COMMUNICATIONS OF EASST

Context Management and Semantic Modelling for Ambient Intelligence

Fano Ramparany*, Jérôme Euzenat,**
Tom Broens*, Jérôme Pierson*,**
André Bottaro*, Remco Poortinga ***
* France Telecom R&D
** INRIA Rhône Alpes
*** Centre for Telematics and Information Technology
**** Telematica Instituut

Abstract. *Ambient Intelligence aims at pushing forward a user centric vision of Pervasive Computing, where the environment better serves our need. This paper describes our current work on modelling and managing context information for smart environments.*
Keywords: Context-Awareness, Management, Modelling, OWL

1 Introduction

Ambient Intelligence (AmI) builds upon concepts from the Pervasive Computing (PC) paradigm. PC aims at flooding our daily physical environment in computing and communication in such a way that the environment can act as an interactive collection of interconnected network of 'daily things'. AmI aims at pushing forward a vision where this environment also proactively serves our needs by understanding our activities, anticipating our needs and collaborating with us in achieving our daily tasks. To make this vision come true, the AmI environment needs to be aware of any information that is helpful for identifying user's activities, needs and tasks. This information is to be found in the physical environment as well as from the users themselves or from the computer systems they use. We call such kind of information 'contextual information'.

In this paper, we introduce an infrastructure for managing context information that we are developing in the Amigo[1] project. We state the main problems we are addressing while designing this infrastructure. We illustrate its role through concrete scenarios. We then introduce our architecture for the Context Management infrastructure. We then present the current state and preliminary results of our on-going work. We finally discuss how a service-oriented approach could exploit the context management infrastructure to give rise to context aware services and conclude.

[1] The authors would like to thank the CEC for partially supporting the work reported here, in the framework of the IST-Amigo collaborative project.

2 Problem Statement

The term 'context' is overloaded with a wide variety of meanings, depending on application purposes and on the research community standpoint ([1]). Several research communities like Information bases, Artificial Intelligence, Human Computer Interaction and Ubiquitous Computing have proposed ‘context’ definitions. We adopt the general definition proposed by Dey ([2]): "…Context is 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 application themselves…"

In ambient intelligence, we can distinguish three categories of context ([3]):

- Device context like memory, computation power, networks (and their quality), codecs, etc.
- User context divided into personal (health, mood, activity, etc.), applicative (emails, visited websites, preferences, etc.) and social (relatives, employment, activism, etc.) contexts.
- Physical context: contextual information related to the physical environment of an entity (device, room, building, and user). Examples are location, time, weather, altitude, light.

To be able to use contextual information in an AmI environment, we need some management functionalities. Context management is responsible for propagating context information from sensors to the application, storing it, controlling various manipulations on it (e.g., aggregation), and providing access control to the context information.

2.1 CONTEXT AWARENESS

Context awareness denotes the use of contextual information in computer system functionality. According to Dey ([2]), "Context-awareness is the property for a system of using context to provide relevant information and/or services to the user, where relevancy depends on the user’s task".

2.2 CONTEXT AWARENESS SCENARIOS

In this section, we introduce 3 simple scenarios, which illustrate the concept of context awareness applied to ubiquitous computing services. Context awareness here means that the primary function of the service adapts to the current context of the physical object, i.e. the context that holds while invoking the service.

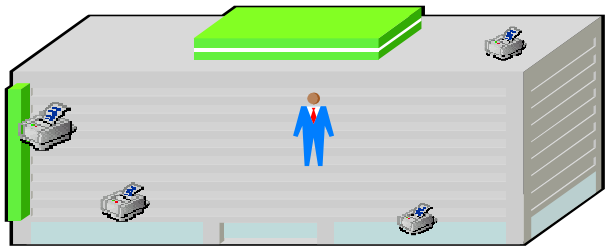

Figure 1: Printer location scenario.

In the printer location scenario, the user (precisely the printing service) benefits from the printers' physical location information and his/her own location information. The service offers printing facilities minimizing the moves throughout the building.

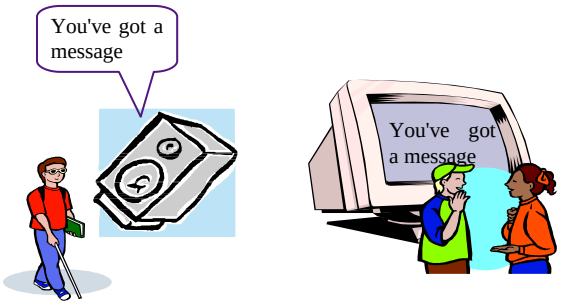

Figure 2: Profile aware notification scenario.

In the recipient's profile aware notification scenario, a high level service aims at notifying a user that "he/she has just received a mail". The service takes the personal context of the user (e.g. accessibility profile – deaf/blind/) into account to select the most appropriate notification modality (voice/sound notification service/device or text/screen notification service/device).

Figure 3: Energy-preserving bulb scenario

In the energy preserving bulb simple scenario, the context aware bulb adjusts the light intensity to the environmental context (e.g. darkness/sunshine) or user context (e.g. activity).

3 Approach

Scenarios described above quite naturally suggest the use of a central context management service, designed to supply end user services, applications, or devices with the context information they need. The main advantage of this approach is to gather contextual information at a single place, making it easier to find and retrieve context information. The main drawback of a central context management service is the centralized approach; which could even be inconsistent with the very notion of context, as it sets context as task insensitive (as perceived by client services). On the opposite, a purely decentralized management fails in federating resources and tends to overload sensor devices.

Our solution is to define services in such a way that they become capable of managing their own context, and let some of them supply other devices and services with context information. This last configuration makes it possible to aggregate context information. Instead of implementing context management as a mere central service, we prefer to distribute it over devices and application services, as software components. In the following sections,

we introduce the architecture of the context management infrastructure and focus on one of its main components: Context information modeling.

3.1 CONTEXT MANAGEMENT INFRASTRUCTURE

Context information is managed in a distributed way so that it is shared among interested entities. Devices (e.g. sensor devices) provide low-level context information. Context-aware services consume information coming from devices and other services in order to build their own high level context information which they may in turn offer to other services. None of these entities holds a complete global view of context information. Figure 4 displays how services and devices exchange context information.

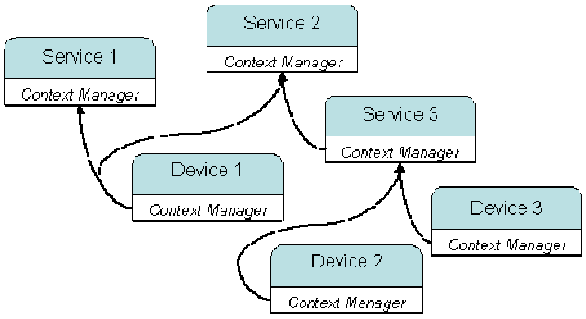

Figure 4: Context information flow.

Every entity (i.e. device or service) contains parts of the functionality of the whole context management infrastructure, denoted as 'Context Manager' blocks in figure 4. The different functionalities that can be part of these blocks are:

- Context Sources providing context information to any interested party. Two types of Context Sources can be distinguished:
  - Context Wrappers adapting raw data coming from sensors into the context model.
  - Context Reasoners interpreting context queries and aligning heterogeneous models.
- Context Brokers keeping track of the different context sources and their information.
- Context Stores storing information used to satisfy incoming queries. Each entity holds its own context model according to a specific ontology.

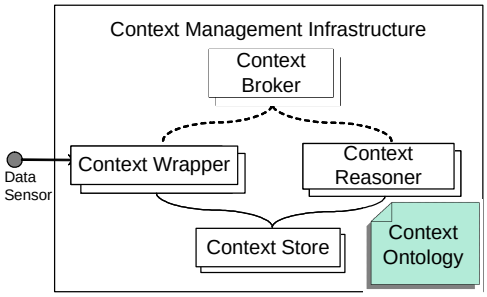

Figure 5: Context Management Infrastructure functional elements.

There are three complementary ways to restore interoperability between ontologies:

- Standardizing the required ontologies a priori.
- Recording correspondences as soon as ontologies are made available by parties.
- Dynamically matching ontologies.

In every case, the infrastructure must be able to provide the best possible use of available context information. To this purpose, we use and adapt technologies developed in the framework of the Semantic Web [4].

3.2 CONTEXT INFORMATION MODELING

Developing context-aware services requires an appropriate way of representing context information, building it incrementally, maintaining/updating it over time, and being able to retrieve it dynamically. We address the representation issue in this section.

Based on the analysis of previous work ([5]), we base our model of context information on anontology of physical context. The latter defines concepts, i.e. the types of the relevant objects in the studied context. For instance, temperature, weight, and distance are concepts of the context ontology. Every concept is defined in terms of attributes, attribute constraints or restrictions, and relations with other concepts. For example the temperature concept has a numerical attribute "value" which is restricted to be higher than -273.15ºC and it shares the relation "characterizes" with the concept "place". The context ontology could be viewed as an abstract model. Context information is then modeled as instances of this ontology.

Primary types are modeled in the first version of the infrastructure. They include low level concepts such as discrete, scalar, continuous, array data (Figure 6). Higher level concepts and relations which form the context ontology will be integrated in subsequent versions. Such incremental approach via composing, completing and consolidating preliminary models suit well the process of building ontologies.

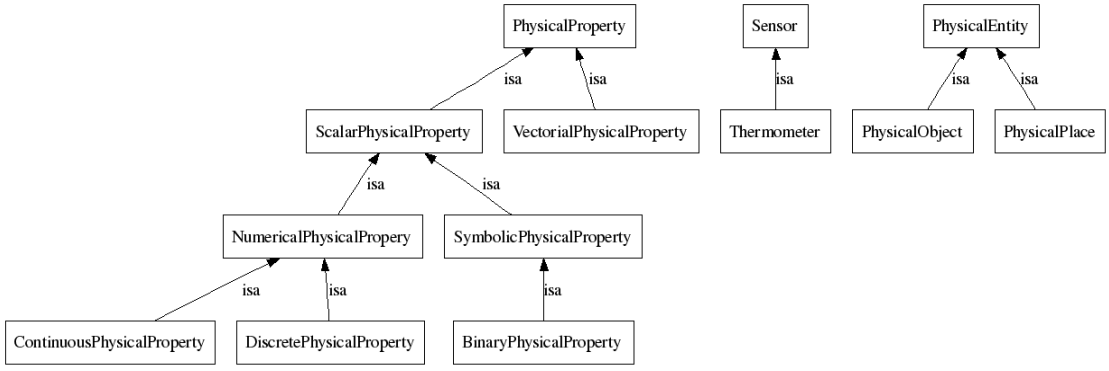

Figure 6: Physical Context model excerpt.

Sensor and PhysicalEntity class hierarchies attach context information to physical places or objects, and relate context information to its source (Figure 7). Context information is made

of instances of PhysicalProperty's terminal subclasses. For example, if a Thermometer named Thermometer_1 provides a temperature measurement of 25.0ºC, an instance of ContinuousPhysicalProperty class called Temp_measurement_1 is created, the number 25.0 is assigned to the slot "Value" and a relation "measured_by" is established between this instance and the sensor (Figure 7). Generic data properties like precision, reliability, timestamp and resolution are defined at the topmost level of the Physical properties hierarchy.

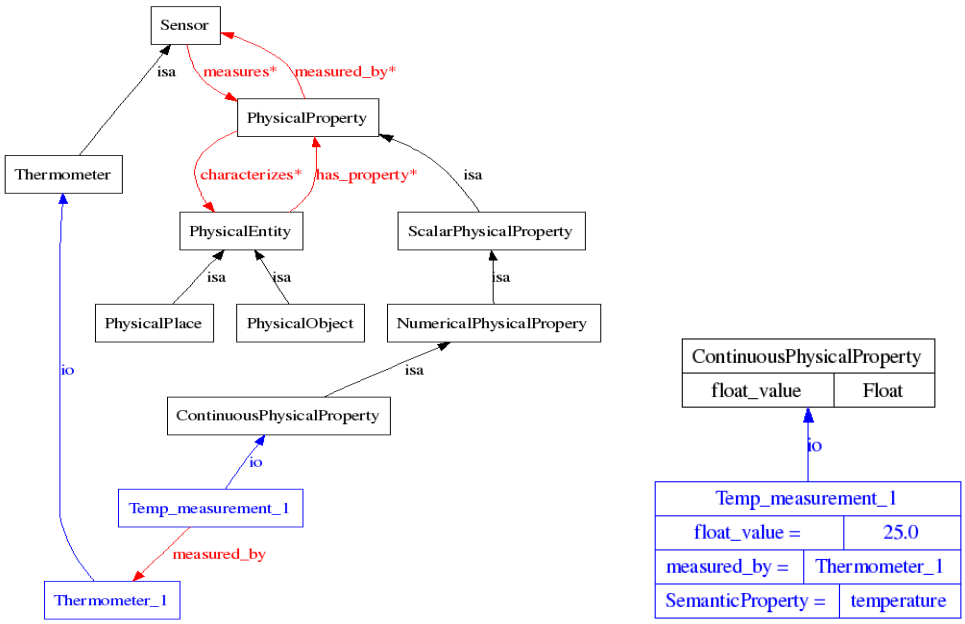

Figure 7: Instance of context representation.

3.3 CONTEXT QUERYING AND REASONING

Context information inference makes it possible to derive implicit context information from explicit one which is present in the context store. For example, in order to know whether a room is lightened although there is not any sensor assessing this environmental property, we might derive this information from the time of the day and the status of the shutters (open/closed). The reasoner provides support to context information inference. Our reasoner is built upon the Jena2 system ([6]). Jena2 includes a generic rule based inference engine together with configured rule sets for RDFS and for OWL languages.

4 Discussion

The presented work is still in progress. So far, we found it very convenient to rely on foundation technologies of the Semantic Web and Web Services. The main advantages are:

- Flexibility: we have redesigned the abstract model of context information (context ontologies) many times. These changes induced insignificant modifications on the

implementation of application services and sensor wrappers. Only the formulation of context information queries is impacted.

- Sharing and Reusability: some ontologies of context information are available "off the shelf" (e.g. location context). Current project is developing specific ontologies on time and geographical information. Such ontologies could be easily reused or adapted to fit our need.
- Openness: adopting Semantic Web will foster the accessibility of "local", "physical" services in pervasive computing from "virtual", "disembodied" Web services, and vice-versa.

In the very short term, we will come up with an operational version of a context-aware service infrastructure which will exhibit the following features:

- Context aware service lookup. Context information is used to complement service requests at the client side (e.g. use the location of a user to find the nearest restaurant), or at the server side (e.g. use the current date to filter out closed restaurants).
- Proactive services: Context information will be used to trigger specific services as predefined context information configurations occur.
- Context dependent services: This is the "conventional" use of context, where services adapt their behaviour by posting context queries and adjusting their computation based on the replies they receive to their queries.

With respect to existing work, as other research teams ([7], [8]) we aim at developing a generic context management environment suitable to the ubiquitous communications world. However we have departed from a centralized solution which we believe fails in capturing dependency of context on the task to achieve or on the service it will alter.

Adopting a decentralized solution where context description is to be shared among its consumers (services) and its producers (services or sensors) requires a semantic modeling of context information. Such semantic modeling is necessary to cope with the heterogeneity of the resulting partial models. Some work such as the CoBra system ([9]) has already addressed this issue, however with the limitations incurred by a centralized approach.

We interestingly pursued a path similar to Crowley et al. ([10])’s with the concept of "contextors". We adopted the Semantic Web ontology language OWL, to model context information. We believe this will facilitate interoperability with the Web Services world.

5 Conclusions

In this article we have described our approach for modeling and managing context information in a service oriented framework for Ambient Intelligence environments. This work is conducted within the IST-Amigo project ([11]). So far, our efforts have focused on context modeling and reasoning and our results reveal promising.

We still have to tackle the integration of our work into a complete service infrastructure which is developed in parallel within the Amigo project, and the assessment of the infrastructure in several real world applications.

We also plan to conduct additional work on context management, dealing with the:

- Merging and aggregating contextual information
- Learning of context and context descriptions
- Interoperability mechanisms with external context information sources, such as legacy mobile location servers

References

[1] Theodorakis, M., Analyti, A., Constantopoulos, P. and Spyratos, N. (1998) Context in information bases, in Intelligent agents, in the proceedings of the third IFCIS Conference on Cooperative Information Systems (CoopIS'98), New Yorks.

[2] Dey, A., Salber, D. and Abowd, G. (2001) A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. in Human-Computer Interaction vol. 16. p. 97-166.

[3] Schilit, B, Adams, N. and Want, R. (1994), Context-Aware Computing Applications IEEE Workshop on Mobile Computing Systems and Applications, Santa Cruz, California, USA, 1994.

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

[5] Flury, T., Privat, G., and Ramparany, F. (2004) OWL-Based location ontology for context-aware services in Proceeding of the worshop on Artificial Intelligence in Mobile Systems. Bristol - UK. p. 52-58.

[6] www.hpl.hp.com/semweb/jena2.htm.

[7] Gray, P and Salber, D. (2001) Modelling and using sensed context information in the design of interactive applications, in Proceedings of the 8th IFIP working conference on engineering for human-computer interaction (EHCI'01), Toronto, Canada, May 2001.

[8] Dey, A. (2000), Providing Architectural Support for Building Context-Aware Applications PhD Thesis, College of Computing, Georgia Institute of Technology, Dec. 2000

[9] Chen, H., Finin, T. and Joshi, A., 2003, The Role of the Semantic Web in pervasive Context-Aware Systems, Proceedings of ISWC 2003

[10] Crowley, J.L., Coutaz, J., Rey, G., and Reignier, P. (2002) Perceptual components for context aware computing. in International Conference on Ubiquitous Computing. Göteborg, Sweden. p. 117-134

[11] http://www.hitech-projects.com/euprojects/amigo/.