A context information manager for pervasive environments

Jérôme Euzenat, Jérôme Pierson, Fano Ramparany

A Context Information Manager for Pervasive Computing Environments

Jérôme Euzenat[1], Jérôme Pierson[2], Fano Ramparany[2]

Abstract. In a pervasive computing environment, heterogeneous devices need to communicate in order to provide services adapted to the situation of users. So, they need to assess this situation as their context. We have developed an extensible context model using semantic web technologies and a context information management component that enable the interaction between context information producer devices and context information consumer devices and as well as their insertion in an open environment.

1 INTRODUCTION

In a pervasive computing environment, various basic services can be provided by smart devices (e.g., sensors, actuators, human-computer interface). More advanced services can be provided when they act together and cooperate, but smarter services can only be achieved if the devices could adapt their behaviour to the user, his/her preference and his/her task, than if users have to find the specific service they want among all the smart devices.

This idea requires the perception of the environment in which devices and users interact. There are pieces of information that can be considered common to all services. In particular, spatial and temporal location as well as information related to the physical environment in which services are made available [1, 2]. These elements are part of the context in which applications operate. We are here concerned with context-aware applications, i.e., applications whose behaviour is determined to some extent by the context.

Our goal is to design a context management system general enough to be used by different pervasive computing applications, specific enough for encompassing existing services and applications, and flexible enough for supporting the dynamic addition of new devices.

First we introduce our proposal for a distributed architecture that manages context information (Section 2), then we define a context representation (Section 3) which is independent of applications and an architecture enabling their evolution. The openness of the system will require dealing with heterogeneous representations that will have to be reconciled before being used (section 4). For that purpose, we will take advantage of solutions developed for the "semantic web".

2 CONTEXTS

Context is the set of information (partly) characterizing the situation of some entity [5]. The notion of context is not universal but relative to some situation [15, 11]. This can be a physical situation (as the spatio-temporal location of some person) or functional (as the current task of the person).

Although, several scientific domains have considered the notion of context, the standpoints from which this notion is considered are different: in pervasive computing, the context of an application in terms of its physical parameters has been especially considered ; in human-computer communication, the context is most often the user task and the history of its dialogue with the computer [4]; in artificial intelligence, the context is rather considered as the conditions of validity of an assertion [14].

2.1 Context in pervasive computing

In pervasive computing, the physical context is of the utmost importance. In general, it is acquired through sensor data. These data are further elaborated into context characterization adapted to their use (for instance « high temperature » for some air conditioning controller). With regard to the sensor data (a temperature), the information has been weakened (i.e., made less precise) but is more adapted.

The various definitions of context in pervasive computing are very often related to an application or a particular domain [6, 15]. The drawback of this characterization is its reliance on the task: « high temperature » is not an absolute characterization. It depends on the use of the room (a sauna or a sleeping room). More than context, pervasive computing tends to manipulate a characterization of the context in the perspective of an application. As a consequence, it is difficult to dynamically implement non expected applications with the characterization of context made for another one.

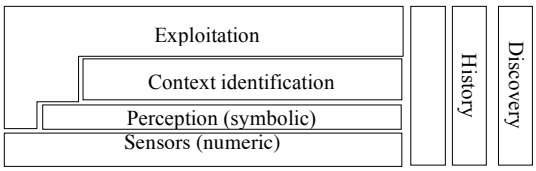

Figure 1: Model for context in pervasive computing. Data coming from sensors are aggregated and elaborated into the context used by applications (from[7]). This paper does not consider the orthogonal aspects (discovery, history and security).

However, multi-application context modelling is now understood in pervasive computing [7] and raises the issue of considering context independently from applications. Figure 1 shows the way to progressively elaborate context information from sensors to applications. We will follow this approach and this paper details the content of the perception and situation layers so that they can support the dynamic nature of the environment (new sensors and applications appear and disappear).

[1]INRIA Rhône-Alpes, France
[2]France Telecom R&D, France

2.2 Contexts in artificial intelligence

In artificial intelligence, the notion of context is, in general, concerned with the representation of information. It is used for accounting for two phenomena: the context of validity of information [16] and the efficiency of reasoning in narrower contexts [1].

John McCarthy [17] proposed a formalization of context based on context « reification » as well as the « meta-predicate » *ist*, *ist*(*p*,*c*) meaning that assertion *p* is true in context *c*. The approaches of context in artificial intelligence allow grouping knowledge in micro-theories [1] and to reason within those. In this framework (that of Cyc), the context is a more precise frame for interpreting information. This kind of approach can be used in pervasive computing in order to integrate and interpret data provided by sensors. Taking advantage of the theory associated with the sensor enables reducing the ambiguity of the data it delivers. In that view, raw data issued from sensors, are generally not weakened but rather enriched (and aggregated with other information sources allowing to further precise their interpretation). [14] describes the way to express this kind of context within the semantic web by providing each triple information on its origin (« quad »). The same model is implemented in modern RDF managers [2].

Although work from McCarthy and Guha consider contexts as independent theories related to some particular knowledge field, Fausto Giunchiglia instead considers contexts as concurrent viewpoints on the same information. He expresses the relations between contexts as « mappings » used for importing information under some context into another. This approach can be useful in pervasive computing when several information sources provide comparable information. These works found their way within semantic web tools through the C-OWL language [18]. A comparison of both approaches is made in [19].

2.3 Synthesis

In summary, pervasive computing tends to consider context as what characterizes the situation while artificial intelligence rather characterizes the information itself. More notably, Pervasive computing very often deals with the particular context of an application while artificial intelligence determines the context in function of the information source. In pervasive computing, information coming from sources is very often weakened in order to fit the application needs while artificial intelligence tends to enrich it with further information.

Of course, these approaches are rather complementary than competitors. In general, raw data can go through weakening and enrichment, thus bridging both approaches.

In pervasive computing, upgrading the environment is not an option: the environment must be designed from scratch in order to evolve. Our goal is to contribute to dealing with the dynamic evolution of context [7]. For that purpose, we design an architecture supporting the introduction of new context elements (provided from some new device) and the introduction of new applications without interruption of the environment.

This component-based context management architecture relies on a context modelling formalism based on semantic web technologies. We demonstrate how they can be used to dynamically extend the environment.

3 A CONTEXT INFORMATION MANAGEMENT COMPONENT

Pervasive Computing applications retrieve context data directly or indirectly from sensors, which are grounded in the physical environment. We propose an architecture in which applications do not need to directly connect to each sensor available and where adding a new sensor does not require all applications to be recompiled and redeployed.

3.1 Architecture

Designing an architecture for hosting context-aware services, suggests the development of a context management service for providing other services or devices with context information [6, 7, 11]. We have identified several alternative approaches for designing the target architecture. The first approach lets applications directly communicate with sensors they have an interest in. This approach requires applications to know in advance who they need to communicate with to get the information they need. Furthermore it adds complexity to the process of information aggregation, as this process should then be handled by the applications themselves and overloads sensors activity. Finally this approach makes it difficult to insert new sensors into the environment and thus doesn't comply with our flexibility requirement.

In the framework of service oriented architectures, the second approach consists of building a context management service [4] whose job is to collect sensors information and forward this information to applications that need it. This approach makes it possible to gather sensor information in a single place so that information could be easily aggregated. For example, a system that provides local temperature and atmospheric is very useful in a home environment. At a city level, the same information is useful; however it doesn't need the same degree of precision. The drawback of such a system is that it centralizes the management of context information, which is contradictory to the concept of context. More specifically, this system provides information about the activity environment (a special case of context information), however this information is not contextual as it is independent of the current task or situation, i.e. that of the client application. Moreover, with such a system, the scope of context management would be efficient in a limited area only.

We have adopted a third approach in which each device or service embeds a context management component (CMC) for maintaining context information for its own use or for the benefit of others (Figure 2). The main advantage of this approach is that new devices can join online or leave, without having to recompile or reinitialize any part of the whole environment. This component provides mechanisms for helping context-aware devices to request context information from context sensitive devices.

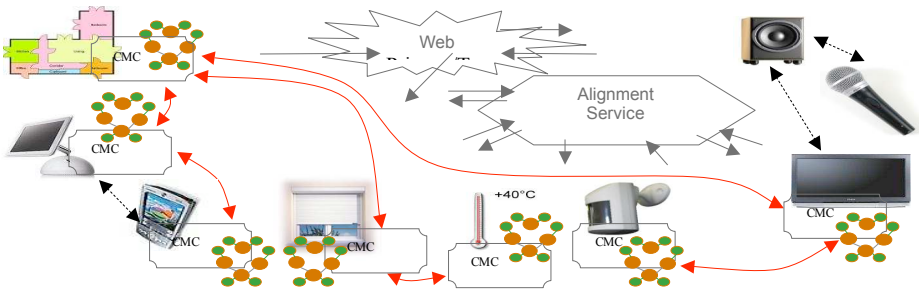

Figure 2: Each device embeds a context management component (CMC) and a semantic description of its context.

3.2 Interaction

Applications should be able to query context information they are interested in and some services should be able to provide context information, such as aggregated context information to other devices. For this purpose we design a protocol that makes the best of available services. We need to be able to identify a service, to know what kind of context information it could provide and to interact with it to get access to this information. Thus the context management component provides a few methods. In our description the first element is the query, the second is the response type:

`Id() -> URI`: The identifier of the service;
`Cl(URI) -> URI`: The class of the identified service;
`Desc(URI) -> OWL`: The description of the information that the component can provide;
`Req(RDQL) -> RDF`.

The first method allows identifying devices that are available in the environment. The identifier can then be used to contact the device. Alternatively, it could be used to get a more detailed description of the device (e.g., in case the identifier is a URI pointing to a network location where a description of the identified object is stored). A second method identifies the class (in OWL terminology) of the device. In theory, this class should be accessible from the network and once its definition is found, it provides a detailed description of the device. A third method provides the device description (or rather that of context information they provide) in an OWL formalism (OWL-S). A fourth method is used to post queries to the devices and to get the context information returned.

Thus any device is able to: find out, in its environment, services that are able to provide information relevant to its own context, get features of services that have been found (for example, measurement precision), connect to the selected service to get the information sought.

We need a language to describe the context model of heterogeneous devices so that these devices can interact in a dynamic environment.

4 OPENESS, DYNAMICS AND HETEROGENITY

The languages developed for the semantic web, and particularly RDF and OWL, are adapted to context representation in pervasive computing and particularly to the representation of dynamically evolving contexts for two reasons: these languages are open: they implement the open world assumption under which it is always possible to add more information to a context characterization; and they have been designed to work in a networked way.

4.1 Context model and language

In this dynamic pervasive computing environment, each CMC manages context information of its device. To express its context model, its needs or its capabilities, we use semantic web languages. They ensure interoperability between these heterogeneous devices.

The ground language for the semantic web is RDF (Resource Description Framework [8]). It enables expressing assertions of the form subject-predicate-object. The strength of RDF is that the names of entities (subjects, predicates or objects) are URIs (the identifiers of the web that can be seen as a generalization of URLs: http://www.w3c.org/sw). This opens the possibility for different RDF documents to refer precisely to an entity (it is reasonable to assume that a URI denotes the same thing for all of its users).

The OWL language [9], has been designed for expressing « ontologies » or conceptual models of a domain of knowledge. It constrains the interpretation of RDF graphs concerning this domain. OWL defines classes of objects and predicates and makes it possible to declare constraints applying to them (i.e., that the « output » of a « thermometer » is a « temperature »).

The context model that we use at that stage is very simple: a context is a set of RDF assertions. Interoperability is guaranteed through considering that context-aware devices are consumers and producers of RDF. However, this is not precise enough and devices may want to extract only the relevant information from context sources. For that purpose, a language like RDQL [10] is useful for querying or subscribing to context sources. In order to post the relevant queries to the adequate components, it is necessary that components publish the OWL classes of objects and properties on which they can answer.

4.2 Why ontologies?

If we can add components at any time, they may not be easily usable. Indeed, there is no a priori reason that components available, new applications and new sensors are compatible. Fortunately, knowledge representation techniques, and namely the open world assumption, makes it possible to introduce new device specifications in the environment by extending the ontology, through specifying a new concept or a property. Using ontologies to characterize the situations permits new equipment whose capabilities have not been known at the beginning to enter and new applications to benefit from these possibilities. The applications

must be as general as possible describing the information they need whereas the context management system must be as precise as possible on the information it makes available. This approach enables the most specialized applications to take advantage of CMCs. The essential point is to have sufficiently generic ontologies to cover the various concepts implied in pervasive computing applications [12].

4.3 Taking advantage of heterogeneous resources

The context management system we propose makes it possible to introduce new devices in the environment by extending the ontologies in such a way that existing applications can make the best use of them. However, this view holds if all parties share the same ontology.

Unfortunately this is not always the case and agreeing on standard, universal and self contained context ontology is not a reasonable assumption. This raises the issue of matching context information with applications context information requirements. There are three alternative approaches addressing interoperability in pervasive computing environments: (*i*) A priori standardisation of ontologies, (*ii*) setting up mediators among ontologies and (*iii*) a dynamical ontology matching service. These three approaches are not incompatible and might even be jointly used. For example parties could agree on sharing common high level ontologies. Letting more specific level ontology evolve freely and independently is a strategy enabling a close account for a fast evolving domain.

As ontologies, matching services should be available for applications and context managers through network access. They provide an interface that allows the explicit handling of ontologies alignments developed in the framework of the semantic web [20]. We propose to set up one (or more) ontology matching service(s) (Figure 3). The goal of such services is to help agents (context managers in our case) to find a matching between different ontologies. These services provide mechanisms for finding out ontologies close to a given ontology, archiving (and retrieving) past alignments, dynamically computing matching between two ontologies and translating queries and responses to queries between context managers that use different ontologies [13].

5 RELATED WORKS

In pervasive computing, it is largely recognized that handling context information is essential. As we presented, there are many different management systems for context information. The one which is the nearest to what we presented here is the work on contextors [11]. It proposes a library of elements able to provide context information: it makes it possible to combine contextual information on a distributed mode. On the other hand, this system does not establish how to dynamically add devices without stopping the system or other devices. Regarding to the use of the semantic Web technologies to represent context, there are several proposals to extend the languages of the semantic Web in order to contextualize the assertions [14, 19, 2]. With regard to the use of OWL to represent the context information, [12] introduces a high level ontology of contextual information for pervasive computing.

6 CONCLUSION AND PERSPECTIVES

We specifically addressed the problem of adaptability of context management to an ever-evolving world. This is achieved by providing a distributed component-based architecture and by using semantic web technologies. Components enable the addition, at any moment, of new devices that can provide information about the context of applications. The use of RDF and OWL ensures interoperability between components developed independently by taking advantage of the open character of these technologies. Moreover, using ontology alignment modules allows dealing with the necessary heterogeneity between components. The proposed approach relies on a minimal commitment on basic technologies: RDF, OWL, and some identification protocol.

We are currently developing a demonstrator of this technology. It consists of a toolkit for developers of pervasive applications which help them deploy a distributed context management system. This toolkit provides a component for managing (searching, broadcasting and updating) context information.

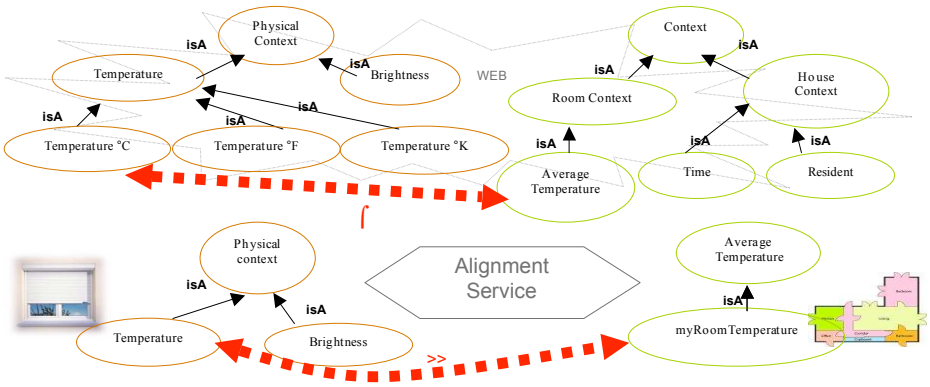

Figure 3: For finding correspondence between its model and the model of the context information provider, the window service asks to an alignment service to translate his model to another device model.

ACKNOWLEDGEMENTS

Fano Ramparany and Jérôme Pierson are partially supported by the European project Amigo (IST-2004-004182); Jérôme Euzenat is partially supported by the European network of excellence Knowledge Web (IST-2004-507482).

REFERENCES

[1] R. Guha, Contexts: a formalization and some applications, PhD thesis, Stanford university (CA US), 1991 (Technical Report STAN-CS-91-1399-Thesis et MCC ACT-CYC-423-91).

[2] O. Khriyenko, V. Terziyan, Context description framework for the semantic web, Proceedings Context 2005 Context representation and reasoning workshop, Paris (FR), 2005

[3] A. Dey, D. Salber, G. Abowd, A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, *Human-Computer Interaction* 16:97-166., 2001

[4] P. Dourish, Seeking a foundation for context-aware computing, *Human-Computer Interaction*, 16(2-3), 2001.

[5] M. Chalmers, A Historical View of Context, *Computer supported cooperative work* 13(3), 223-247, 2004.

[6] A. Dey, Understanding and using context, *Personal and ubiquitous computing* 5(1):4-7, 2001.

[7] J. Coutaz, J. Crowley, S. Dobson, D. Garlan, Context is key, *Communications of the ACM* 48(3):49-53, 2005.

[8] G. Klyne, J. Carroll, Eds., Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, 2004 http://www.w3.org/TR/rdf-concepts/

[9] M. Dean, G. Schreiber Eds, OWL Web Ontology Language: Reference, W3C Recommendation, 2004. *http://www.w3.org/TR/owl-ref/*

[10] A. Seaborne, RDQL — A Query Language for RDF, W3C Member submission, 2004. *http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/*

[11] J. Crowley, J. Coutaz, G. Rey, P. Reignier, Perceptual components for context aware computing, Proceedings International Conference on Ubiquitous Computing, Göteborg (SW), pp. 117-134, 2002.

[12] X. H. Wang, D. Q. Zhang, T. Gu, H. Keng Pung, Ontology based context modeling and reasoning in OWL, Proceedings 2nd International conference on pervasive computing and communication Workshop on Context Modeling and Reasoning (CoMoRea), Orlando (FL US), 2000.

[13] J. Euzenat, Alignment infrastructure for ontology mediation and other applications, Proceedings ICSOC 1st international workshop on Mediation in semantic web services, pp.81-95, Amsterdam (NL), 2005.

[14] R. Guha, R. Fikes, R. McCool, Contexts for the Semantic Web, Proceedings 3rd ISWC, Hiroshima (JP), *LNCS* 3298:32-46, 2004.

[15] H. Chen, T. Finin, A. Joshi, An Ontology for Context-Aware Pervasive Computing Environments, *Knowledge engineering review* 18(3):197-207, 2004.

[16] J. de Kleer, An assumption-based TMS, *Artificial Intelligence* 28(2):127-162, 1986.

[17] J. McCarthy, Notes on formalizing context, Proceedings 13th IJCAI, Chambéry (FR), pp. 555-560, 1993.

[18] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, H. Stuckenschmidt, C-OWL: contextualizing ontologies, Proceedings 2nd ISWC, Sanibel Island (FL US), *LNCS* 2870:164-179, 2003

[19] L. Serafini, P. Bouquet, Comparing formal theories of context in AI, *Artificial Intelligence* 155:1-67, 2004.

[20] J. Euzenat, An API for ontology alignment, Proceedings 3rd ISWC, Hiroshima (JP), *LNCS* 3298:698-712, 2004.