J Braz Comput Soc (2010) 16: 163–176
DOI 10.1007/s13173-010-0015-1

ORIGINAL PAPER

Content distribution in trail-aware environments

Jader M. Silva · João H. Rosa · Jorge L.V. Barbosa · Débora N.F. Barbosa · Luiz A.M. Palazzo

Received: 25 November 2009 / Accepted: 3 June 2010 / Published online: 11 July 2010

Abstract In mobile computing environments, the tracking of users allows applications to adapt to the contexts visited by users (Context Awareness). In recent years, the use of context information and users' profiles has been considered an opportunity for context-aware content management. The improvement and the wide adoption of location systems are stimulating the tracking of users, allowing the use of Trails. A trail is the history of the contexts visited by a user during a period. This article proposes a model for trails management and its application in the content management. In this text, we consider that Trail-Aware is an evolution of the simple use of contexts and profiles. The text presents a prototype and its application in an educational environment for distribution of learning objects using trail-aware.

Keywords Mobile computing · Context awareness · Trails · Content distribution

J.M. Silva · J.H. Rosa · J.L.V. Barbosa (✉)
University of the Sinos Valley, Unisinos Ave., São Leopoldo, RS, 950, Brazil
e-mail: jbarbosa@unisinos.br

J.M. Silva
e-mail: jader.marques@gmail.com

J.H. Rosa
e-mail: joaohenrique89@gmail.com

D.N.F. Barbosa
La Salle University College, Victor Barreto Ave., Canoas, RS, 2288, Brazil
e-mail: nice@unilasalle.tche.br

L.A.M. Palazzo
Catholic University of Pelotas, Félix da Cunha Street, Pelotas, RS, 412, Brazil
e-mail: lpalazzo@ucpel.tche.br

1 Introduction

Currently, the evolution of mobile devices and high-speed wireless networks has been stimulating researches related to Mobile Computing [15, 39]. In this area, the improvement and proliferation of Location Systems [21, 22] have motivated the adoption of solutions that consider the user's precise location in the providing of services (Location-Based Services [13, 44]). However, lately, mobile applications have also become to take into account the user's current context to distribute content and services (Context Awareness [5, 7, 23, 30]).

The state-of-the-art context-aware computing enables applications taking decisions based on users' data (for example, their profiles) and their current contexts [23, 37]. Nonetheless, we understand that would be valuable also to consider the user's past actions performed in the contexts visited during a period, such as the activities did, the applications used, the contents accessed, and any other possible event. We believe that this would improve the distribution of content and services in context-aware environments, because applications would be using an additional and more complete information source. In other words, applications, instead of using only context and profile data, would also use a historic register of the user's actions to take decisions.

Studies on users' monitoring in mobile computing systems have shown that is possible to record the history of contexts visited by users and their actions performed in each context [29, 41]. Usually, this history is called Trail [16, 28]. Thus, in context-aware applications, trails would enable using the user's past behavior to distribute content and services. Hence, we propose the application of trails to context-aware computing, considering that Trail-Aware is an advance in Context-Aware computing. In this article, we introduce a model, which is named UbiTrail, for trails management in mobile computing environments. In this model,

applications can create their own services to obtain standard behaviors of the monitored entities. Moreover, UbiTrail propose an ontology to standardize the trails data.

An important application area of the joint use of context and profile is the education [4, 31, 46]. In this field, some applications distribute contextualized and personalized pedagogical content [6, 7, 30]. In the same way that occurs in context-aware applications, in the learning domain, applications are limited to take decisions based on profile and context information. Furthermore, the use of students' profiles in context-aware environments [8] and the automatic improvement of these profiles [29] are recent research topics. Nevertheless, content distribution guided by the students' trails is still an opened research area. Hence, we propose the use of trails in the educational scenario, expecting to improve the content distribution. In this article, we present a case study in which UbiTrail was integrated with a ubiquitous learning system (LOCAL [8]) and used for contextualized distribution of learning objects. The integration occurred in the context of the UbiCampus project, which is financed by CNPq (Brazilian Council for Scientific and Technological Development) through the Universal Announcement 15/2007.

The article is organized into six sections. Section 2 describes the UbiTrail architecture and the proposed ontology. The third section discusses the UbiTrail use in the content distribution guided by trails. Section 4 presents the UbiTrail and the LOCAL prototypes as well as the integration between them. This section still describes and evaluates the results obtained from a case study involving the use of trails in contextualized distribution of learning objects. Section 5 discusses related works, focusing on the contribution of this work. And finally, Sect. 6 presents the conclusions and the future works.

2 The UbiTrail model

This section presents the model of trails management. Section 2.1 describes the main terms used by the model, mainly those related to the trails composition. Section 2.2 describes the ontology proposed by UbiTrail, and finally, Sect. 2.3 presents the model architecture.

2.1 Entities, trails, ptrail, and trailpoint

UbiTrail was designed to manage entities' trails. In this context, entities represent persons, accessing computational resources (for example, a smartphone), or mobile objects (for example, a vehicle). We choose entity, instead of user, because then the model could manage trails of any mobile object. In this scenario, different applications can be developed for trails. For example, a vehicle could have a trail, recording information considered relevant, such as the occurrences of maintenances or failures.

Applications that use UbiTrail must temporally store data that composes a trail. For this reason, we propose that the applications have internally a structure called *ptrail* (piece of trail), which is composed of the following attributes: *entity*, *resource*, *event*, *extension*, and *location*. The values of the *entity* and *resource* attributes are stored statically in the *ptrail* after that an entity logs in the model. Nonetheless, the values of the *event*, *extension*, and *location* attributes are automatically updated as the actions or movements made by the entity and monitored by the application. The *ptrail* architecture was defined according to the UbiTrail ontology, which is discussed in the next section. The possible values of the attributes also are defined according to the ontology.

The records related to the contexts visited by an entity are stored in only one trail. The trail is composed of a sequence of records of the *ptrail*. A record is composed of ten attributes organized into three categories: *identification*, *content*, and *properties*. The first category contains only one attribute to identify the trail. All the records that compose a trail have the same identification. The second category is used to store the *ptrail* content. The last category contains the following attributes: *date/time*, representing the time of the record creation; *visibility*, indicating if the record is public or private (access restricted to the own entity).

We named *trailpoint* the process that makes the trail composition. This process sends the values contained in the *ptrail* to be recorded in the entity's trail, which is stored in a server. The *trailpoint* occurs when the application automatically identifies that an entity performed an event, for example, entry or exit from a location, interaction with another entity, access to a file, among others.

Figure 1 shows the trail creation of a mobile device user. This user moves around the four regions. On the way, each star represents the occurrence of an event, which causes a *trailpoint*.

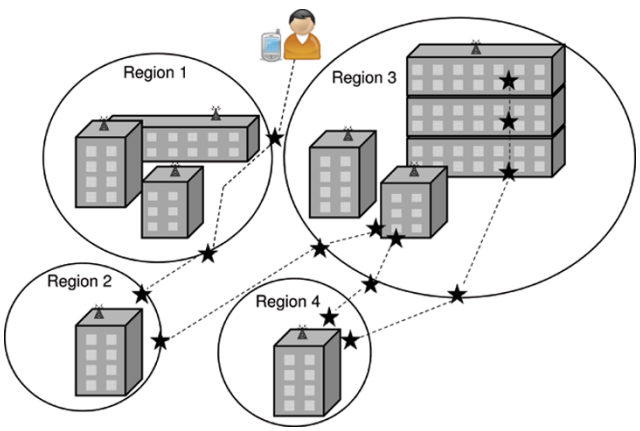
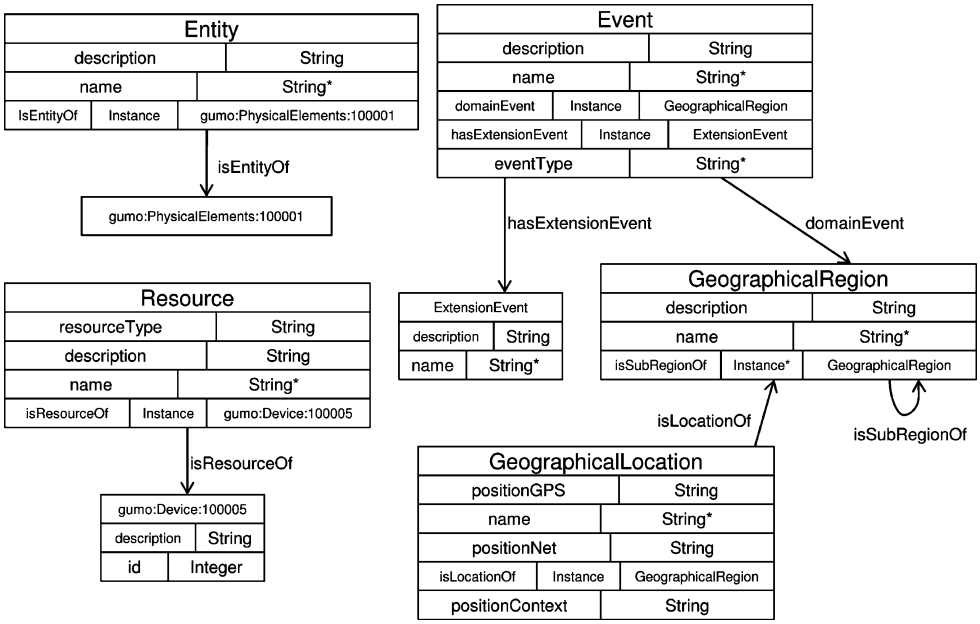


Fig. 1 Example of a trail composition



2.2 Ontology

Aiming at standardizing the trails information, we propose an ontology, which is presented in Fig. 2. The objective of a trail is to record the entities' movements and their activities. To satisfy the first requirement, the ontology provides the *GeographicalRegion* and the *GeographicalLocation* classes, which map the places where users can go. The former class identifies the geographical regions that the users can visit, whereas the later keeps the physical identification of these regions. Depending on the type of location technique used, an instance of the GeographicalRegion class can have many instances of the GeographicalLocation class referring to it. For instance, if a region is mapped through GPS coordinates, it would be necessary many coordinates to represent it. On the contrary, it would be difficult to identify the region. The model supports three location forms: (1) by satellites coordinates (see the *positionGPS* attribute of the *GeographicalLocation* class); (2) by network identification, using the host's IP (supported by the *positionNET* attribute); (3) and by contexts sent by a location system (represented by the *positionContext* attribute).

Aiming at supporting the description of entity's possible activities, the ontology provides the *Event* and the *ExtensionEvent* classes. The *Event* represents the activities that entities can do. This class has the *domainEvent* attribute, which enables to restrict the regions that an event can occur. The *ExtensionEvent* describes the additional fields that an event supports, thus enabling to register context data. For example, the file reading event could have a context information indicating the format of the file.

Beyond the entity's movements and actions, the trails must record information about the entity and the resources they are using. Hence, the ontology provides the *Entity* and the *Resource* classes, which extend the *PhysicalElements* and *Device* classes proposed by the UbisWorld project [20]. The UbisWorld provides a set of ontologies that represent various elements of the real world.

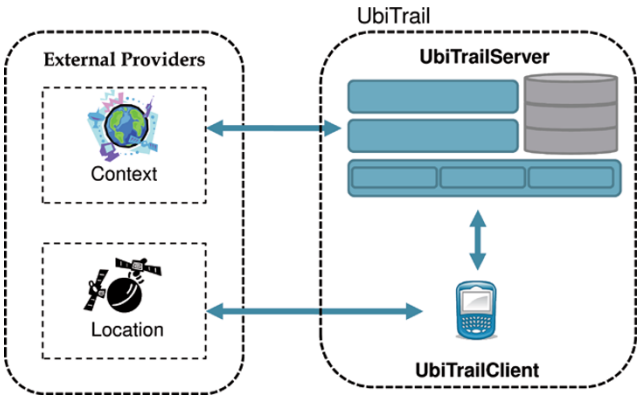


Fig. 3 The UbiTrail architecture

2.3 The model architecture

2.3.1 General organization

Figure 3 presents a general view of the model, which is composed of a server (UbiTrailServer) and a client (UbiTrailClient). UbiTrailServer supports the trails management and provides services for the applications. UbiTrailClient runs on mobile devices and supports the communication between the applications and the server. In addition, UbiTrail considers the existence of two external providers that provide context and location information. Next sections

describe all these components, justifying why they are necessary.

2.3.2 External providers

There are many different ways to acquire location data in mobile computing [21]. For instance, using measurements taken within cellular networks, the following methods have been proposed [43]: *received signal strength* (RSS); *time of arrival* (TOA); *angle of arrival* (AOA); and *enhanced observed time difference* (E-OTD). On the other hand, using an external positioning system, there is the satellite-based Global Positioning System technology [11]. Furthermore, other location techniques have been studied, such as response rate of Bluetooth inquiries [9] and triangulation of Wi-Fi antennas [10]. There are still studies on the hybrid use of technologies to obtain more precise location and to optimize the power consumption [18, 48].

In the same way, there are many different approaches on how to acquire context information [5]. Chen et al. [12] presents three different methodologies (*Direct sensor access*, *Middleware infrastructure*, and *Context server*). Differently, Winograd [45] introduces three different context management models (*Widgets*, *Networked services*, and *Blackboard model*). For this reason, it is necessary to separate the detection and the use of context data to improve the extensibility and reusability of systems [5]. In the same way, this strategy could be applied to location.

In UbiTrail, we separated from the model the obtainment of context and location information. This task is performed by external providers, which must be attached to the model. The external location provider must be periodically accessed by UbiTrailClient due to the entities' mobility. The time of obtainment of the location data can be set according to the application's necessity. UbiTrailClient obtains the location and sets the value of the *location* attribute of the *ptrail*.

Many authors describe the "context" word according to their own definitions of what context actually includes. However, a popular form to classify context instances is to define different context dimensions [5]. Gustavsen [19] and Prekop and Burnett [33] name these dimensions *external* and *internal*. In the same sense, Hofer et al. [24] refer to *physical* and *logical* context. The *external* or *physical* dimension refers to context that can be measured by hardware sensors (for example, sound, light, touch, temperature, and air pressure). Differently, the *internal* or *logical* dimension is mainly described by users or obtained by monitoring their interactions (for example, the user's goals, tasks, work context, business processes, and emotional state).

Due to the two context dimensions, UbiTrailServer was designed to receive context data from two different sources. The first is the UbiTrailClient, which provides information related to the internal or logical dimension. As this type of data is related to the entity's interactions, we had to choose the component closer to the entities. Thus, as UbiTrailClient runs on the entities devices, it was the chose component. UbiTrailClient obtains data from the application and sets the value of the *extension* attribute of the *ptrail*.

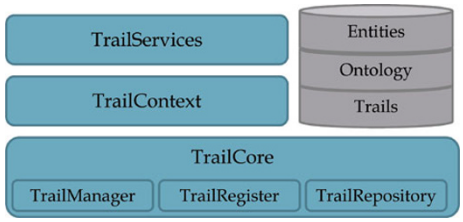


Fig. 4 The UbiTrailServer architecture

The second source is the context external provider, which provides information related to the external or physical dimension. This kind of data is related to the physical area around the entity. Consequently, entities within the same location can have the same context information, such as, light, vehicles flow, and weather. Hence, it is not necessary that each entity sends information of its physical context. On the contrary, this data could be captured by only one component, thus being available for all entities within the same location. UbiTrailServer is the component responsible for this task. It receives the *ptrail* sent by UbiTrailClient and analyses whether is necessary to obtain physical context information. Being necessary, it consults the context external provider and complements the received *ptrail* before records it.

2.3.3 UbiTrailServer

Some of the UbiTrailServer basic tasks are to manage the entities' trails and to enable application querying the trails. To perform these functions, we designed the **TrailServices** module (see Fig. 4, which presents the UbiTrailServer architecture). The *TrailServices* provide some *basic services* to manage the trails and, moreover, enables applications building their own *specialized services* according to their needs (see Fig. 5).

The *basic services* perform generic tasks which are considered strategic for the management of the server and the trails. These services are organized into three groups: *TrailControl*, *TrailCompose*, and *TrailQuery*. The *TrailControl* group supports the services used for the control and management of the applications, such as, the entities' authentication and the management of the server settings. The *TrailCompose* group supports the services used for the trails composition. In this group some services stand out, such as: (1) the register of entities for the trails management; (2) notification of the beginning or ending of a trail composition period; (3) the sending of *ptrails* in the *trailpoint* process; (4) the visibility change of a *ptrail*. The *TrailQuery* group contains

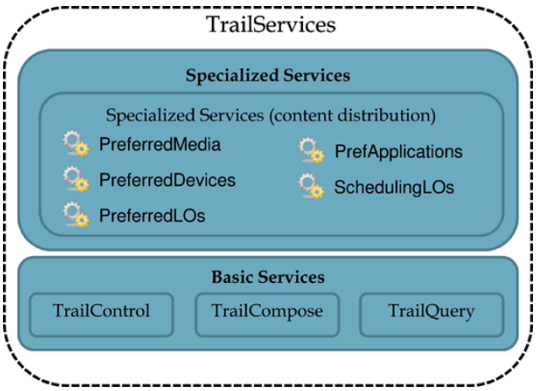


Fig. 5 Organization of the TrailServices layer

the services that allow queries related to the trails, such as: (1) quantity of *ptrails* in a trail (total or within a period); (2) parts of trails linked to a period (set of *ptrails*); (3) content of a specific *ptrail*.

UbiTrailServer enables applications adding to the model their own *specialized services*, which can be composed of one or more basic services. Section 3.2 describes five specialized services created to support content distribution guided by trails. These services are presented in Fig. 5.

As discussed in Sect. 2.3.2, UbiTrailServer is in charge for the physical context information, i.e. it has to communicate with the context external provider. To perform this task, we propose the **TrailContext** component. When the *TrailServices* component receives a *ptrail*, it analysis if the event of the *ptrail* has associated a physical context information. Case it has, the *TrailServices* calls the *TrailContext* to obtain the data. Due to the different approaches on how to acquire context information [5], we delegated the *TrailContext* specification to those who will apply UbiTrail in a specific domain.

The server has three repositories to store information of the entities, the ontology, and the trails. The entities repository stores identification data of the entities related to the trails that are been managed. The ontology repository stores the standard terms used for the regions, locations, events, extension of events, entities, and resources. The trails repository stores the *ptrails* that compose the entities' trails.

Some services can have common functions, for example, to query the ontology and to manage the trails. For this reason, we designed the **TrailCore** component, which performs basic operations that can be used by the services. The TrailCore is divided into three subcomponents: the *TrailManager*, the *TrailRegister*, and the *TrailRepository*. The *TrailManager* manages the trails, mainly consulting the ontology repository and organizing the data in the access to the trails repository. One of its main tasks is the composition of the record for the storage of a *ptrail* at the *trailpoint* solicitation moment. The *TrailRegister* module manages the

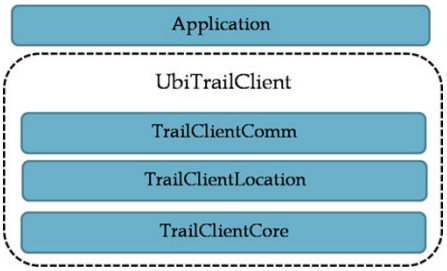


Fig. 6 The UbiTrailClient architecture

entities' register as well as their access in the server, using mainly the entities repository. The *TrailRepository* provides an interface to access the repositories, thus supporting the *TrailManager* and the *TrailRegister*.

2.3.4 UbiTrailClient

Figure 6 presents the UbiTrailClient architecture. This component runs on mobile devices and has three basic tasks: (1) to communicate with UbiTrailServer; (2) to obtain the entities' location; (3) and to capture the events made by the entities. Thus, UbiTrailClient is divided into three components. The **TrailClientCore** module has internally a *ptrail* structure, which is set and sent to the server whenever occurs an event. This component constantly monitors the entity's behaviors within the application. When the entity performs an event, TrailClientCore obtains its location, through the **TrailClientLocation** component, and sends the *ptrail* instance to the server through the **TrailClientComm** module. The TrailClientLocation consults the location external provider to obtain the location data.

The final entities do not interact directly with UbiTrailClient. They only interact with the applications that are using UbiTrail to manage their trails. UbiTrailClient monitor those applications, sending events information to UbiTrailServer.

3 Content distribution

This section presents the strategy used in the content distribution guided by trails. Section 3.1 describes the ubiquitous learning environment, in which UbiTrail was integrated. Section 3.2 describes how the trails were used in the selection process of Learning Objects (LOs).

3.1 The ubiquitous learning environment

There are many ubiquitous learning systems, for example, Japelas [32] supports the teaching of treatment expressions in the Japanese language according to the learner's context. It is a system designed specifically to that purpose and does

not support the automatic distribution of content. The Selene [34] and the Elena [40] projects consider user's profiles and learning objects in distributed environments, but they do not take into account location data. The SmartClassroom [47] and the AmbientWood [35] systems use location information, but they do not consider profiles and learning objects. The LOCAL model [7, 8] uses these three resources together, i.e., it distribute content guided by the users' profiles and their locations. In the same way, GlobalEdu [6] supports profiles, location information, and learning objects. Nevertheless, GlobalEdu needs of a middleware (for example, ISAM [2, 3]) to perform the main services of ubiquitous computing. LOCAL does not requires complement of a middleware, i.e., the model is self-sufficient.

Initially, LOCAL (Location and Context Aware Learning) was organized into six components and supported only basics functionalities [7]. Thenceforth, it was improved through the inclusion of support to the treatment of events as well as the improvement of the User Profiles and the Learning Objects Repository subsystems [8]. Therefore, the environment became to be composed of seven components (Fig. 7): (1) **User Profiles**—stores learner's specific information; (2) **Personal Assistant** (PA)—runs on the learner's mobile device; (3) **Location System**—determines the physical location of the mobile devices; (4) **Learning Objects Repository**—stores and indexes the content related to the pedagogical process; (5) **Communication System**—establishes the communication with the learners; (6) **Event System**—schedules tasks; (7) **Tutor**—is an analysis engine that makes inferences based on the users' profiles and their locations.

The content distribution depends on its representation, selection, and availability [7]. The representation, in LOCAL, is based on a metadata, which categorizes the learning objects according to the IEEE/LOM standard [26]. There are five categories for content representation. The **General** category groups general information which describes the objects. The **Technical** category groups their requirements and technical features. The **Educational** category groups the educational and pedagogical features. The **Rights** category groups the proprietary rights and the objects terms of use. The **Relations** category supports the description of relationships among objects, allowing, for example, the definition of prerequisite among objects. The selection of the learning objects is made by Tutor using the learner's location and profile and the Communication System in charge for providing the selected objects to the users.

3.2 Content selection using trails

3.2.1 Specialized services in UbiTrail

The content distribution guided by trails was based on specialized services created in UbiTrail and accessed by Tutor

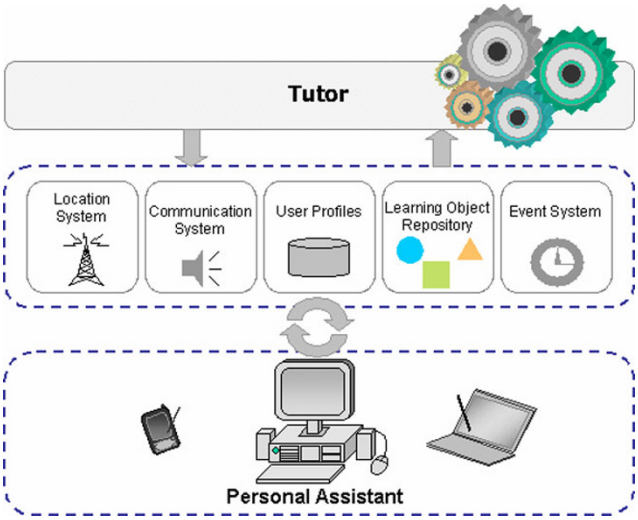


Fig. 7 The LOCAL architecture [8]

Table 1 Specialized services created in UbiTrail

Name	Description
PreferredMedia	Returns a list with the learner's preferred media in a context (for example, text, video, or audio).
PreferredDevices	Returns a list with the learner's preferred devices in a context (for example, iPAQ or desktop).
PrefApplications	Returns a list with the learner's preferred applications in a context (for example, Adobe Reader or Microsoft Word).
SchedulingLOs	Returns a list containing time suggestions to access an LO to the learner.
PreferredLOs	Returns a list with the LOs most accessed by all learners in a context.

of LOCAL. Table 1 summarizes the created services. The information returned by the services is obtained through the analysis of the learners' trails. The preferences are determined by the use frequency of media, devices, and applications in a context.

3.2.2 Dynamic and contextualized profile

The LOCAL profile model follows the PAPI standard [25] expanded with the learning styles proposed by Felder and Silverman [17]. The profile main information (*Contact*, *Preferences*, *Interests*, and *Styles*) is static, i.e., it does not change and follows the user independently of the visited context. The *Contact* section stores the users' basic information (name, address, e-mail, and phone). The *Preferences* section stores their preferences related to devices, applications, and media (for example, video, audio, or text). The information about *Interests* follows the ACM knowledge

area [1]. Finally, the *Style* section contains the user's learning style [17]. Tutor manages the LOs distribution to the learners according to their profile and still considering the available objects in each context.

The UbiTrail/LOCAL integration uses the trails to manage the profiles dynamically and contextually. Using the first three services provided by UbiTrail (see Table 1), Tutor dynamically determines the learner's preferences in the context that is being visited. The history of visits in the contexts, recorded in the learner's trail, enables Tutor to determine the learner's contextualized and updated preferences, because they could have changed while the learner interacted with LOs. The integration of UbiTrail does not replace the learner's profile kept by LOCAL. Tutor uses the trails and the profiles together, as described in next section. Furthermore, we did not create any new storage structure for the profiles, because the UbiTrail services use the trails stored in the *UbiTrailServer* repository to run the specialized services.

3.2.3 The UbiTrail/LOCAL integration

The UbiTrail/LOCAL integration was based on the following changes in LOCAL: (1) we included the support to the UbiTrail services in the Personal Assistant; (2) the Tutor inferences became to consider the learners' profiles and the services available by UbiTrail (described in Table 1).

Figure 8 presents the dynamic of the integration. In LOCAL, the learning objects are provided to the learners according to the opportunities that emerge during their changes of contexts. Location System informs to Tutor the context in which the learner is (step 1). Tutor uses this information, combined with the learner's profile, to determine the relevant objects to the learner (step 2). In this stage, Tutor uses the UbiTrail services to improve its inferences. Tutor notifies the learner of the available objects in the context through Communication System (step 3). The learner receives a message informing the availability (step 4). This contextualized distribution cycle can be started by two events: (1) the learner's context change or (2) the insertion of new material in the objects repository (in this case, the system only performs the last three steps).

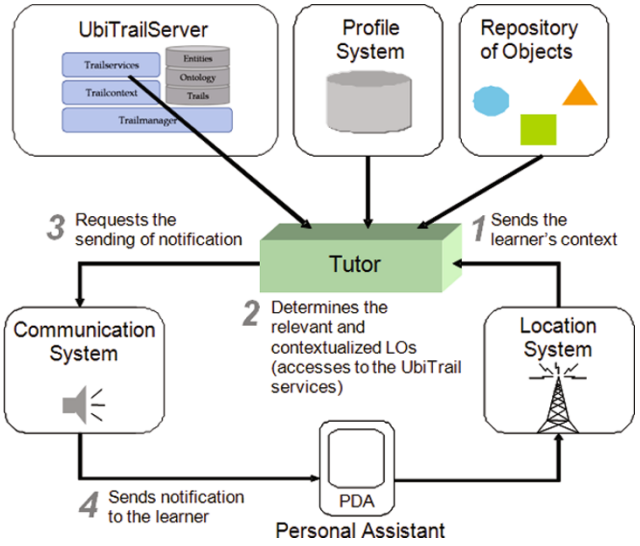


Fig. 8 Distribution of LOs guided by trails

4 Prototype and case study

This section describes the UbiTrail and the LOCAL implementations (Sect. 4.1). Furthermore, it discuss the case study (Sect. 4.2) used to validate the model.

4.1 The UbiTrail and the LOCAL prototypes

The UbiTrailServer prototype was implemented through the Java programming language. The JENA framework [27] was used to access the ontology represented in OWL. The services of the *TrailServices* layer are available through Web services.

The LOCAL prototype [8] was implemented through the C# programming language. The Tutor component was designed as a Windows service. The Personal Assistant module was developed with .NET Compact Framework and runs on HP iPAQs 4700 with Windows Mobile. The additional components provide their services to the Tutor and to the Personal Assistant through Web services. Moreover, to capture Wi-Fi signals, we used the OpenNETCF library.

The integration between the UbiTrail and the LOCAL prototypes was mainly based on changes made in Tutor in order to access the services provided by the Web services of *TrailServices*. The UbiTrailClient prototype was developed through the addition of new functionalities in Personal Assistant.

The LOCAL profile system was implemented using MySQL. The information is registered by the users using a Web page created in PHP and AJAX. This Web page can be accessed through either mobile devices or desktop computers. The metadata that describes the LOs are stored in XML, using the LOM standard [26]. Currently, the repository does not support the objects storage. The metadata contains the Web addresses in which the objects are stored.

The LOCAL location system [36] is based on triangulation of Wi-Fi antennas. Personal Assistant obtains the antennas' powers and forwards to Location System through a Web service. The system uses this information to infer the mobile device location. The locations must be previously registered on the system database. The mobile device location is inferred through a comparison between the data sent by Personal Assistant and the data previously captured through samples.

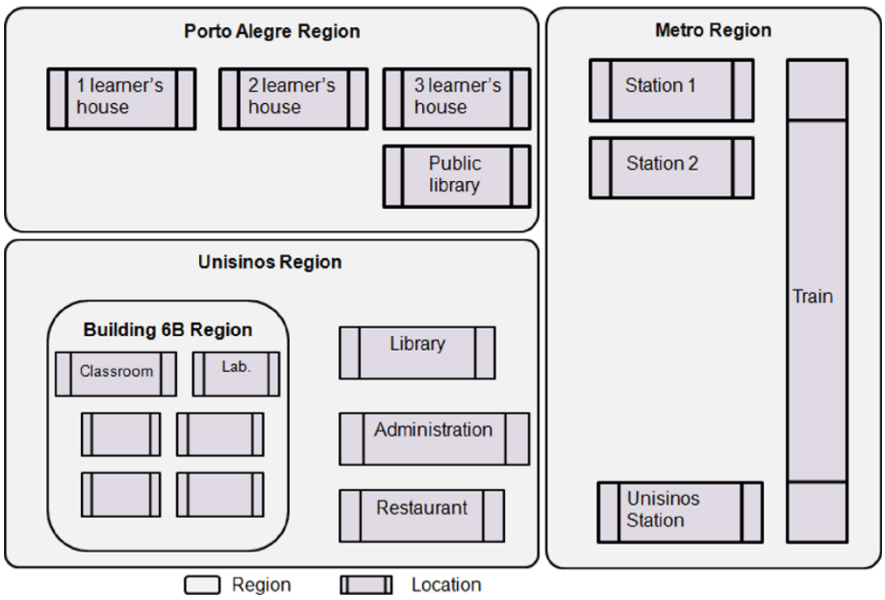


Fig. 9 Map of the simulation environment

4.2 Case study

4.2.1 Obtainment of the trails

The case study was based on simulated generation of learners' trails, which were used in the real scenario described in the validation environment (Sect. 4.2.2). The choice of simulation was based on the following factors:

- Interest in the generation of trails for many regions with wide geographical distribution (involving Porto Alegre and São Leopoldo cities in the south of Brazil);
- Interest in the generation of trails that covered at least one semester of the Computer Engineering[1] course at University of the Sinos Valley (Unisinos);[2]
- The impossibility of the students use the mobile devices outside the University campus, which is based in São Leopoldo city.

Figure 9 presents the simulation environment used in the trails generation, which is composed of the *Porto Alegre*, the *Metro*, and the *Unisinos* regions. We defined these regions because we wanted to monitor the learners' activities while they were moving from their houses to the university. Within the regions, we mapped some location where the users could go, for instance, the public library, the restaurant, and station 1.

The *Metro* region represents the area in which the students move from their houses to the university campus and vice versa. This region contains a location for each one of the three stations and a location that represents the student's presence on the train. Therefore, the system can record the events made by the learner during the movement between the stations.

The third region corresponds to the university campus. Within the university region, we defined a subregion representing the rooms of the Building 6B, in which is based the Computer Engineering course. These rooms were organized into six locations. Among them, the classroom and the laboratory stand out.

The regions and locations of the environment were implemented through instances of the *GeographicalRegion* and the *GeographicalLocation* classes of the UbiTrail ontology. A region can be composed of one or more subregions, but in both cases the subregions are instances of the *GeographicalRegion* class. Each region or subregion contains at least one location with the values of its geographical position.

The simulation involved five students of the Programming Techniques[3] discipline during the second semester of 2008 of the Computer Engineering course. The students' profile information was registered in LOCAL, focusing their preferences, possible learning styles, and interest related to the studied discipline (using the ACM knowledge areas [1]).

The trails generation was based on the possible activities (events) made by the learners during the second semester of 2008, which covers the period from August 1 to December 23. The trails generation was simulated using an application developed specifically for this purpose.

Considering the schedule of theoretical classes (presence in the *classroom*) and practical activities (presence in the

[1] Reference Ungraduate Course in Computer Engineering http://www.unisinos.br/graduacao/bacharelado/eng_comp.

[2] University of the Sinos Valley (Unisinos) http://www.unisinos.br.

[3] Homepage of the Programming Techniques discipline http://www.inf.unisinos.br/~barbosa/grefe/pa2.htm.

Table 2 Quantity of records generated in the simulation

Learner	Change of location	Device use	Application use	Access to LOs	Total
João	609	783	348	1479	3219
Leonardo	480	512	110	598	1700
Vicente	345	223	211	466	1245
Rodrigo	231	167	89	433	920
Douglas	126	122	33	542	823
Total of records					**7907**

laboratory), the application generated records related to the events of entry and exit from the locations. The learners João and Leonardo still had records of movements from their houses (*1 learner's house* and *2 learner's house*) to Unisinos, passing through the locations related to the Metro (*Station 1, Station 2, Train,* and *Unisinos Station*).

During the generation of the records related to the movement events, the application created registers of devices and applications use based on the contents scheduled for the Programming Techniques discipline. In addition, the application generated records of learning objects access.

Table 2 presents the quantity of event records generated during the simulation. Table 3 presents an example of these records, which are used in the trails composition.

4.2.2 *Validation environment*

Figure 10 presents the map of the second floor of the Building 6B at Unisinos. On this floor are based the MobiLab[4] laboratory and the graduate course in Computer Engineering. The validation scenario is composed of nine rooms, in which we installed four wireless antennas Cisco Aironet 1100. Location System of LOCAL [36] is based on the strategy of antennas triangulation and allows the exactly determination of the rooms in which the users are. In the experiment, we used iPAQs 4700. The equipments used in the development and validation of the prototype were donated by HP Computers.[5]

In the evaluation of the UbiTrail/LOCAL prototype, the rooms presented in Fig. 10 were mapped to the simulation environment presented in Fig. 9. Therefore, we established a relationship between the real environment (Fig. 10) and the simulated environment (Fig. 9), according to the presented in Fig. 11. The rooms 206 and 215 were mapped to the representations of the locations *train* and *1 learner's house*. The other rooms were considered the university's real rooms in Building 6B.

[4]Laboratory of Research and Development in Mobile Computing http://www.inf.unisinos.br/~mobilab.

[5]Prize awarded to MobiLab—"Grant HP Mobile Technology for Teaching 2005—Latin American Region".

Table 3 Examples of the records

Learner	Device	Event	Location	Extension	Date
João	iPAQ	Entry into location	Classroom	–	Aug. 4 20:01
Leonardo	iPAQ	Entry into location	Classroom	–	Aug. 4 20:12
Vicente	Desk.	Entry into location	Laboratory	–	Aug. 6 14:32
Vicente	Desk.	Exit from location	Laboratory	–	Aug. 6 15:31
Vicente	Desk.	Login device	Laboratory	–	Aug. 7 16:55
Rodrigo	Desk.	Login appli.	Laboratory	Microsoft Media Player	Aug. 7 19:22
Rodrigo	Desk.	Logoff appli.	Laboratory	Microsoft Media Player	Aug. 7 19:33
Rodrigo	Desk.	Logoff device	Laboratory	–	Aug. 7 19:55
João	iPAQ	Access to LO	Classroom	File: PDF Media: Text Subj.: Interfaces	Aug. 18 8:25
Leonardo	Desk.	Access to LO	Laboratory	File: MPEG Media: Video Subj.: Networks	Aug. 18 14:13
Vicente	iPAQ	Access to LO	Laboratory	File: MP3 Media: Audio Subject: Java	Aug. 25 07:45

Therefore, during the prototype evaluation, the movements among the real rooms represented virtual movements. For example, a movement from room 214 (classroom) to room 206 (train) represented the learner's presence on the train, moving from Unisinos to Porto Alegre. The presence in the virtual representation *train* makes Tutor initiates the analysis process of educational opportunities, as though the learner actually was physically in the train location.

4.2.3 *Validation through a scenario*

The scientific community has been using scenarios to validate context-aware environments (according to Dey's approach [14]) and ubiquitous environments (according to Satyanarayanan [38]).

Following this strategy, we created a scenario for tests and evaluations of the UbiTrail/LOCAL integration, involving the daily routine of a Computer Engineering student.

Fig. 10 Map of the validation environment

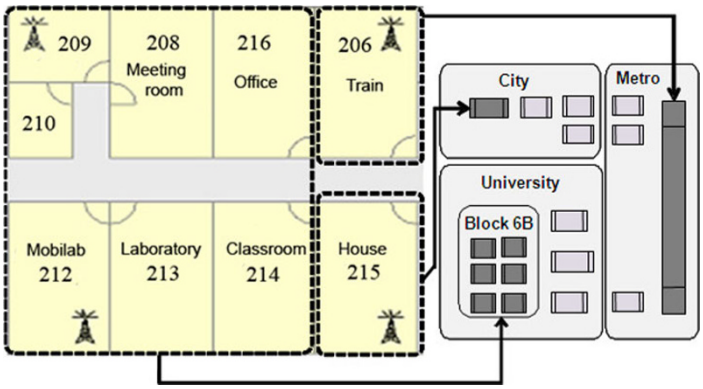


Fig. 11 Validation scenario

This scenario focuses on the distribution of LOs in different contexts based on the learner's trail, according to the described below:

"Jorge is professor of the Programming Techniques discipline of the Computer Engineering course at Unisinos. At the beginning of the week scheduled to the basic teaching of programming using the Java language, he registered in the LOCAL repository a group of learning objects (LOs) that support different preferences of media (for example, text, video, and audio). The register involved the indication of the contexts in which the objects would be available.

In the classroom, Jorge registered some curricular objects concerning the basic concepts. In the laboratory, he registered optional objects that will be available for improvement, involving different themes (such as "Interfaces" and "Parallel Programming").

João is a Computer Engineering student. At the day of the Programming Techniques class, João entered the classroom and logged in the LOCAL system (see Fig. 12(a)*), subsequently he received a notification of the availability of a curricular LO. João accessed the object and received a PDF file (preference text). LOCAL used the* ***PreferredMedia*** *service of UbiTrail to determine this preference, because João already had many classes and was used to use textual objects. Objects of all preferences of media were available, i.e., João could have accessed an object related to other preferences, fact that would be recorded in his trail and used in the dynamic determination of media preferences in that context.*

After the class, João entered the laboratory and received a notification regarding the availability of a LO related to the subject of his interest in Java (in this case, "Interfaces"). This object had been used by all students who were interested in Java. Thus, LOCAL, using the ***PreferredLOs*** *service, recommended it to João.*

In the laboratory, João preferred audio objects, because he could listen to them while he was using the computer (determined by the ***PreferredMedia*** *service). João accessed the object and, after its use, received another notification of available object (see Fig.* 12(b)*) concerning the subject of his interest (this time, the object was about "Parallel Programming").*

Always that the students receive a notification of LOs, they can delay the access. This action indicates that the students are interested in the object, but they do not want to access it at the moment, i.e., they want to access the object at the first opportunity of available time. João postponed the access to the optional object (see Fig. 12(c)*).*

At the end of the day, João moved to the train station. When he entered the train, LOCAL notified him that a postponed LO was identified (Fig. 12*(d)). LOCAL used the*

Table 4 Dynamic of the validation scenario

Character	Action
Professor	Registers the learning objects in LOCAL, relating them to contexts.
Learner	Enters the classroom and logs in LOCAL using an iPAQ 4700; see Fig. 12(a)
UbiTrail/LOCAL	LOCAL detects the entry of the learner into the classroom. Tutor uses the *PreferredMedia* service to determine the learner's preference of media in the context. Subsequently, it requests the sending of a message about the LO available in the preferred media.
Learner	Accesses the curricular object.
Learner	Moves to the laboratory.
UbiTrail/LOCAL	LOCAL detects the entry of the learner into the laboratory and uses the *PreferredLOs* service to verify the available LOs, related to the student's interest, that had many accesses in that context. Tutor still uses the *PreferredMedia* service to define the learner's preferred media in the context. Finally, Tutor requests the sending of messages about the available LOs; see Fig. 12(b).
Learner	Accesses one of the two identified objects and postpones the access to the other to the first moment of available time (see Fig. 12(c)).
Learner	Moves to the train.
UbiTrail/LOCAL	LOCAL detects the entry of the student into the train and calls the *SchedulingLOs* service to verify the LOs that can be accessed during the trip. LOCAL identifies the postponed object and notifies the learner, see Fig. 12(d).
Learner	Accesses the object that has been postponed.
Professor	Registers a new optional LO concerning "Polymorphism" in Java that can be accessed in any context.
Learner	Enters his home.
UbiTrail/LOCAL	LOCAL informs the learner of the new optional LO available that corresponds to his areas of interest.
Learner	Accesses the new object.

SchedulingLOs *service to define the sending. Getting home, João still received another notification saying that the professor had provided an additional object to a subject of his interest ("Polymorphism" in Java). In the register, the professor informed that the interested students should be notified independently of their locations."*

The scenario was tested in the validation environment described in Sect. 4.2.2. The João student carrying an iPAQ 4700 simulated the scenario, moving around the environment rooms. The train and the learner's house were simulated through the rooms 206 and 215. Table 4 summaries the actions, emphasizing the character involved in each one.

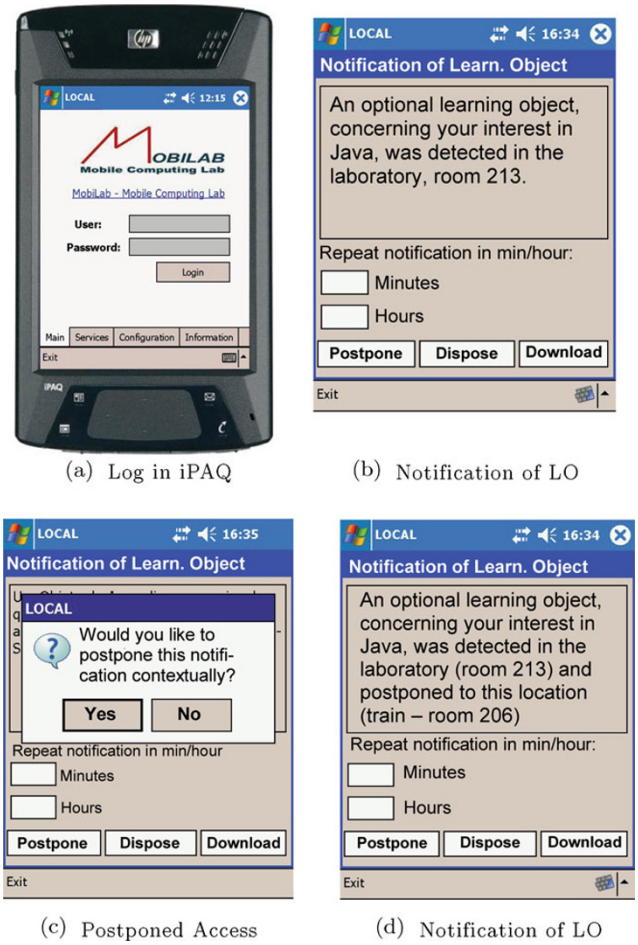


Fig. 12 The scenario snapshots

5 Related works

Hermes [16] is a framework for creating applications that involve the movement of users during the performance of actions in different locations. In Hermes, a trail consists in the route suggested to the users accomplish their tasks efficiently. The proposal does not consider the record of activities in contexts, as proposed by UbiTrail. Hermes does not support the distribution of content either.

Levene and Peterson (Trail Records [28]) propose the use of a mobile device to record learning experiences using different media (for instance, text, video, and audio) while visiting different locations. The sequence of records forms a structure named trail, which can be used to recover the learner's experience in the visited path. Although Trail Records links content with locations, it does not contain any approach to support content distribution guided by the contexts. Furthermore, the model is not generic, focusing specifically on the record of experiences with media.

PELEP [29] proposes the automatic improvement of the learner's profile in ubiquitous learning environments. The model records the learner's activities during a period and,

Table 5 Comparison among the proposals

Criterion	Hermes	Trail records	PELEP	Life annotation	UbiTrail
History of contexts	No	Yes	No	Yes	Yes
Content distribution	No	No	Yes	No	Yes
Standardized register	Yes	No	Yes	No	Yes
Automatic generation	Yes	Yes	Yes	Yes	Yes
Register of context	No	Yes	Yes	Yes	Yes
Generality	Yes	No	No	Yes	Yes

based on these records, updates their preferences constantly. PELEP does not record the learners' history in contexts, i.e., it does not support the management of trails. Furthermore, the model focuses on ubiquitous learning specifically.

Life Annotation [41, 42] proposes the annotation of the locations visited by mobile devices users, composing a trail of the visited locations. In addition, the model proposes the record of the much information as possible related to each context. The objective is to record a user's activities, allowing future requests. The proposal allows generics requests related to the trails, but it does not contain any mechanism enabling content distribution. Furthermore, Smith proposes as a future work of his doctoral dissertation [42] the use of ontology to standardize the trails information.

Table 5 shows the comparison among the proposals, including the UbiTrail. The table is based on the following criteria: (1) **history of contexts** indicates if the proposal stores the history of visited contexts; (2) **content distribution** indicates the existence of some mechanism that supports content distribution based on trails; (3) **standardized register** indicates if it is used some strategy to standardize the trails information; (4) **automatic generation** indicates if the trails information is generated automatically; (5) **register of context** indicates if the proposal register context information in trails; (6) **generality** indicates if the model supports different applications.

Based on the bibliographic review of this article, we can affirm that the UbiTrail/LOCAL integration creates the first environment supporting content distribution guided by trails. Furthermore, among the researched works, UbiTrail is the only model that standardizes the information through an ontology, and also the only model capable of managing trails of any interested mobile entity.

6 Conclusion

This article has proposed a model to manage trails and its application in the content distribution in ubiquitous learning environments. Although UbiTrail initially has been integrated with LOCAL, its proposal is generic enough to be applied to other systems that might use trails for improvement. The use of trails allows the ubiquitous learning systems to act more effectively, because they can use more precise information of the learners' behaviors.

The objective of the case study was to demonstrate that is viable to integrate ubiquitous systems with UbiTrail as well as to use the trails to improve automatic decisions, in our case, content distribution. Nonetheless, the UbiTrail/LOCAL integration is limited, because the specific services created to improve content distribution consider basically only technical aspects, such as most used applications, devices, and media. There is a service that considers a temporal characteristic (SchedulingLOs) and another that takes into account a social aspect (PreferredLOs), however, they are very simple. The PreferredLOs service suggests the learning objects most accessed by all learners within a specific context. This service could be improved, becoming to take into consideration the learners with similar trails. In addition, this service could use the time that the users spend reading a learning object, for instance, it could attribute higher importance to the objects read for more than five minutes than to those just opened and closed.

Although we have created only one service related to social aspect, UbiTrail supports any other. All that is necessary to develop a service is to have the correspondent information recorded in the trails. Thus, firstly we have to define the desired social events in the ontology (for example, interactions, tasks, activities). After that, it is necessary to develop the UbiTrailClient, which is responsible for monitoring the entities' activities and composing the trails. So, the application can develop the desired service and add it in UbiTrailServer.

Currently, the *trailpoint* process, i.e., the sending the *ptrail* to the server, occurs whenever the entity performs an event. If an entity does too much events in a short period of time, depending on the network bandwidth, it can influence on the network speed. Hence, it could be interesting to keep the entities' trails in the mobile device for some time, for example, the activities performed during the day could be sent to the server only in the end of the day. In this way, the network speed can also be influenced, but it would be only during the synchronization between the client and the server. So, we believe that would be important to carry out further studies in this perspective.

The main conclusions were the following: (1) the use of trails confirms the mobile computing potential to improve the content distribution; (2) the availability of the history

of visited contexts stimulates the use of mobile devices as instruments to access contextualized contents; (3) UbiTrail supports content distribution guided by trails; (4) the prototype and the experiment attest the proposal feasibility.

UbiTrail as well as its integration with LOCAL is an initial proposal. The following activities will allow the continuity of the study: (1) we will run additional tests with the trails obtained from simulation, for example, the *PreferredDevices* and *PrefApplications* services were not used yet; (2) the heuristic to determine the preferences in the contexts (frequency of use) can be improved through additional information, for example, time of each use; (3) the services performance involving a significant amount of records must be evaluated, because the experiment considered a few number; (4) the ideal evaluation will involve the monitoring of users to generate real trails, during a period that allows the evaluation of their behaviors, according to the Smith's approach that involved one user during 2 years [41, 42].

References

1. ACM Computing Classification System (2010) http://www.acm.org/class/1998. Accessed April 2010
2. Augustin I, Yamin AC, Barbosa JLV, Geyer CFR (2002) ISAM: a software architecture for adaptive and distributed mobile applications. In: VII IEEE symposium on computers and communications, Messina, pp 333–339
3. Augustin I, Yamin AC, Barbosa J, Silva LC, Real R, Geyer C (2004) ISAM, joing context-awareness and mobility to building pervasive applications. In: Mobile computing handbook, New York, US, pp 73–94
4. Al-Mekhlafi K, Hu X, Zheng Z (2009) An approach to context-aware mobile Chinese language learning for foreign students. In: ICMB'09: proceedings of the 2009 eighth international conference on mobile business, pp 340–346
5. Baldauf M, Dustdar S, Rosenberg F (2007) A survey on context-aware systems. Int J Ad Hoc Ubiquitous Comput 2:263–277
6. Barbosa DNF, Geyer CFR, Barbosa JLV (2005) GlobalEdu: an architecture to support learning in a pervasive computing environment. In: IFIP workshop on educational technology (EDUTECH), Perth, Australia, pp 1–10
7. Barbosa JLV, Hahn RM, Rabello SA, Barbosa DNF (2007) Content distribution in context-aware environments. In: XIII Brazilian symposium on multimedia and the Web (WebMedia), Gramado, RS, Brazil, pp 1–8
8. Barbosa JLV, Hahn RM, Rabello SA, Barbosa DNF (2008) LOCAL: a model geared towards ubiquitous learning. In: Proceedings of the 39th SIGCSE technical symposium on computer science education, Portland, OR, USA. ACM Press, New York, pp 432–436
9. Bargh M, Groote R (2008) Indoor localization based on response rate of bluetooth inquiries. In: International conference on mobile computing and networking, San Francisco, California, USA, pp 49–54
10. Brunato M, Battiti R (2005) Statistical learning theory for location fingerprinting in wireless LANs. Comput Netw 47:825–845
11. Buluso N, Heidemann J, Estrin D (2000) GPS-less low cost outdoor localization for very small devices. IEEE Pers Commun Mag 7:28–34
12. Chen H, Finin T, Joshi A (2003) An intelligent broker for context-aware systems. In: Adjunct proceedings of UbiComp, pp 183–184
13. Dey AK, Hightower J, Lara E, Davies N (2010) Location-based services. IEEE Pervasive Comput 9:11–12
14. Dey AK, Salber D, Abowd GD (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware application. Hum Comput Interact 16:97–166
15. Diaz A, Merino P, Rivas FJ (2009) Mobile application profiling for connected mobile devices. IEEE Pervasive Comput 9:54–61
16. Driver C, Clarke S (2008) An application framework for mobile, context-aware trails. Pervasive Mob Comput 4:719–736
17. Felder RM, Silverman LK (1988) Learning and teaching styles in engineering education. Eng Educ 78:674–681
18. Guillemette M, Fontaine I, Caron C (2008) Hybrid RFID-GPS real-time location system for human resources: development, impacts and perspectives. In: HICSS Hawaii international conference on system sciences, p 406
19. Gustavsen RM (2002) Condor—an application framework for mobility-based context-aware applications. In: Proceedings of the workshop on concepts and models for ubiquitous computing, Goeteborg, Sweden
20. Heckmann D, Loskyll M, Math R, Recktenwald P, Stahl C (2009) UbisWorld 3.0: a semantic tool set for ubiquitous user modeling. In: User modeling, adaptation, and personalization (UMAP), Saarbrücken, Germany
21. Hightower J, Gaetano B (2001) Location systems for ubiquitous computing. Computer 34:57–66
22. Hightower J, LaMarca A, Smith I (2006) Practical lessons from place lab. IEEE Pervasive Comput 5:32–39
23. Hoareau C, Satoh I (2009) Modeling and processing information for context-aware computing: a survey. New Gener Comput 27:177–196
24. Hofer T, Schwinger W, Pichler M, Leonhartsberger G, Altmann J (2002) Context-awareness on mobile devices—the hydrogen approach. In: Proceedings of the 36th annual Hawaii international conference on system sciences, pp 292–302
25. IEEE LTSC 1484.2 (2010) Draft standard for learning technology—public and private information (PAPI) for learners (PAPI learner). http://www.cen-ltso.net/Users/main.aspx?put=230. Accessed April 2010
26. IEEE/LTSC/LOM (2010) Draft standard for learning object metadata. ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf. Accessed April 2010
27. JENA (2010) a semantic web framework for Java. http://jena.sourceforge.net. Accessed April 2010
28. Levene M, Peterson D (2002) Trail record and ampliative learning. Research Report BBKCS-02-01, School of Computer Science and Information Systems, University of London
29. Levis D, Barbosa JLV, Pinto SCCS, Barbosa DNF (2008) Automatic improvement of the learner's profile in ubiquitous learning environments. Braz J Inform Educ 16:29–41
30. Lewis M, Nino C, Rosa J, Barbosa J, Barbosa D (2010) A management model of learning objects in a ubiquitous learning environment. In: IEEE international workshop on pervasivE learning (PerEL), Baden-Württemberg, Mannheim, Germany, pp 256–261
31. Ogata H, Yano Y (2003) How ubiquitous computing can support language learning. In: Proceedings of KEST, pp 1–6
32. Ogata H, Yin C, Yano Y, Oishi Y (2005) JAPELAS: supporting Japanese polite expressions using PDA(s). In: Towards ubiquitous learning
33. Prekop P, Burnett M (2003) Activities, context and ubiquitous computing. Comput Commun 26:1168–1176. Special issue on ubiquitous computing
34. Rigaux P, Spyratos N (2010) SeLeNe report: metadata management and learning object composition in a self eLearning network. http://www.dcs.bbk.ac.uk/selene/reports. Accessed April 2010

35. Rogers Y, Price S, Randell C, Fraser DS, Weal M, Fitzpatrick G (2005) Ubi-learning integrates indoor and outdoor experiences. Commun ACM 48:55–59
36. Rolim CR, Sonntag N, Barbosa JLV (2008) HLS: model for development of applications sensitive to location. In: IX workshop in high-performance computing systems, Campo Grande, Mato Grosso do Sul, Brazil, pp 227–234
37. Sangkeun L, Sungchan P, Sang-goo L (2009) A study on issues in context-aware systems based on a survey and service scenarios. In: 10th ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing, pp 8–13
38. Satyanarayanan M (2001) Pervasive computing: vision and challenges. IEEE Pers Commun 8:10–17
39. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. IEEE Pervasive Comput 8:14–23
40. Simon B, Miklós Z, Nejdl W, Sintek M (2003) Elena: a mediation infrastructure for educational services. In: WWW conference, Budapest, Hungary
41. Smith AD, Hall W, Glaser H, Carr LA (2006) Towards truly ubiquitous life annotation. In: Memories for life colloquium. The British Library, London, England
42. Smith AD (2008) Who controls the past controls the future—life annotation in principle and practice. Ph.D. thesis, University of Southampton
43. Türkyilmaz O, Alagöz F, Gür G, Tugcu T (2008) Environment-aware location estimation in cellular networks. EURASIP J Adv Signal Process 139:1–9
44. Vaughan-Nichols SJ (2009) Will mobile computing's future be location, location, and location? Computer 42:14–17
45. Winograd T (2001) Architectures for context. Hum Comput Interact J (HCI) 16:401–419
46. Yang SJH, Chen IYL (2006) Providing context aware learning services to learners with portable devices. In: Proceedings of the sixth IEEE international conference on advanced learning technologies, pp 840–842
47. Yau SS, Gupta SKS, Gupta EKS, Karim F, Ahamed SI, Wang Y, Wang B (2003) Smart classroom: enhancing collaborative learning using pervasive computing technology. In: II American society of engineering education, pp 13633–13642
48. Zou D, Deng Z, Xu L, Ren W (2008) Seamless LBS based on the integration of WSN and GPS. In: ISCSCT international symposium on computer science and computational technology, pp 91–96