Journal of Applied Computing Research, 5(1):1-16
January-June 2016
Unisinos - doi: 10.4013/jacr.2016.51.01

MaPS: A framework to aid the development of collaborative applications for ubiquitous environments

Daniel de Souza Martins, Cassia Nino, Jorge Barbosa
Universidade do Vale do Rio dos Sinos. Av. Unisinos, 950, Cristo Rei, 93022-750, São Leopoldo, RS, Brasil
danieldesouzamartins@gmail.com, cnino@unisinos.br, jbarbosa@unisinos.br

Débora Barbosa
Universidade Feevale. RS-239, 2755, Vila Nova, 93352-000, Novo Hamburgo, RS, Brasil
deboranice@feevale.br

Abstract. A research topic of growing interest is the convergence between Collaborative Systems and Ubiquitous Computing, where context awareness is becoming a tool for enhancing collaboration processes. The application of Ubiquitous Computing concepts in the improvement of collaboration strategies created a research front called Ubiquitous Collaboration. This article proposes a framework to aid the development of collaborative applications for ubiquitous environments, called MaPS. MaPS works at one relevant stage of the collaboration. It uses context information and user profiles to improve the search for peers and the selection of communication channels. The article proposes the framework, its requirements and its architecture. Moreover, we describe a prototype and two applications which were developed with it. The framework was evaluated considering software development, based on the experience got in the implementation of the applications and aspects of functionalities. It was made through a scenario involving active participants. The results of both evaluations show the potential for using MaPS.

Keywords: collaboration, collaborative applications, ubiquitous computing, ubiquitous collaboration, ubiquitous environments, context awareness.

Introduction

Nowadays, studies on mobility in distributed systems have been stimulated by the proliferation of portable electronic devices (for example, smartphones, tablet PCs and notebooks) and the use of interconnection technologies based on wireless communication (such as WiMAX, WiFi and bluetooth). This mobile and distributed paradigm is called *Mobile Computing* (Díaz *et al*., 2010; Satyanarayanan *et al*., 2009). Moreover, the improvement and proliferation of *Location Systems* (Hightower *et al*., 2006) have motivated the adoption of solutions that consider the user's precise location for the provision of services – Location-based Services (Dey *et al*., 2010). The adoption of these technologies combined with the diffusion of sensors enabled the availability of computational services in specific contexts – *Context-aware Computing* (Bellavista *et al*., 2012; Knappmeyer *et al*., 2013). The idea consists in the perception of characteristics related to the users and their surroundings. These characteristics are normally referred to as context, i.e. any information that can be used to describe the circumstances concerning an entity. Based on perceived context, the application can modify its behavior. This process, in which software modifies itself according to sensed data, is named *Adaptation* (Lopes *et al*., 2012). In this scenario, the *Ubiquitous Computing* initially introduced by Weiser (1991) and Satyanarayanan (2001) is becoming reality (Costa *et al*., 2008; Caceres and Friday, 2012).

The application of Ubiquitous Computing concepts in the improvement of collaboration strategies created a research front called *Ubiquitous Collaboration* (Izadi *et al*., 2002; Laso-Ballesteros, 2006; Farshchian and Divitini, 2010; Caceres and Friday, 2012). There are proposals to support the development of ubiquitous collaborative applications as discussed in *Ubiquitous collaboration*. These proposals span from conceptual frameworks to service platforms. Nonetheless, the selection of peers and communication channels as an initial step to foster Ubiquitous Collaboration is still an open research topic.

This article proposes the MaPS framework, which stands for *Matching People to Share*. MaPS aims at aiding the development of collaborative applications for ubiquitous environments. It focuses on the search for peers and for communication channels to foster interactions among participants. In this sense, MaPS uses context information (Bellavista *et al*., 2012; Knappmeyer *et al*., 2013) and user profiles (Wagner *et al*., 2014) to improve the results of searches. In this work we evaluated MaPS through the development of two prototypes that used the framework in their implementation.

The article is composed of six sections. *Ubiquitous collaboration* presents the main concepts of Ubiquitous Collaboration. *Related works* discusses seven related works that support collaboration in ubiquitous environments. A comparison between these works serves as the basis to discuss the contributions of MaPS. *Evaluation* describes MaPS, discussing its requirements, architecture, and prototype. The fifth section (*Conclusions*) approaches the evaluation of the framework mainly based on the implementation of two applications. Finally, the last section presents final remarks and directions for future works.

Ubiquitous collaboration

Groupware applications can be defined as "computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment" (Ellis *et al*., 1991). In this type of application, communication is a key element, because interactions among people are constant. Computers also have an important role, because computing provides several technologies to support interactions. As observed by Stahl (2002), the computer's potential as a support tool tends to be greater when applied to groups rather than individuals. Communication among members of groups has always suffered several restrictions. Nonetheless, these restrictions can now be reduced through computational support.

In this sense, Ubiquitous Computing is a computational model that aims at satisfying the user's needs pro-actively, acting in an invisible way (in the background) (Weiser, 1991). In addition, this model aims at providing continuous integration between technology and environment, assisting users in daily tasks. In the Ubiquitous Computing concept, applications are available anywhere and anytime and the access to networks is continuous and independent of devices. Systems developed considering this model can reconfigure themselves dynamically, adapting to the users' contexts (Lopes *et al*., 2012). Moreover, invisibility (Weiser, 1993) guarantees that users access computational resources seamlessly, causing the use of computers to be an intuitive activity.

Ubiquitous Collaboration (Izadi *et al*., 2002; Laso-Ballesteros, 2006; Farshchian and Divitini, 2010; Caceres and Friday, 2012) explores the Ubiquitous Computing technologies to foster collaborative work. In Ubiquitous Collaboration, the mobility of users and the perception of elements around them (i.e. their contexts) are part of the collaboration process. While users are moving with their mobile devices, the system dynamically supports collaboration processes, using opportunities provided by the users' contexts to improve their collaboration experiences.

The collaboration process can be divided into three steps (Zhang *et al*., 2005): (i) *Find People* – participants search for available users who are knowledgeable about the area of their interests; (ii) *Communication among Participants* – users must be able to communicate with other participants, so communication should not be restricted to a predefined set of tools; on the contrary, it should be supported regardless of the employed technology; (iii) *Collaboration* – participants, considering the person to communicate with and the communication mediums available, start the collaboration process. Collaboration can be described as a discussion, an exchange of ideas or computer files, or even corrections in a document. Expanding this idea, the 3C collaboration model determines that computer-supported collaboration processes are performed by *Communication*, *Coordination*, and *Cooperation* (Fuks *et al*., 2007). This model proposes that when participants are collaborating, they should talk to each other (Communication), organize themselves (Coordination), and work together in a shared space (Cooperation) (Gerosa *et al*., 2003).

The first step for collaboration is to know which people are available and how to find them. After this, users can interact and form groups. This process (search and selection of users) is critical for collaboration (Zhang and Jin, 2005; Yang and Chen, 2008). Another critical point in these elementary steps is the type of communication channel employed (Fuks *et al*., 2007). Ubiquitous Computing with its vision of the user's mobility and invisibility acts as a catalyst for these collaborative steps (Weiser, 1991). Mobile devices and contextual adaptation can improve the quality of search results, helping the users to access the resources that they may need (Satyanarayanan, 2001; Caceres and Friday, 2012). Ubiquitous Computing creates new opportunities of interaction and collaboration, especially among people who would not meet otherwise (Divitini *et al*., 2004; Farshchian and Divitini, 2010).

Related works

uLearning focuses on searching for people for collaboration processes (Zhang *et al*., 2005; Jin *et al*., 2005). Nonetheless, it does not address the problem of selecting communication channels. This work presents a conceptual model and describes an application that implements the proposed concepts. However, uLearning does not provide an API or a library for developers, which makes its reuse less in different scenarios.

Chen and Yang (Chen and Yang, 2006; Yang and Chen, 2008) propose a system that has the functionality of searching for peers based on peer-to-peer (P2P) and social networks. This proposal attaches great importance to the concept of searching for resources, which can be either digital contents or individuals. Although the system is designed for ubiquitous environments, it does not consider context information, namely, it does not use available resources in the environment to improve the search results.

UbiCollab (Divitini *et al*., 2004; Farshchian and Divitini, 2005) and Collaborator (Bergenti *et al*., 2002, 2003) provide support for developers to build collaborative applications, supplying APIs or development environments. However, these works do not provide explicit support for the functionality of selecting people and communication channels, which is one relevant requirement to provide a setting conducive to collaboration.

UCAVE (Basu *et al*., 2012) is a ubiquitous collaborative activity virtual environment, which enables immersive virtual collaborations with minimal and portable infrastructure. According to the work, UCAVEs are portable immersive virtual environments that leverage mobile communication platforms, motion trackers and displays to facilitate ad-hoc virtual collaboration. With a focus on immersive environments, this work does not provide a framework for the development of collaborative applications using selected people and communication channel.

MUSIC (Hallsteinsen *et al*., 2012) is a software development framework for collaborative applications. The framework supports several adaptation mechanisms and offers a model-driven application development approach supported by MUSIC middleware. Therefore, MUSIC is a proposal that includes a framework, methodology, and execution platform. Although the work supports applications that select people and communication channels, these applications need to run in MUSIC platform.

Luna *et al*. (2015) propose a methodology based on ontologies to process user profiles and to represent the interactions process between the user and the different contexts that surround this user. As a case study, some user profiles and the context of a school are presented. The work does not provide a framework for the development of collaborative applications and does not support selecting people and communication channels.

Table 1 shows a comparison between the related works. The works classified as "Yes" have the analyzed characteristic. The works classified as "No" discuss the characteristic, but do not specify it. Finally, the classification "N/A" means that the work does not specify and does not discuss the characteristic. The aspects analyzed were the following:

- User's data model – it indicates whether the work uses user's information in any of its operations;
- Context information – it shows whether the work uses any type of context information to optimize its process;
- Communication channels – it indicates whether the proposal allows to add new forms of interaction when necessary;
- Mechanism to search for people – it shows whether the work allows performing search using filters to restrict the results;
- Communication channels in searches – it indicates whether the proposal uses communication channels to restrict the search results. This information can be obtained through a profile, in which users select the forms of interaction that they would

like to have, or even determining the type of device that the users are carrying;
- Libraries or services – it shows whether the work provides libraries or services for the development of collaborative applications;
- Extensibility – it indicates whether the work can be extended to other types of scenarios beyond the initially predicted ones;
- Environment – it indicates whether the work depends on a specific execution platform to support collaborative applications.

Besides the characteristics listed in Table 1, we highlight two additional characteristics that we consider a main contribution of the MaPS proposal. First, the works studied generally consider data explicitly provided by the users. Only MUSIC (Hallsteinsen *et al.*, 2012) and the ontology proposed by Luna *et al*. (2015) use a model for user profile. MaPS considers the user profiles, and also uses context information, i.e., MaPS considers aspects around the users to perform the searches. Second, MaPS proposes to foster the collaboration considering, in an integrated approach, communication channels, users' profiles and contexts. Although MUSIC supports a lot of collaboration characteristics, the proposal needs the specific execution platform.

Therefore, MaPS is a proposal that uses context information and users' profile to improve the search for peers and the selection of communication channels. As a framework, MaPS allows developing applications in different scenarios for collaboration in ubiquitous environments, without considering a specific execution platform.

The MaPS Framework

The next subsections describe the framework requirements, its architecture, and its prototype.

Requirements about the Target Environment

One of the goals of MaPS is the definition of a solution that allows its use in different scenarios for collaboration in ubiquitous environments. However, there are basic requirements that an environment must meet to support systems developed using the framework. These requirements include functions that are not covered by MaPS, but are necessary for its operation, such as: (i) a service to manage users' profiles (Wagner *et al*., 2014); (ii) a service to provide context information (Bellavista *et al*., 2012); (iii) and a lexical database service (Miller, 1995) with semantically related terms, which will be used to establish semantic equivalence or similarity with the terms used in the search process.

In MaPS, the user profile model is a hotspot, so it can be instantiated and customized for each application that uses the framework. Nonetheless, there is a set of basic information that must be covered by the profile schema used by the applications. The information necessary for the proper functioning of the framework is listed in Table 2.

MaPS aims at automating and optimizing the steps of selecting peers and communication channels, suggesting the most appropriate options for the users. The suggestions are based on information from the users' contexts.

Table 1. Comparison between the related works.

Proposal	User's data model	Context information	Communication channels	Mechanism to search for people	Communication channels in search	Libraries or services	Extensibility	Environment
uLearning	Yes	Yes	Yes	No	Yes	N/A	Yes	No
UbiCollab	Yes	Yes	Yes	No	N/A	Yes	Yes	No
P2P and Social Networks	Yes	N/A	N/A	Yes	N/A	N/A	Yes	No
Collaborator	Yes	Yes	Yes	N/A	N/A	Yes	Yes	No
UCAVE	Yes	Yes	No	No	No	Yes	No	Yes
MUSIC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Luna *et al*.	Yes	Yes	No	No	No	No	Yes	No

In a search for peers and communication channels, context elements of the involved users can be considered to refine the search. In this case, it is important to emphasize that the selection of the context elements is influenced by the scenario in which the framework is running. The context information service, which must belong to the environment, has the task of interpreting the context data and providing transparent access to the framework, as will be discussed in the next sections.

The MaPS architecture

Figure 1 presents the framework architecture. The construction of peer selection is made through three key concepts, which are the following:

- Profiles: it contains the user's information. This data is kept by the *Profile Repository* component;
- Matcher: it matches the users' profiles. The *Profile Matcher* component performs this task;
- Matching Criteria: it uses predefined operators to specify a search. These operators are managed through the *Criterion* elements of the *Requisition* component.

As described in the previous subsection, the environment must provide services to manage users' profiles, a lexical database, and to provide context information. The *Profile Repository*, the *Lexical Knowledge Base*, and the *Context Middleware* external components represent these services.

Table 2. Basic information of the user profile.

Information	Description
Name	Name through which the user will be recognized by other participants in the system.
Identification	Unique identifier used by the application. It is used to differentiate users.
Knowledge	Set of user's knowledge
Communication channels	List of communication channels available to the user.
Preferred communication channels	List of the user's preferred communication channels.

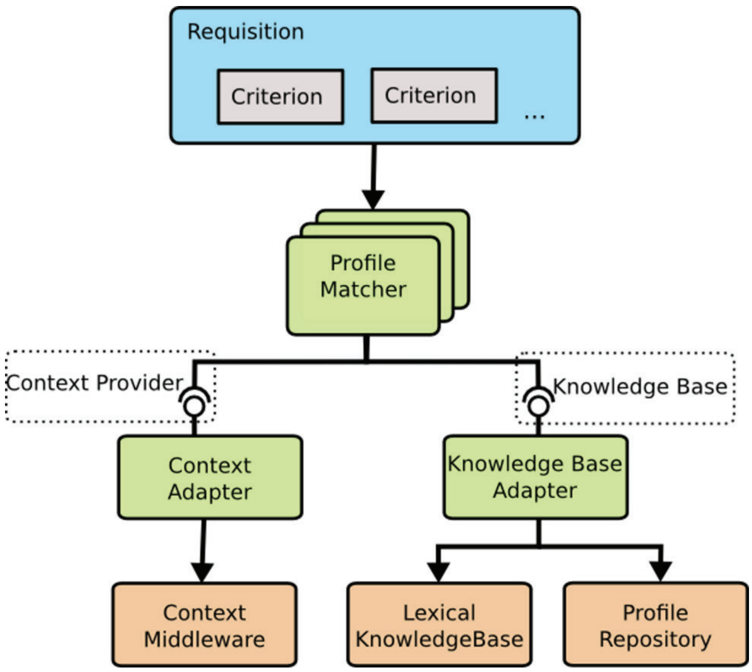

Figure 1. The MaPS architecture.

The *Profile Matcher* component uses information of the *Knowledge Base* interface, which represents an abstraction of the required repositories. These repositories centralize relevant data used in the searches for peers and communication channels. Information about the users' contexts is provided by the *Context Provider* interface. It is also used to improve the searches. The framework accesses the external services through the *Context Provider* and the *Knowledge Base* interfaces. These interfaces support the *Context Adapter* and the *Knowledge Base Adapter*, following the *Adapter* design pattern (Gamma *et al*., 1994).

The *Profile Matcher* component was built based on the *Strategy* design pattern (Gamma *et al*., 1994). This pattern defines a common interface for all heuristics of profile searches that match a given criterion, as presented in Figure 2. The *Simple Matcher* and the *Location-Aware Matcher* classes provide concrete strategies to search for profiles implementing the *Profile Matcher* interface. The relationship between the *Profile Matcher* and its implementations is marked with the “incomplete” stereotype of UML-F (Fontoura *et al*., 2000). This stereotype indicates that this is a hotspot and, thus, allows new implementations with different search algorithms. The reasons for enabling more than one algorithm to perform the same function are the following: (i) not all scenarios require the same information to perform a search; (ii) it is not efficient to have an algorithm to cover different scenarios; (iii) the information that we want is not always available.

Each implementation of the *Profile Matcher* contains a different searching heuristic. The use of a heuristic is determined by a specific configuration of the framework or by a state of the context. For example, if there is information about the user's location available, the algorithm can consider this data to select peers.

The MaPS framework allows the *Profile Matcher* implementations to be structured as a pipeline, enabling searches for profiles to be refined by various filters in an incremental manner. Figure 3 presents a *User Request* being mapped by the *User Agent*, which is a software agent that translates and forwards the user requests to the corresponding instance of *ProfileMatcher*. The agent searches for people who would satisfy a particular set of criteria. These criteria are transported through the pipeline and could be refined by the heuristics implemented at each stage. When the search reaches the final stage, that *Profile Matcher* instance accesses the *Knowledge Base* and returns a preliminary list of members who meet that potentially expanded set of criteria. The list flows in the opposite direction of the pipeline and can be filtered again at each stage to improve the quality of the results. For example, an intermediary step of the pipeline can classify the elements of the list according to an attribute of the context (for example, location) or can reduce the list of results because the user carries a device with limited resources.

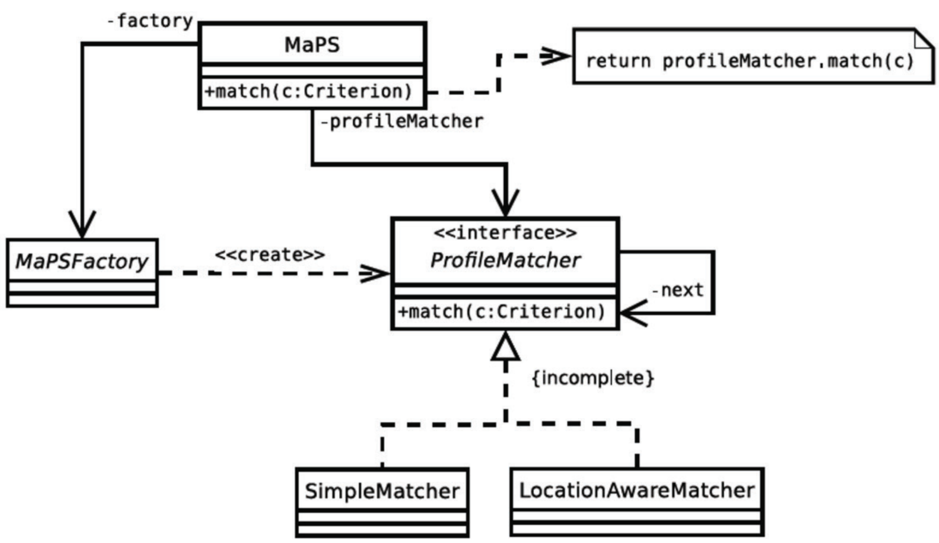

Figure 2. The ProfileMatcher class diagram.

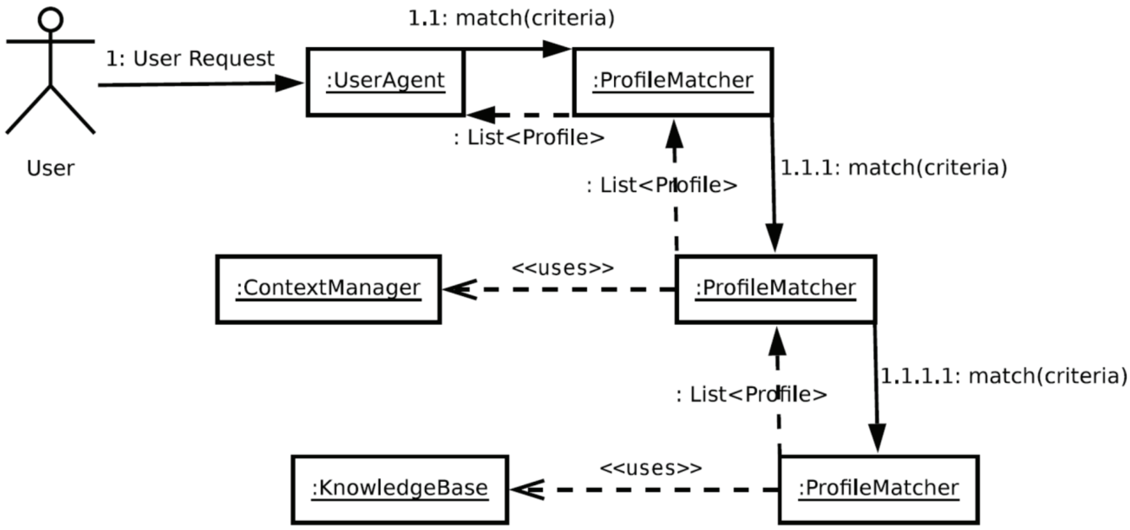

Figure 3. Pipeline of filters.

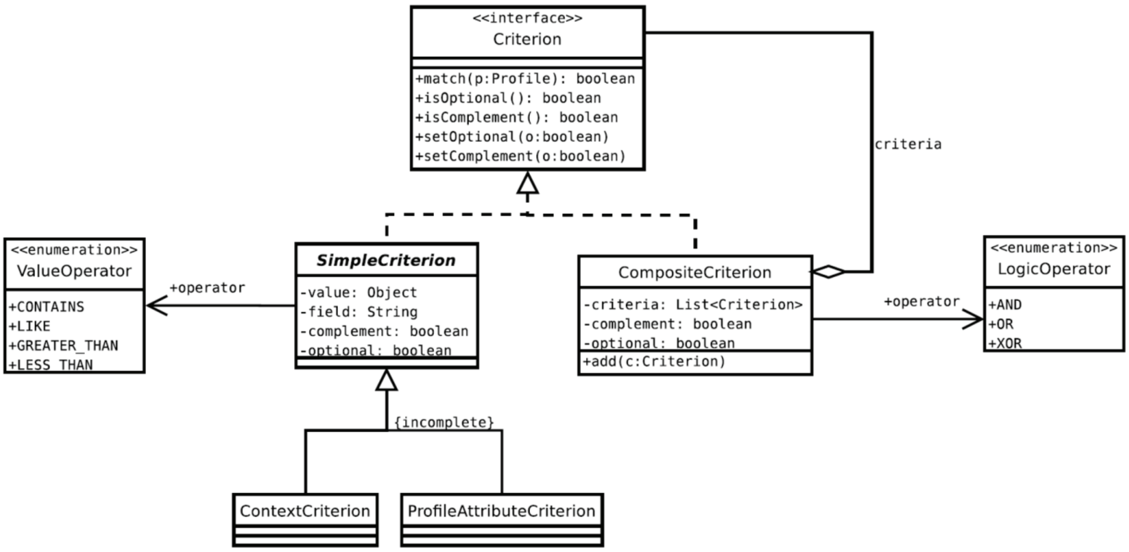

Figure 4. The Criterion class diagram.

The search is made according to criteria established by the user. The criteria are represented by the *Criterion* elements shown in Figure 1. They can be defined as simple or as compound. A simple criterion relates pairs of attributes and values, through operators, that must be satisfied in a query. A compound criterion represents a group of criteria that are combined to make a more specialized query. This definition of criteria was implemented using the *Composite* design pattern (Gamma *et al*., 1994) which is presented in Figure 4. The *Criterion* interface represents a criterion in its abstract form and the *SimpleCriterion* and the *CompositeCriterion* classes represent a simple criterion and a compound criterion.

The *SimpleCriterion* class uses values and operators. The operators are defined in the *ValueOperator* class and act as modifiers in the searches. There are four operators, which are the following: (1) *contains* – it matches the exact terms informed; (2) *like* – it searches for semantically similar terms; (3) *greater than* – it looks for values higher than the informed ones; (4) *less than* – it searches for values lower than the informed ones. The *CompositeCriterion* class organizes a group of criteria using three logical operators: (1) *and* – it indicates that all conditions must be met; (2) *or* – it signifies that at least one of the conditions must be met; (3) *xor* – it indicates that exactly one of the conditions must be met.

Prototype

The MaPS prototype was developed using the Java programming language. Figure 5 shows a diagram of the prototype main classes.

The *MaPS* class is based on the *Facade* design pattern (Gamma *et al*., 1994) which provides to the applications a point of unified access. The *MaPS* class contains references to the main classes used in the search process, i.e. the *ProfileMatcher*, the *KnowledgeBase*, and the *ContextProvider* classes. These references are obtained through the *getProfileMatcherChain()*, the *getKnowledgeBase()*, and the *getContextProvider()* methods. All these methods access the *MaPSFactory* class to create or to get instances of the classes. The MaPS class, through the match method, receives and forwards search requests to the *ProfileMatcher* class.

The *MaPSFactory* class is responsible for setting the framework according to the parameters of the *MaPS.properties* configuration file, which is presented in Figure 6. MaPS can run in local mode or in a client-server mode. The main difference between these modes is the capacity to build a pipeline of the *Profile Matcher* that has a part in the server and a part in the client. This configuration is set by the *maps.factory* parameter (line 3) of the configuration file. For a local operation, the parameter must be set to *DefaultFactory*. For a distributed operation, the following aspects must be observed: (i) both client and server must have an instance of the MaPS framework; (ii) the client application must have the *maps.factory* parameter set to *ClientFactory*; and, (iii) the application server must have the *maps.factory* parameter set to *ServerFactory*.

The *ClientFactory* configuration adds a stub as the last element in the *ProfileMatcher* pipeline. This stub encapsulates the aspects of network communication and is responsible for forwarding the client request to the server. The *ServerFactory* configuration makes the framework start the pipeline with an external request, which came from a client.

The configuration of the *ProfileMatcher* pipeline can be made through the configuration file. The *createProfileMatcherChain* method of the *MaPSFactory* class mounts the sequence of instances based on the *maps.profileMatcher* properties, following the order specified in the file. Valid values for these properties are classes that implement the *ProfileMatcher* interface. As shown in the diagram of Figure 2, the framework already has two classes that imple-

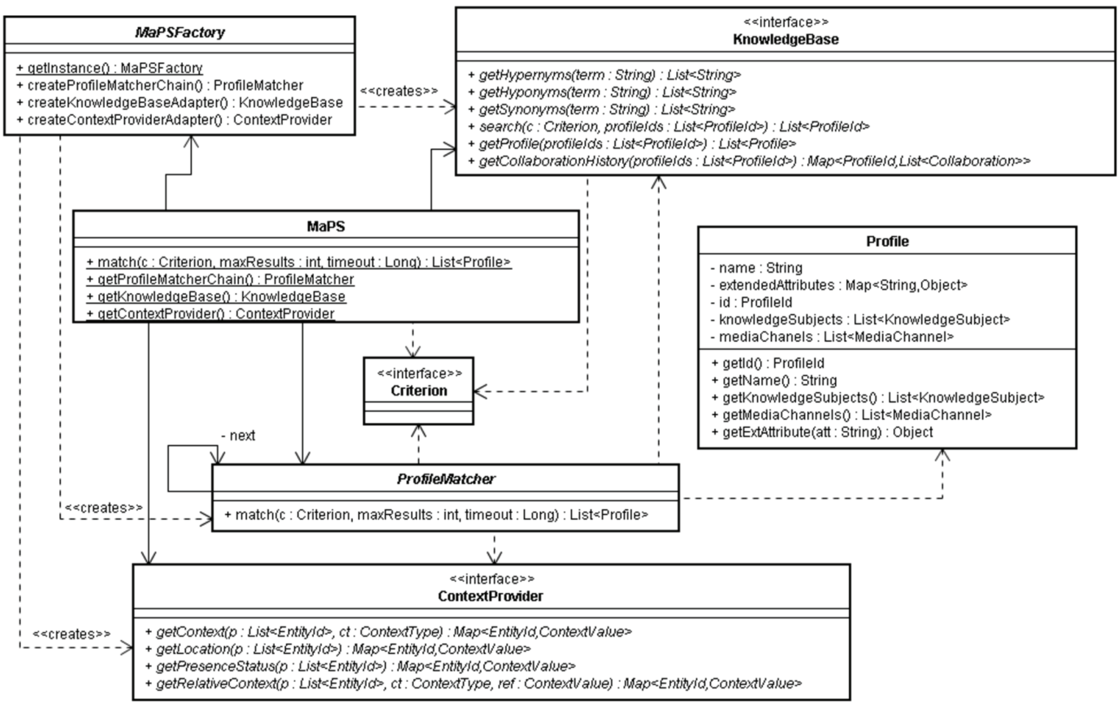

Figure 5. The prototype main class diagram.

```
# MaPS Settings
# Factory of the MaPS instance
maps.factory=maps.factory.DefaultFactory
#maps.factory=maps.factory.ClientFactory
#maps.factory=maps.factory.ServerFactory

## ProfileMatcher chain
maps.profileMatcher.0=maps.matcher.LocationAwareMatcher
maps.profileMatcher.1=maps.matcher.SimpleMatcher
#URL of the server to the client access
maps.remoteServer=10.0.0.1

## ContextAdapter options
maps.contextAdapter=

## KnowledgeBase options
maps.knowledgeBaseAdapter=
```

Figure 6. The MaPS configuration file.

ment different algorithms: *SimpleMatcher* and *LocationAwareMatcher*.

Moreover, the configuration file must have indicators of adapter classes (see Figure 1). These classes implement the *KnowledgeBase* and the *ContextProvider* interfaces to access external services.

Evaluation

MaPS is a framework characterized by being a semi-finished structure. According to Edwards *et al*. (2003), the evaluation of a framework is problematic, since it is not visible to the final users. The authors state in their work that it is only possible to evaluate the functionalities of a framework by building applications that use it and then evaluating these applications. In that way we can obtain an indirect evaluation of the framework.

Therefore, we decided to implement two applications using the framework. However, it is important to point out that when testing an application, the users will not only assess the framework, but the software as a whole. Thus, based on these prototypes, we evaluate MaPS from two perspectives: *Software Development* and *Framework Functionalities*. The following three subsections present the applications implemented and both perspectives used in the evaluation.

Applications developed using MaPS

Aiming at evaluating MaPS, we developed two applications using the framework: *Looking4U* and *PeopleFinder*. Both applications have the same requirements, which are shown in the use case diagram of Figure 7. However, the prototypes were implemented in different ways, i.e. they differed in the architecture and in the employed technology.

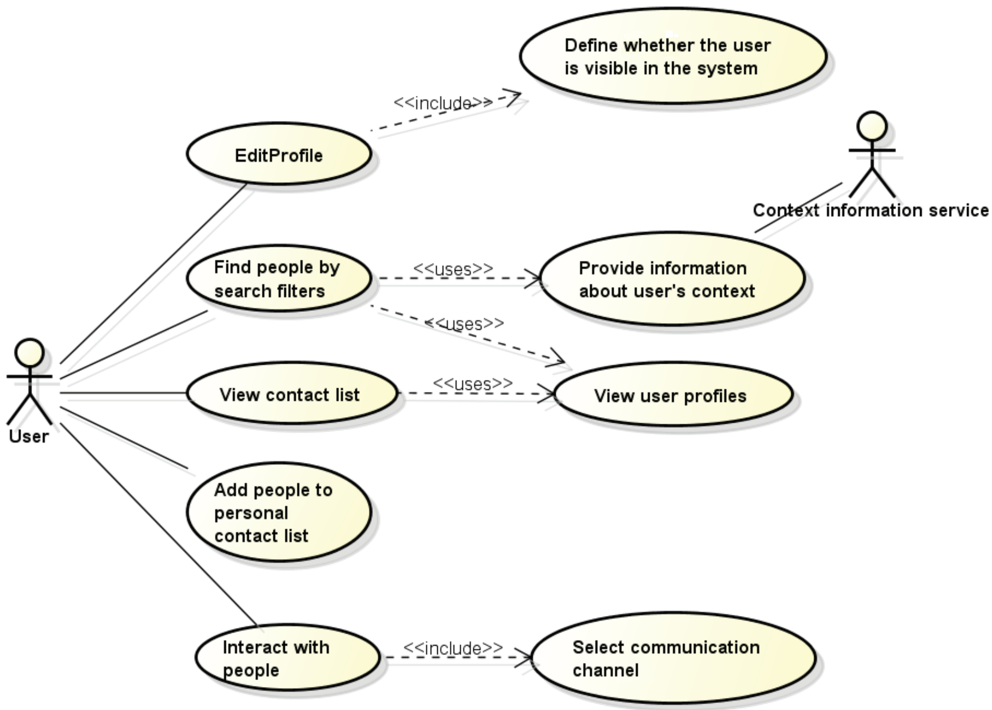

Figure 7. Use case diagram of the applications.

As shown in the diagram, the user's objective is to interact with people. They can communicate with anyone in the environment, even if the person is not in their contact list. To find people in the environment, the users define search filters. The searches made by the applications consider user profiles and context information, which are provided by external services. For the evaluation, the external services were simulated, generating pre-defined information.

Interactions among users can occur through different ways (for example, SMS, email, and phone). However, these interactions are not made by the applications. In the search results, the prototypes only report which communication channels are the best ones.

The ***Looking4U Application*** was developed using the Java programming language and the Eclipse IDE. We also used the JBoss application server, because it is a client-server application. The architecture of the application is shown in Figure 8. It includes a multi-level selection of profiles, because of the pipeline of the *ProfileMatcher* instances. The pipeline is divided between client and server. In this organization, users can use different types of devices (for example, desktops, notebooks, and smartphones). The client's devices run the client application, which is composed of the *UserAgent* and the *ProfileMatcher* components. The *UserAgent* is a software agent that manages the interactions between the client and the server.

In the server application, the *ContextProvider* and the *KnowledgeBase* are adapters that access external services. These services are determined by the environment where the framework is being used. The *ContextProvider* component abstracts the access to the context middleware. Figure 8 shows three possibilities of middleware which could be accessed by the application: LOCAL (Barbosa *et al*., 2011), EXEHDA (Yamin *et al*., 2003) and ContextToolkit (Salber *et al*., 1999). The *KnowledgeBase* component abstracts the access to the *Profile Repository* service and to the *Lexical Knowledge Base* service. A possible lexical database that could have been used is the one proposed in the WordNet project (Miller, 1995). Moreover, a possible profile management system that could have been integrated is the eProfile (Wagner *et al*., 2014). For testing purposes, the external services were simulated using Java classes within the application. As the framework encapsulates the access to these services, the use of a data simulator or a real service is not a relevant aspect in the evaluation.

The *Looking4U* architecture is based on a distributed solution, where both the client and the server have an active role in the searches for profiles. The configuration files used in this application reflect the architecture. They are presented in Figure 9 and Figure 10.

The factory property (line 3) of both files indicates the type of communication that will be used between the client and the server. In this case, the application is using the RMI technology. The *RMIClientFactory* and the *RMIServerFactory* classes are extensions of the *Client-*

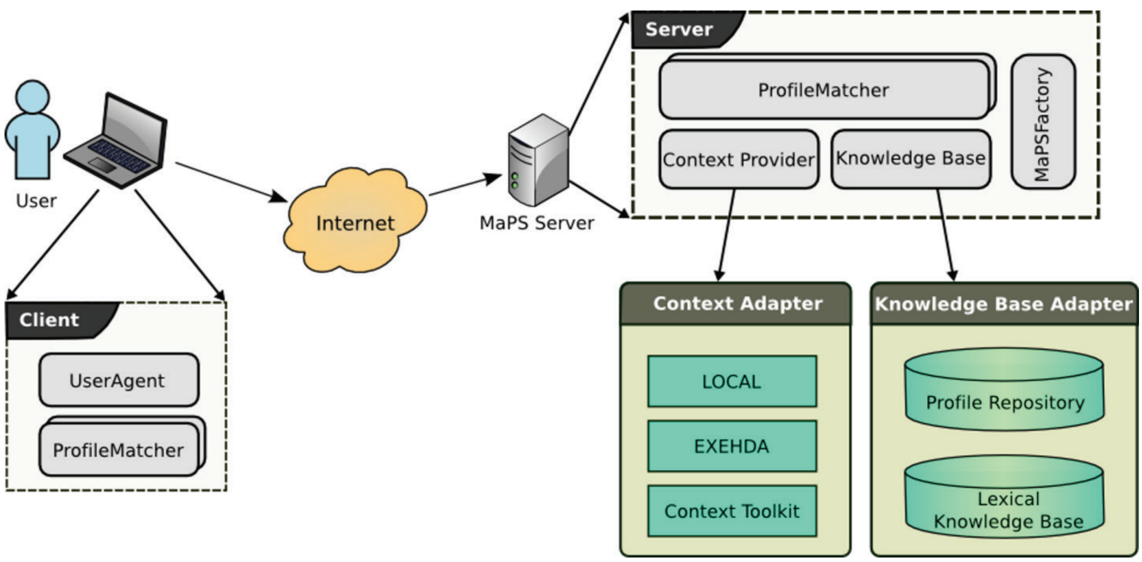

Figure 8. The Looking4U architecture.

```
# MaPS Settings - Configuration File
# Factory of the MaPS instance
maps.factory=maps.factory.RMIServerFactory

## ProfileMatcher chain
maps.profileMatcher.0=maps.matcher.LocationAwareMatcher
maps.profileMatcher.1=maps.matcher.SimpleMatcher

maps.contextAdapter=com.looking4u.ContextAdapter
maps.knowledgeBaseAdapter=com.looking4u.KnowledgeBase
```

Figure 9. Configuration file of the Looking4U server.

```
# MaPS Settings
#Factory of the MaPS instance
maps.factory=maps.factory.RMICLientFactory

## ProfileMatcher chain
maps.profileMatcher.0=com.looking4u.PresenceMatcher
maps.remoteServer=127.0.0.1

maps.contextAdapter=com.looking4u.ContextAdapter
maps.knowledgeBaseAdapter=com.looking4u.KnowledgeBase
```

Figure 10. Configuration file of the Looking4U client.

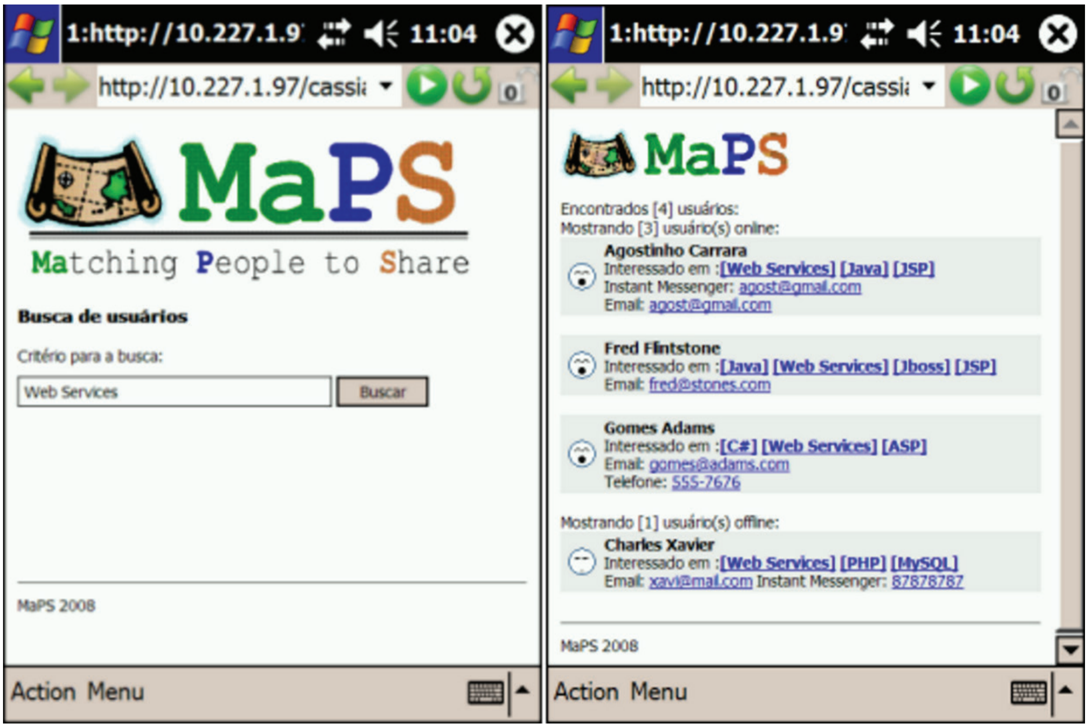

Figure 11. Screenshots of the PeopleFinder application.

Factory and the *ServerFactory* abstract classes, which extend the *MaPSFactory* class.

The second application implemented is called ***PeopleFinder***. Figure 11 shows the main page of the application and the page with the list of results of a search.

This is a web application that accesses a server through a web service. The server contains a Java application that uses an instance of the MaPS framework to offer a service to find people. The *PeopleFinder* was developed with the following technologies: the PHP programming language for the client application, the Java programming language for the server application, the JBoss application server for Java, the PHP server, and the Apache HTTP Server.

Differently from the first prototype, MaPS was used by this application in local form, i.e. the pipeline of filters was entirely contained in the server. In this case, the entire search is made by the server application; the ProfileMatcher chain is located only in the server.

Evaluation of the software development

The evaluation of the software development aims at assessing the framework through three different questions, which are the following:

(i) *Does the framework support software development with different architectures?* MaPS does not restrict the architecture of an application. Although the framework's architecture is defined, it can be adapted for different application architectures;

(ii) *Does the framework support the use of different user profiles and communica-*

tion channels? One of the functional requirements of MaPS is extensibility. Although the framework requires basic profile information, as described in *Applications developed using MaPS*, it does not restrict this support to a unique profile model. Therefore, it is possible to use MaPS in any application that uses profiles or that provides this information. Moreover, MaPS supports any media as communication channel, since it has information about its type (synchronous or asynchronous) and an identifier name;

(iii) *Is the framework customizable?* This question aims at evaluating the necessary efforts to extend MaPS considering different cases of usage.

The results of this evaluation are empirical, and they were originated from the experiences that the developers had during the implementation of the applications. Each application was developed with a different architecture. The development of *Looking4U* used a distributed architecture, i.e. dividing the pipeline of filters between the client and the server. For the *PeopleFinder* application, the approach used was different. Since this is a web application, all functionalities of searching for users were kept centralized in a server, and the application accessed them through a web service. Both applications used the MaPS framework, and their different architecture did not affect the functionalities of MaPS.

To establish the remote communication, the *Looking4U* used the RMI technology. The *factory* classes implement this type of communication. The *PeopleFinder* used the web service technology. Since the MaPS framework does not provide support for this technology, the developer had to develop the web service. However, it is important to note that the search results were not compromised by the implementation of the communication mechanism that was not provided by the framework; on the contrary, the search results worked in the same way.

The analysis made during the development of the applications showed that MaPS satisfies the requirements indicated in the three questions. First, the development of two applications with different architectures allows to answer positively the first question. Moreover, the applications treated a different management of user profiles. *Looking4U* used the *KnowledgeBase* component to simulate the access to an external *Profile Repository* service. *PeopleFinder* implemented the user profiles directly in the application. The communication channels were used by both applications to select the peers, although none of them supports the use of the channels. So, MaPS supported different user profiles and communication channels to select peers and to indicate the best channel. The second question can also be answered positively. Finally, answering the third question, it is possible to consider the framework customizable, because it was used to develop two different applications without a significant effort of development.

Evaluation of the framework functionalities

The evaluation of the framework functionalities aims at assessing the functionalities that MaPS must support in a collaborative application, i.e. the search for peers and communication channels. The evaluation was made through a test with 12 participants from a community of software developers in two phases.

In the first phase we presented to the participants a scenario based on an application implemented with the framework functionalities. This scenario helped the participants to understand the goal of MaPS. After this, the participants did an activity where they played the role of the search filter. The steps of this phase were the following:

(i) We presented to the participants a list of 16 fictional people (characters) with their profiles, composed of the current status (offline or online), knowledge areas and type of communication channels used;
(ii) Based on a scenario, the participants received a set of criteria that should be satisfied in the search for peers;
(iii) The participants indicated which characters should be returned in the search, explaining their decisions.

This test uses context information only about the character's status, but any other information could be used, for example, information about the character's location could be included in the search by creating a new context location criterion. The scenario presented to the participants in step 2 involved a person looking for a peer to collaborate with. This person used some criteria related to knowledge areas and communication channels to select the possible peers.

Table 3. Number of votes of the character G in each position of the rank.

Character	First position	Second position	Third position	Fourth position
G	3	5	4	0

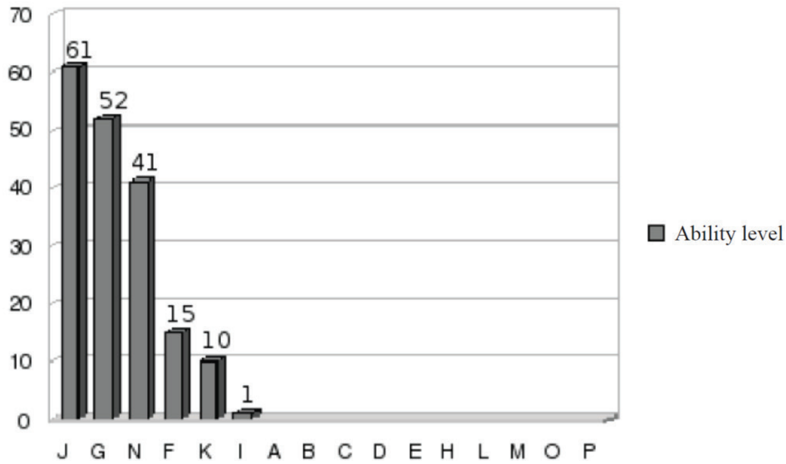

Figure 12. Characters' ability level.

The 12 participants rated the characters according to their ability to collaborate in the scenario. For example, Table 3 shows the number of votes that the character **G** had in each position of the rank. As the table shows, three participants selected the first position for the character **G**, five others selected the second position, and the rest selected the third position. Thus, all 12 participants classified the character **G** in some position. In addition, it is important to note that nobody selected the fourth position for this character.

After the participants classified the characters, we calculated their level of ability to collaborate. The ability level of each character was calculated through Equation 1.

Equation 1. Ability level

$$Ability(i) = \sum_{n=1}^{x} (F_n(i) * 2^{x-n})$$

The *Ability(i)* represents the ability level of the *i* character to collaborate in the scenario. This level is calculated through the weighted sum of the frequencies, which are represented by *F*, that the character appears in each one of the *n* positions of the rank ($F_n(i)$). As the rank has four positions, the *x* value is 4. The weight given to each element of the sum (2^{x-n}) is based on a geometric progression in order to highlight the most suitable elements from others. Figure 12 presents the results of the ability evaluation. Of the 16 characters, 6 appeared at least once in the participant's ranking. The four characters with the highest ability level, ordered in descending order, were the following: J, G, N, and F.

In the second phase of the evaluation we used the *PeopleFinder* application to make the same search that the 12 participants carried out. The application considered the same characters and search criteria. In the end of the *PeopleFinder* search, we obtained the following list (in decreasing order of importance): J, G, N and F. This list matches the list obtained from the participants' classification. Analyzing this result, we conclude that the application returned relevant results for the scenario considered. In addition, analyzing together the participants' answers and the applications' results, we obtained the following complementary information:

- 1 participant (8.33% of the sample) provided exactly the same result as the *PeopleFinder*;
- 7 participants (58.33% of the sample) provided the same three first results as the application, but not in the same order;
- All participants presented in their lists, in some position, the first and the second character provided by the application;

- 11 participants (91.67% of the sample) presented, in some position, the first, the second, and the third character provided by the application;
- 7 participants (58.33% of the sample) presented, in some position, the same results provided by the application.

We can note that all participants presented in their lists, in any order, the first two results provided by the prototype developed with MaPS. In addition, only one user did not cite, in any order, the first three results provided by the application. According to these results, we conclude that for more than 90% of the participants the MaPS framework was correct in the first three results.

This experiment of course has a minimum population to search. This was done to allow the participation of the volunteers. With 16 profiles we presented a situation where people could realize a search manually considering a defined criterion.

With the results obtained by the prototype, which are very similar to the ones indicated by the participants, we see that they are relevant. Considering this and the fact that the MaPS framework has a potential to be used by an application with thousands of profiles, we can see the advantages of its use. It is very difficult for a human to make a search in this scenario.

Conclusions

This article proposed a framework to assist in the development of collaborative applications for ubiquitous environments called MaPS. The framework focuses on the process of searching for communication channels and people to collaborate with, which are considered relevant aspects of Ubiquitous Collaboration (Izadi *et al*., 2002; Laso-Ballesteros, 2006; Caceres and Friday, 2012). MaPS uses context information and users' profiles to make the search more effective.

Related works presented a discussion of seven related works considering eight aspects of comparison. MaPS comprehends all the aspects discussed. As we can see in the comparison, the processes involved in the elementary steps of a collaboration, peer and communication channel selection, are not fully explored by the presented works. Thus there are opportunities for a new contribution.

Moreover, we can underline three additional contributions. First, all related works use in the searches data explicitly provided by the users. On the other hand, MaPS introduces the possibility of using context information provided by an external context middleware. Second, MaPS is the only model that proposes an integrated approach involving user profiles, context information and a lexical knowledge base. Finally, MaPS consists in a general framework that can be customized to support different collaborative applications, as shown in its evaluation.

The following general conclusions were reached in this work: (i) it is possible to use context information and users' profiles in an integrated approach to improve the search for collaborative peers; (ii) it is viable to create a general framework dedicated to develop ubiquitous collaborative applications. Moreover, specific conclusions about the proposed model can be highlighted: (i) MaPS contains the basic components to support the search for collaborative peers in ubiquitous environments; (ii) the initial applications developed attest to the usefulness of the framework and its extensibility; (iii) the initial results obtained in the evaluation of the framework functionalities attest to the good precision that can be obtained using MaPS.

According to Pree (1999), a framework must be specialized several times, continually, in order to discover its faults and to correct them, and still to identify new hotspots that were not discovered at a first moment. In this sense, it will be important to improve the functionalities of the two applications developed. Moreover, MaPS should be applied in the development of new applications, mainly considering different domains. MaPS enables one to develop applications that use any kind of context information in their searches. This characteristic should be better explored and evaluated in future applications. Additionally, MaPS can be extended to support Recommendation Systems, where users would evaluate the performance and proficiency of their collaborators. MaPS can use this information as a filter to improve the search process.

Acknowledgement

This work was financed by Capes/Brazil (Coordination for the Improvement of Higher Education Personnel; www.capes.gov.br) and CNPq/Brazil (National Council for Scientific and Technological Development; www.cnpq.br). We would like to acknowledge both institutions for their support. Finally, we would like to thank Unisinos (www.unisinos.br) and Feevale (www.feevale.br) for embracing this research.

References

BARBOSA, J.; HAHN, R.; BARBOSA, D.; SACCOL, A. 2011. A ubiquitous learning model focused on learner interaction. *International Journal of Learning Technology*, **6**(1):62-83.

http://dx.doi.org/10.1504/IJLT.2011.040150

BASU, A.; RAIJ, A.; JOHNSEN, K. 2012. Ubiquitous collaborative activity virtual environments. *In:* ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12), 12, Seattle, Washington, 2012. *Proceedings…* **1**:647-650. http://dx.doi.org/10.1145/2145204.2145302

BELLAVISTA, P.; CORRADI, A.; FANELLI, M.; FOSCHINI, L. 2012. A survey of context data distribution for mobile ubiquitous systems. *ACM Computing Surveys (CSUR)*, **44**(4):1-45. http://dx.doi.org/10.1145/2333112.2333119

BERGENTI, F.; POGGI, A.; SOMACHER, M. 2002. A collaborative platform for fixed and mobile networks. *Communications of the ACM*, **45**(11):39-44. http://dx.doi.org/10.1145/581571.581591

BERGENTI, F.; COSTICO, S.; POGGI, A. 2003. A portal for ubiquitous collaboration. *In*: Conference on Advanced Information Systems Engineering (CAiSE'03), 15, Klagenfurt/Velden, 2003. *Proceedings…* **75:**16-20. Disponível em: http://ceur-ws.org/Vol-75/files/UMICS_13.pdf

CACERES, R.; FRIDAY, A. 2012. Ubicomp systems at 20: Progress, opportunities, and challenges. *IEEE Pervasive Computing*, **11**(1):14-21. http://dx.doi.org/10.1109/MPRV.2011.85

CHEN, I.Y.L.; YANG, S.J.H. 2006. Peer-to-peer knowledge sharing in collaboration supported virtual learning communities. *In:* IEEE International Conference on Advanced Learning Technologies (ICALT'06), 6, Kerkrade, 2006. *Proceedings…* **1:**807-809.

http://dx.doi.org/10.1109/ICALT.2006.260

COSTA, C.A.; YAMIN, A.; GEYER, C.F.R. 2008. Toward a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing*, **7**(1):64-73. http://dx.doi.org/10.1109/MPRV.2008.21

DEY, A.; HIGHTOWER, J.; DE LARA, E.; DAVIES, N. 2010. Location-based services. *IEEE Pervasive Computing*, **9**(1):11-12. http://dx.doi.org/10.1109/MPRV.2010.10

DIAZ, A.; MERINO, P.; RIVAS, F.J. 2010. Mobile application profiling for connected mobile devices. *IEEE Pervasive Computing*, **9**(1):54-61. http://dx.doi.org/10.1109/MPRV.2009.63

DIVITINI, M.; FARSHCHIAN, B.; SAMSET, H. 2004. Ubicollab: Collaboration support for mobile users. *In:* ACM Symposium on Applied Computing, 19, Nicosia, 2004. *Proceedings…* **1:**1191-1195. http://dx.doi.org/10.1145/967900.968141

EDWARDS, W.K.; BELLOTTI, V.; DEY, A.K.; NEWMAN, M.W. 2003. The challenges of user-centered design and evaluation for infrastructure. *In:* SIGCHI Conference on Human Factors in Computing Systems (CHI '03), 3, Ft. Lauderdale, Florida, USA, 2003. *Proceedings…* **1**:297-304. http://dx.doi.org/10.1145/642611.642664

ELLIS, C.A.; GIBBS, A.; REIN, G. 1991. Groupware: Some issues and experiences. *Communications of the ACM*, **34**(1):39-58. http://dx.doi.org/10.1145/99977.99987

FARSHCHIAN, B.; DIVITINI, M. 2005. Ubicollab: improving collaboration with location services. *In:* International Conference on Pervasive Services (ICPS'05), Santorini, 2005. *Proceedings…* **1:**417-420.

http://dx.doi.org/10.1109/PERSER.2005.1506557

FARSHCHIAN, B. A.; DIVITINI, M. 2010. Collaboration support for mobile users in ubiquitous environments. *In*: H. NAKASHIMA *et al.* (ed.), *Handbook of Ambient Intelligence and Smart Environments*, Boston, Springer US, p. 173-199. http://dx.doi.org/10.1007/978-0-387-93808-0_7

FONTOURA, M.; PREE, W.; RUMPE, B. 2000. Uml-f: A modeling language for object oriented frameworks. *In:* European Conference on Object-Oriented Programming (*ECOOP 2000)*, 14, Sophia Antipolis and Cannes, 2000. *Proceedings…* **1:**63-82. http://dx.doi.org/10.1007/3-540-45102-1

FUKS, H.; RAPOSO, A.; GEROSA, M.A.; PIMENTEL, M.; LUCENA, C.J.P. 2007. The 3c collaboration model. *In:* N.F. KOCK (ed.), *The Encyclopedia of E-Collaboration.* Hershey, IGI Global, p. 637-644. http://dx.doi.org/10.4018/978-1-59904-000-4.ch097

GAMMA, E.; HELM, R.; JOHNSON, R.; VISSIDES, J. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, Addison-Wesley Professional, 416 p.

GEROSA, M.A.; FUKS, H.; LUCENA, C.J.P. 2003. Support to perception in learning digital environments. *Brazilian Journal of Informatics in Education*, **11**(2):1-19.

HALLSTEINSEN, S.; GEIHS, K.; PASPALLIS, N.; ELIASSEN, F.; HORN, G.; LORENZO, J.; MAMELLI, A.; PAPADOPOULOS, G.A. 2012. A development framework and methodology for self-adapting applications in ubiquitous computing environments. *Journal of Systems and Software*, **85**(12):2840-2859. http://dx.doi.org/10.1016/j.jss.2012.07.052

HIGHTOWER, J.; LAMARCA, A.; SMITH, I.E. 2006. Practical lessons from place lab. *IEEE Pervasive Computing*, **5**(3):32-39.

http://dx.doi.org/10.1109/MPRV.2006.55

IZADI, S.; COUTINHO, P.; RODDEN, T.; SMITH, G. 2002. The fuse platform: Supporting ubiquitous collaboration within diverse mobile environments. *Automated Software Engineering*, **9**(2):167-186. http://dx.doi.org/10.1023/A:1014534414062

JIN, Q.; ZHANG, G.; SHIH, T.K. 2005. Peer-to-peer based social interaction tools in ubiquitous. *In:* International Conference on Parallel and Distributed Systems, 11, Fukuoka, 2005. *Proceedings*... **1:**230-236.

http://dx.doi.org/10.1109/ICPADS.2005.224

KNAPPMEYER, M.; KIANI, S.L.; REETZ, E.S.; BAKER, N.; TONJES, R. 2013. Survey of context provisioning middleware. *IEEE Communications Surveys & Tutorials*, **15**(3):1492-1519. http://dx.doi.org/10.1109/SURV.2013.010413.00207

LASO-BALLESTEROS, I. 2006. Collaboration@work 2020: Ubiquitous collaboration research perspectives. *In:* International Conference on Collaborative Computing: Networking, Applications and Worksharing, 2, Atlanta, 2006. *Proceedings*... **1:**17-20.

http://dx.doi.org/10.1109/COLCOM.2006.361833

LOPES, J.; SOUZA, R.; GEYER, C.F.R.; COSTA, C.A.; BARBOSA, J.; GUSMAO, M.; YAMIN, A. 2012. Managing adaptation in Ubicomp. *In*: Latin America Conference on Informatics (CLEI 2012), XXXVIII, Medellin, 2012. *Proceedings*... **1:**1-10. http://dx.doi.org/10.1109/CLEI.2012.6427254

LUNA, V.; QUINTERO, R.; TORRES, M.; MORENO-IBARRA, M.; GUZMÁN, G.; ESCAMILLA, I. 2015. An ontology-based approach for representing the interaction process between user profile and its context for collaborative learning environments. *Computers in Human Behavior*, **51**(b):1387-1394.

http://dx.doi.org/10.1016/j.chb.2014.10.004

MILLER, G.A. 1995. Wordnet: a lexical database for English. *Communications of the ACM*, **38**(11):39-41. http://dx.doi.org/10.1145/219717.219748

PREE, W. 1999. Hot-spot-driven framework development. *In:* M. FAYAD; D.C. SCHMIDT (ed.), *Building Application Frameworks: Object-Oriented Foundations of Framework Design*. Austria, John-Wiley & Sons, p. 379-393.

SALBER, D.; DEY, A.; ABOWD, G.D. 1999. The context toolkit: Aiding the development of context-enabled applications. *In:* SIGCHI Conference on Human Factors in Computing Systems, Pittsburgh. *Proceedings*… **1:**434-441. http://dx.doi.org/10.1145/302979.303126

SATYANARAYANAN, M. 2001. Pervasive computing: vision and challenges. *IEEE Personal Communications*, **8**(4):10-17.

http://dx.doi.org/10.1109/98.943998

SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. 2009. The case for VM-based cloudlets in mobile computing. *IEEE Pervasive Computing*, **8**(4):14-23.

http://dx.doi.org/10.1109/MPRV.2009.82

STAHL, G. 2002. Groupware goes to school. *In:* International Workshop on Groupware: Design, Implementation, and Use, 8, La Serena, 2002. *Proceedings*... **2440:**7-24.

http://dx.doi.org/10.1007/3-540-46124-8_2

WAGNER, A.; BARBOSA, J.L.V.; BARBOSA, D.N.F. 2014. A model for profile management applied to ubiquitous learning environments. *Expert Systems with Applications*, **41**(4):2023-2034. http://dx.doi.org/10.1016/j.eswa.2013.08.098

WEISER, M. 1991. The computer for the 21st century. *Scientific American*, **265**:94-104.

http://dx.doi.org/10.1145/329124.329126

WEISER, M. 1993. Some computer science issues in ubiquitous computing. *Communications of the ACM*, **36**(7):75-84.

http://dx.doi.org/10.1145/159544.159617

YAMIN, A.; BARBOSA, J.; AUGUSTIN, I.; SILVA, L.C.; REAL, R.; GEYER, C.; CAVALHEIRO, G. 2003. Towards merging context-aware, mobile and grid computing. *The International Journal of High Performance Computing Applications*, **17**(2):191-203.

http://dx.doi.org/10.1177/1094342003017002008

YANG, S.J.H.; CHEN, I.Y.L. 2008. A social network-based system for supporting interactive collaboration in knowledge sharing over peer-to-peer network. *International Journal of Human-Computer Studies*, **66**(1):36-50.

http://dx.doi.org/10.1016/j.ijhcs.2007.08.005

ZHANG, G.; JIN, Q. 2005. Research on collaborative service solution in ubiquitous learning environment. *In:* International Conference on Parallel and Distributed Computing Applications and Technologies, 5, Dalian, 2005. *Proceedings*... **1:**804-806.

http://dx.doi.org/10.1109/PDCAT.2005.202

ZHANG, G.; JIN, Q.; LIN, M. 2005. A framework of social interaction support for ubiquitous learning. *In:* International Conference on Advanced Information Networking and Applications (AINA), 19, Taiwan, 2005. *Proceedings*... **2:**639-643. http://dx.doi.org/10.1109/AINA.2005.26

Submitted on July 07, 2015
Accepted on November 23, 2015