service
Media service
Map service
Traffic service
Find service
Resource Allocattion service
Tourist Events Service

February 26, 2010 9:46 AM
Android
Clear
Ongoing
Removed SD card
SD card removed. Insert a new one.
Notifications
Service Event
Heavy rain 9:46 AM
IncEvt
Incomming events: 1 9:46 AM
IncEvt
Incomming events: 0 9:45 AM
IncEvt
Incomming events: 0 9:45 AM

(a) Notification screen. (b) Subscribed services. (c) Notification tab.

Figure 6. Notification of a server side event at the client.

to information-based services showed the viability of using events in these services for ubiquitous applications. These services were capable of sending notification messages to the client application, due to the creation of remote event objects when changes in remote/global context information were detected.

The future work may follow in several directions: (i) the evaluation of the event-based service in other environments and scenarios (for instance, in vehicular networks and in elderly care centers), (ii) definition of event processing profiles allowing event processing configuration according to the needs of the application (e.g., client, client and server and server), (iii) security (user authentication, anonymity and authorization and data encryption/decryption of messages) and message compression during transmission, and (iv) the evaluation of collaborative context-aware computing scenarios, the relationship and interaction amongst client peers.

References

Caporuscio, M. and Inverardi, P. (2005). Uncertain event-based model for egocentric context sensing. In *SEM '05: Proceedings of the 5th international workshop on Software engineering and middleware*, pages 25–32, New York, NY, USA. ACM.

Carzaniga, A., Rosenblum, D. S., and Wolf, A. L. (2001). Design and evaluation of a wide-area event notification service. *ACM Trans. Comput. Syst.*, 19(3):332–383.

Dey, A. K. (2001). Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7.

Haahr, M., Meier, R., Nixon, P., Cahill, V., and Jul, E. (2000). Filtering and scalability in the eco distributed event model. In *PDSE '00: Proceedings of the International Symposium on Software Engineering for Parallel and Distributed Systems*, page 83, Washington, DC, USA. IEEE Computer Society.

Meier, R. and Cahill, V. (2002). Taxonomy of distributed event-based programming systems. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 585–588, Washington, DC, USA. IEEE Computer Society.

Miller, M. Cloud computing pros and cons for end users. http://www.informit.com/articles/article.aspx?p=1324280.

Muhl, G., Fiege, L., and Pietzuch, P. (2006). *Distributed Event-Based Systems*. Springer, 1st edition.

Pietzuch, P. R. and Bacon, J. (2002). Hermes: A distributed event-based middleware architecture. In *ICDCSW '02: Proceedings of the 22nd International Conference on Distributed Computing Systems*, pages 611–618, Washington, DC, USA. IEEE Computer Society.

Rossi, P. and Tari, Z. (2006). Software adaptation for service-oriented systems. In *MW4SOC '06: Proceedings of the 1st workshop on Middleware for Service Oriented Computing (MW4SOC 2006)*, pages 12–17, New York, NY, USA. ACM.

Sacramento, V., Endler, M., Rubinsztejn, H. K., Lima, L. S., Goncalves, K., Nascimento, F. N., and Bueno, G. A. (2004). Moca: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*, 5(10):2.