The Context Manager: Personalized Information and Services in Mobile Environments

Pablo Curiel, Ana B. Lago
Deusto Institute of Technology - DeustoTech
MORElab - Envisioning Future Internet
University of Deusto
Avda. Universidades 24
48007 - Bilbao, Spain
Email: {pcuriel, anabelen.lago}@deusto.es

***Abstract*—In this paper, we present a context management infrastructure for mobile service environments. Due to the remarkable advances the mobile technologies have experimented in the last years, mobile devices have become one of the most promising scenarios for the deployment of context-aware systems. For this reason, the aim of the proposed infrastructure is to provide context information to applications and services, both executed in the end-user terminals or in the network, enabling them to adapt their behaviour to each user and situation. The solution here exposed relies on semantic technologies and open standards to improve interoperability, and is based on a central element, the context manager, which acts as a central context repository and carries out demanding tasks in behalf of mobile devices.**

***Index Terms*—context management; semantic technologies; pervasive computing; mobile computing; context-aware services**

I. Introduction

Context awareness is a subject which has attracted interest since it was introduced by Schilit and Theimer in [1]. The reason for this growing interest is that, by using context information, context-aware systems are capable of adapting their working behaviour, as well as providing information and services more relevant in the situation of the end user. This way, context is defined as any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves [2].

One of the most promising scenarios for the deployment of context-aware systems are mobile phones. Since its appearance, mobile technology has undergone remarkable changes. First of all, mobile phones have become an everyday-use device, an almost essential element in our daily lives and which we carry with ourselves every time. But with the outstanding step forward in the technology of these devices in the last few years, this relevancy has become more noticeable. Nowadays, mobile phones are powerful hand-held computers capable of carrying out plenty of tasks and remaining online at all times. In addition, they include numerous sensors, and consequently have access to a great amount of personal and environmental information. At the same time, mobile phones are generally used for short and rapid interactions and in distractive environments, so providing information and services adapted to each situation is of great value for the end users. Therefore, mobile devices can greatly benefit from context-aware systems.

Taking this into account, in this paper we present a context management infrastructure for mobile environments. The aim of this infrastructure is to provide context information to applications and services which are either executed in the end-user terminals or in the network. To deal with the changing nature of this kind of environments and the wide range of devices present in them we propose a solution which relies on semantic technologies, easing the knowledge sharing and interoperability, and which provides generic mechanisms to deal with context information. Also, as mobile devices have limited computational capabilities, the context management infrastructure takes most of the computational burden of dealing with context, enabling them to simply request the information they need to carry out their tasks.

The remaining of this papers is structured as follows: in Section 2, related work is reviewed. In Section 3, the context management infrastructure is described. In Section 4, the implementation details are explained along with a preliminary validation scenario. Finally, in Section 5, discussion and future work are exposed.

II. Related Work

Numerous context-aware systems for mobile environments have been developed in the last years.

The work in [3] proposes a Context Managing Framework whose aim is to provide context information to mobile applications in order to adapt their behaviour according to that context. As our system, it uses a blackboard approach, based on a central context server, to decouple source and consumer communication. But in contrast to our solution, it places this central server inside the mobile phone and limits the context information usage to each mobile phone's own applications.

CoBrA project [4] proposes an architecture for supporting context-aware systems in smart spaces. It is based in a central agent, a broker which maintains a shared model of the context representing all the entities in a given space. It also uses a

 ISBN: 978-1-61208-236-3

policy language to allow users to define rules to control the use and the sharing of their private contextual information. However, the system is not explicitly designed to operate in mobile environments.

In [5], CASS, a middleware for context-aware mobile applications is introduced. Its main goal is to give service to devices with low processing capabilities, like mobile phones. For this purpose, as we do in our system, it delegates demanding tasks regarding context processing to external entities in the network. It also resembles our solution in making usage of a central context repository. But, whereas we adopt a semantic representation for that central repository, which enables a more flexible context processing and sharing, CASS uses a database for this purpose, which follows no semantic representation and relies mainly on a rule-based engine to process that context.

The MOBE architecture by Coppola et al. [6] intends to send applications to mobile devices depending on the context the user is in. MOBE differs from our solution in that it delegates most of the computing burden in the mobile phones, making them responsible for both gathering and processing context information, as well as deciding which applications to request in each case. It also restricts context information usage to select and adapt mobile applications, while in our system both mobile applications and applications executed in the network take advantage of this information.

Finally, in the context dissemination middleware [7], a peer-to-peer approach, based on web services is used to share context information between mobile devices. It uses a rule-based publish/subscription paradigm to enable this context information distribution. However, in contrast to our system both the context information both the rules are structured in an ad hoc format based on XSD (XML Schema Definition Language) [8], instead of a standard semantic representation format, which makes it difficult aspects like interoperability with other systems.

III. Context Management Infrastructure

The context management infrastructure is responsible for dealing with context information during its whole life cycle, from the provisioning of this type of information to the usage of it in benefit of the user, including all the intermediate processing needed to present that information in the way needed by the entities which make use of it. Therefore, it is a critical element in every context-aware system, and its design must be carefully carried out to guarantee the right operation of the whole system.

A. Context management requirements

In this section the requirements defined for the context management infrastructure are detailed.

First of all, to ease knowledge sharing and interoperability between the different entities in the system, the managed context information must be represented following a semantic model. Among the existing approaches to model context information, ontologies are used, because, as Strang and Linnhoff-Popien point out, they are the best solution for this task in ubiquitous computing applications [9]. Semantic technologies allow computing entities to better understand the meaning of the information they are working with, enabling them to perform reasoning and thus problem solving and decision taking. They also provide more flexible and expressive logical connections an relationships between data, even among those coming from different sources. And due to they embracing an open-world approach, they have the ability to better deal with the uncertainty and incompleteness of data in the real world. Therefore, all the entities in the system share a common ontological model and exchange context information according to it. Moreover, all the interactions between entities of the system that involve context information follow standards designed to work with semantic data, like RDF [10] to represent and share context information, and SPARQL [11] to query this kind of information.

Due to the limited computational capabilities of mobile devices, the context management infrastructure should take most of the computational burden of dealing with context information. For this purpose, we introduce a central element, the context manager. This element acts as a central repository, receiving context information from the sources, processing it as required and storing it, allowing consumers to access it. This blackboard approach enables resource-limited devices to only act as context source or consumers, relieving them from executing demanding tasks with context information. Indeed, it enables a data-centric approach granting independence between context sources and providers, as the context manager receives context information from the sources and stores it, responding consumers' queries about that information. Thus, it prevents consumers from asking directly to the sources and at the same time enables them to only think about what information they need, not where it comes from.

However, the context manager is not an atomic element, but consists of a series of independent and reusable components, which carry out different tasks with context information, facilitating system scalability, as they are even able to operate in different machines. Nonetheless, the context manager must provide a unique entry point to the context providers and consumers, known as the context manager API, in order to provide its functionality in a standard an unified way. This separation between the context manager logic and the access to it also enables providing different communication protocols to expose its functionality.

Finally, in order to meet the requirements regarding context information accessing of the different kinds of consumers present in the environment, both synchronous and asynchronous access are be provided. Thus, the context manager must be able of responding synchronous queries as well as registering queries, checking when those queries match and asynchronously notifying the corresponding consumer of this event.

B. The Context Manager

As detailed in the previous section, the context manager is the central element of the context management infras-

 ISBN: 978-1-61208-236-3

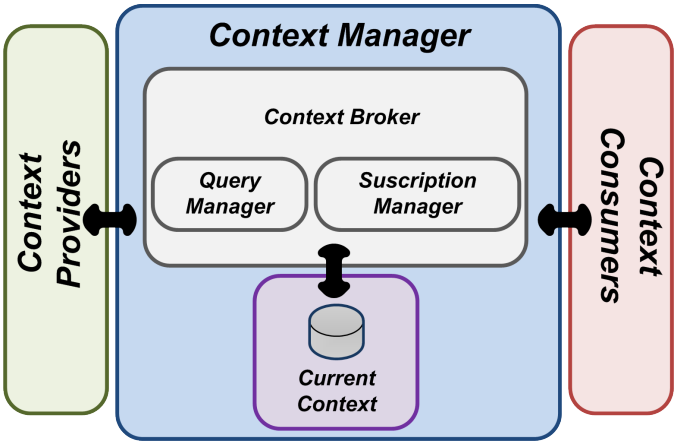

Fig. 1. Logic Architecture

tructure. It is comprised of various components which carry out different tasks with context information and exposes a unified interface for context sources and consumers to access its functionality, relieving them from carrying out demanding tasks with this type of information. In this section, the different components of the context manager, shown in Figure 1, are introduced and detailed.

The **current context** is the element which stores the context information which is valid in each moment, that is, the one that represents the current status of the entities which are considered part of the context in every moment. This information is stored as an RDF triplestore following the ontological model shared by the rest of the system and can be kept either in memory or disk. The **context broker** is the component responsible for managing this context repository. This way, it receives the context information from the sources, stores it in the current context and attends the consumer requests querying these repository. At the same time, it has two subcomponents in charge of answering the consumers' queries: the **query manager**, which deals with synchronous queries, and the **subscription manager**, responsible for registering consumers' subscriptions for context information changes and notifying them asynchronously when those changes take place.

C. The Context Management API

As defined, the context manager, even if it is composed of various independent components, exposes a unique entry point to its functionality. This API exposes several methods to manage context information in a model-independent way and relying on standard technologies. Indeed, the methods here detailed are merely an access layer, as the operational logic belongs to each corresponding component, so several communication protocols could be implemented to support as many devices as needed.

The methods exposed in the context manager API are the following:

- **Add Context Info**. This method enables a context source to add or update the current context space by providing information in RDF format.
- **Remove Context Info**. Using this method enables a context source to delete a context instance from the context space given its identifier.
- **Get Context Info**. This method enables a context consumer to retrieve a known instance from the current context given an identifier.
- **Query**. By calling this method, a context consumer can synchronously access the current context space providing an SPARQL query. The context manager will execute this query and return the corresponding context information to the context consumer in RDF format.
- **Subscribe**. This method enables a context consumer to asynchronously access the current context space. The consumer provides an SPARQL query which the context manager will register and a callback address which the second will use to asynchronously notify the first when the query is matched.
- **Unsubscribe**. To delete a subscription created with the previous method, a context consumer needs to invoke this method providing the Subscription ID which corresponds to the subscription that wants to remove.
- **Notify**. This method is not exposed by the context manager itself, but by those consumers which make use of the Subscription system. This way, each time an asynchronous query is satisfied, the context manager invokes this method of the corresponding consumer, providing the context information in RDF format.

IV. Implementation and Validation Scenario

As a first validation step of our proposal, we have developed a prototype of the context manager which fully implements the API described in the previous section, as well as an end-user application and a service, which act as context source and consumers.

The context manager is developed in Java using the OSGi component framework. To work with semanticized context information, we use the well known Jena2 [12], semantic web toolkit, and Jenabean [13], a library which bridges the gap between working with RDF graphs and Object-Oriented Programming. The context manager API is exposed as a RESTful [14] interface.

To test the functionality, a validation scenario is proposed, which involves an end-user application developed for the Android mobile OS and a service which tracks both user location and Twitter accounts to infer user status and availability, in order to suggest plans to nearby friends. The scenario goes by as described. John, Mike and Greg are friends. The two first live in Madrid, while the third lives in Barcelona. The three of them are tech-savvy, and therefore both social network and smartphone heavy users. They also use a social alerts service, which helps them to keep in touch with their friends. One day, Greg travels to Madrid, and the context manager, which periodically receives users' locations, reported by their smartphones, detects that the three friends are nearby and notifies the social alerts service. However, Mike has recently updated his Twitter account, informing his followers of his busy day,

 ISBN: 978-1-61208-236-3

'What a day! I've got 3 meetings in a row!'. Consequently, the social alerts service, who periodically checks the three friends' Twitter accounts, infers that Mike is not available, but sends alerts to John and Greg suggesting them to arrange a meeting. Finally, if both agree, it checks their user profiles to select a restaurant they both like to organize a lunch.

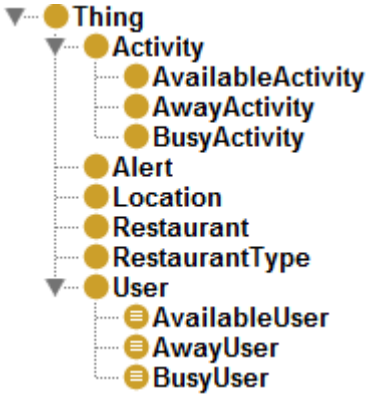

Fig. 2. Ontology of the validation scenario

To enable context information sharing between the context manager, the mobile application and the social alerts service, the ontology show at Figure 2 is used, which models information about Users (their personal data, preferences, current location and current activity), Locations, Restaurants and Alerts sent to users.

V. Conclusion and Future Work

In the present article, we have proposed a context management infrastructure for mobile environments, in which applications and services both executed in the end-user terminals or in the network use this context information to become more relevant for the users. As this kind of environments are of a very changing nature an a wide range of devices coexist in them, our proposal offers generic and abstract methods to work with context information, and relies on semantic technologies and open standards, trying to offer a solution as interoperable and extensible as possible.

On to other matters, even if the computational capabilities of mobile devices have noticeably increased in the last years, their ability to carry out demanding tasks with context information is limited, so our proposal delegates this heavy tasks in a central element, the context manager, which acts as a context information repository, exposing an interface to provide and consume context information. This also enables context source and consumer independence and a data-centric approach, in which consumers only have to worry about what information they need, not where to retrieve it from.

Our next steps will involve including the privacy and security policies required for this kind of systems, which grant both preventing unsolicited access to sensitive data while enabling legitimate access to those entities which need it to successfully carry out their tasks. In addition, implementing support for a context history component, which keeps track of the context information changes, could be a subject of interest as long-term user behaviour and trends can be inferred using this kind of information. Finally, a more rigorous validation will be carried out, involving performance tests with more demanding real-life use cases, in order to detect possible weakness and assess the validity of the proposed solution.

Acknowledgment

This work is supported by the CENIT Research Program, as part of the INGENIO2010 Spanish National Fund.

References

[1] B. Schilit and M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Netw.*, vol. 8, no. 5, pp. 22–32, 1994.

[2] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, Jan. 2001.

[3] P. Korpipää, J. Mantyjarvi, J. Kela, H. Keranen, and E. Malm, "Managing context information in mobile devices," *IEEE Pervasive Comput.*, vol. 2, no. 3, pp. 42–51, 2003.

[4] H. Chen, T. Finin, and A. Joshi, "An intelligent broker for context-aware systems," in *Proc. of Ubicomp*, Seattle, Washington, USA, 2003, pp. 183–184.

[5] P. Fahy and S. Clarke, "CASS-a middleware for mobile context-aware applications," in *Proc. Workshop on Context Awareness, MobiSys*, 2004, pp. 304–308.

[6] P. Coppola, V. Della Mea, L. Di Gaspero, S. Mizzaro, I. Scagnetto, A. Selva, L. Vassena, and P. Riziò, "MoBe: context-aware mobile applications on mobile devices for mobile users," in *Proc. International Workshop on exploiting context histories in smart environments, ECHISE*, Munich, 2005.

[7] G. Gehlen, F. Aijaz, M. Sajjad, and B. Walke, "A mobile context dissemination middleware," in *Proc. International Conference Information Technology, ITNG'07.*, 2007, pp. 155–160.

[8] XML Schema Working Group. (2010, Jan.) XML schema. http://www.w3.org/XML/Schema/. [Last accessed on August 10, 2012].

[9] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *Proc. Workshop on Advanced Context Modelling, Reasoning and Management*, Nottingham, England, 2004.

[10] RDF Working Group. (2004, Feb.) RDF - semantic web standards. http://www.w3.org/RDF/. [Last accessed on July 5, 2012].

[11] E. Prud'hommeaux and A. Seaborne. (2008, Jan.) SPARQL query language for RDF. http://www.w3.org/TR/rdf-sparql-query/. [Last accessed on July 5, 2012].

[12] Apache Software Foundation, "Apache jena," http://jena.apache.org/, Apr. 2012, [Last accessed on August 10, 2012].

[13] "Jenabean," http://code.google.com/p/jenabean/, Feb. 2010, [Last accessed on August 10, 2012].

[14] L. Richardson and S. Ruby, *RESTful web services*. O'Reilly Media, 2007.

 ISBN: 978-1-61208-236-3