International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in; editorijoer@gmail.com

Vol.7., Issue.4, 2019
July-Aug

RESEARCH ARTICLE

ISSN: 2321-7758

AN INFORMATION MANAGEMENT INFRASTRUCTURE OF CONTEXT FOR AMBIENT INTELLIGENCE

ABDULLA MAHMOUD ALSHIBANI MOUSBAH
Faculty of Science, Bani Walid University
Bani Walid, Libya
Alshibani55555@hotmail.com
https://doi.org/10.33329/ijoer.74.30

ABDULLA MAHMOUD ALSHIBANI MOUSBAH

ABSTRACT

Ambient intelligence environments serve as an interface between applications and users. These applications must take into account the context in which the users evolve (the place, the social or hierarchical position or the activity for example) to adapt their behaviour. There must be a flow of information from the environment to the applications. These applications must be able to dynamically account for the arrival of new elements in the environment (users or devices), and context information from the environment must be able to reach incoming applications; these information flows cannot be determined in advance and must be built during execution. Existing context information management models do little to address this dynamic aspect of diffuse computing. We propose to use semantic web technologies to describe and retrieve this information Context information is expressed in RDF and described by OWL ontologies. These technologies, because they are based on the open world hypothesis, ensure the opening of the system and taking into account heterogeneous devices. We show that using a simple protocol that allows each of the devices and applications to exhibit on the network a model information of context that it produces or searches for and to identify itself, all the applications of the environment satisfy their needs for context information. In addition, the opening of ontology description languages allows the extension of context descriptions at any time and the alignment technologies ontologies allow the use of independently developed ontologies. We have implemented a component for context information management based on this model. Then we developed a distributed architecture where devices and applications embed this component and expose a model of the context information they are looking for or producing. We showed how this architecture allows to accept new components without interruption.

1.0 INTRODUCTION

According to Weiser [1] work, computers become invisible in our environment and are integrated into the objects of everyday life. When yesterday, users shared the same computer, the same computer resource, already today, the user has his personal computer. In addition, each of us is often equipped (at least) with a device (often portable) with communication capabilities with one or more networks and computing capabilities. IT is dispersed in the environment, it will become invisible until appear only in the background of our consciousness [1]. This orientation has given rise to several interpretations, which today constitute so many currents of computing: context-sensitive computing,

tangible computing, diffuse computing and ambient intelligence.

The development of context-aware computing is one of the major developments from the vision of Weiser. These new interactive systems are not intended to be used statically behind a desk, the interaction paradigm changes [2] and the environment becomes potentially dynamic (mobility situation, multi-users ...). The designers of these interactive systems do not may not be able to predict their execution environment from conception, it is a matter of designing able to collect and process information about this environment (information from context) to adapt their behavior during execution. By generalizing outside the problematic of interaction, the goal being to make computer systems more "intelligent", by to understand their environment of use (computer, physical, social ...). The intelligence ambient affirms the desire to make the computer invisible by diluting it in the environment daily. More precisely, it brings together several ideas: the ubiquity that refers to the presence in the environment of multiple distributed and interconnected devices (diffuse computing), the context which corresponds to the collection and processing of information concerning the user and his environment, intelligence, which characterizes the ability of the system to produce inferences from information about the user and his environment and to initiate actions based on these inferences, and the minimal natural interaction, that is, an interaction that is based on modalities and gestural repertoires familiar to the user, and which requires the minimum of attention from him.

In this work researcher introduce a scenario that simply illustrates the problem of ambient intelligence that we are studying and the utility of context information in these environments. Described here informally, it will be reconsidered at each stage to show how it is actually possible to implement it.

The main objective of this work is to develop a software infrastructure for developers of applications, devices and communicating objects to facilitate their access to context information. This infrastructure will have to integrate into environments of ambient intelligence, which imposes that it supports the opening in terms of dynamism and heterogeneity of the actors. It will be interesting to circumscribe notions of context and contextual information in the context of ambient intelligence and to determine the best way to access it as well as the means of representing it.

1.2 The context in ambient intelligence environments

The information that characterizes the environment, its actors and their (inter) actions, are essential in ambient intelligence environments. We will call them background information. They must be tapped into the environment, analysed and adapted. The environment can vary as much by the actors who evolve there or by the sources of information which it has that by the applications who populate it.

1.3 Perceive the environment, adapt to serve the user

Ambient intelligence environments are designed around a network on which applications, devices and sensors communicate and that can be open to the world of the Internet. Whether in a public, private or professional environment, ambient intelligence advocates building systems that transform physical spaces into intelligent interactive systems, that is to say, infrastructures capable of making the most of all the devices accessible by the network in order to provide maximum interaction wealth for the user combined with simplicity use.

1.4 Manage context information

In an ambient intelligence environment, applications must be able to acquire and reason information that characterizes the environment without worrying about the source of the information, without having knowledge of networks and different ways to query sources context information. To do this, we propose and study the principle of introduction in the ambient intelligence environments of middleware dedicated to context information management which will capitalize these functions to meet the needs of the applications of the environment. First, it will connect producers of context information with consumers and establishing an information flow from the first to the second. The goal being that a producer not be dedicated to a single consumer and that multiple

consumers can query the same producer of context information. Second, it will facilitate the development of sensitive devices context and sources of context information. It will have a central role in the circulation information without being implemented as a central management server context. On the contrary, we will try to provide each device and each application with functionalitiesmanage their own context information while ensuring their interoperability. The devices sensitive to the context will have to manipulate only one interface to interrogate the environment, and the sources of context information will only have to manipulate a single interface to disseminate the environmental information. Finally, in order to support the opening of ambient intelligence environments, relationships between consumers and producers Context information will need to be dynamically established without stopping the context information management system and without any operator intervention.

To summarize, we identified several requirements for context information management infrastructure:

1. Allow all context information producers to broadcast it.
2. Allow all context information consumer devices to find the sources appropriate.
3. Enable syntactic and semantic interoperability between different devices.
4. Allow the addition or removal of a new consumer or producer in the environment in a way that is transparent to the user

2.0 Experimental

Getting closer to an ambient intelligence environment is about building an environment in which users interact with devices and applications in a transparent and intuitive way. They will use scattered objects in the environment and not a single terminal. The environment is populated with smart devices and sensors, which are designed to provide functionality features will be accessed by a wifi network, allowing to build applications in aggregating the basic features.

In addition, this environment will have to keep the supposed problems of computer environments ambient, which we summarize as follows:

- devices and sensors are heterogeneous,
- the environment is dynamic and unpredictable,
- the needs of users are imprecise and evolving.

Thus, we offer a complete infrastructure for ambient intelligence environments the purpose of simplifying the design of ambient intelligence applications and make them more robust to real conditions. This infrastructure is essentially based on the work developed in this thesis work with regard to context management and the work developed by Mathieu Vallée on the flexible and dynamic composition of applications presented in [3].

Participated in the development of this demonstrator Mathieu Vallée, a trainee, Rémi Dupuis and myself. Rémi Dupuis focused on the development of services and applications and the integration of sensors in the infrastructure. We will present a realistic scenario of use in our environment in the coming sections.

Then we will detail the implementation of the infrastructure focusing on the role of the dedicated part to context information management in section upcoming section.

2.1 Scenario

Our demonstrator consists of four main applications that we have combined for create an ambient intelligence scenario called "Let's go skiing". It is as follows:

A resident of Grenoble, Tom, likes to ski with his friends, especially when the sun is at the rendezvous and that ski conditions are favorable. To watch for the emergence of favorable conditions, he uses the LiveMountain application. This application displays an image representing a mountain landscape, evolving according to the weather conditions and the availability of his friends. When the weather is good and that his friend Paul is available he can get more information using the app TangibleZoom. Approaching his PDA from the image projected on the wall, he has access to the details of the conditions

for different ski resorts. To enjoy a more comfortable reading, he can approach his PDA a larger screen that will display detailed information. If he is satisfied with the conditions posted, he will want to find a means of transport to get to the desired station. For that, he will use the iRiderNotification app that integrates a peer-to-peer carpool system, iRider, accessible from the home environments. He will begin by expressing his wishes by using a Web page. Then iRider searches for and negotiates appointments with car owners. If there is an interesting proposition, iRider Notification is looking for a way to inform Tom using a notification device. The latter is chosen dynamically according to the relevance of the proposition and context of Tom. For example, if Tom is busy and the proposal is judged uninteresting, there is no need to disturb it, so a device of peripheral notification can be used. However, when a very interesting proposal is available, iRider Notification will activate a bell. An offer from his friend Paul who works in a nearby station, will leave in an hour. Paul has already accepted this proposal, so Tom confirms his interest and is getting ready for his day in the mountains.

2.2 Demonstrator applications and devices

We will present the different devices, sensors and applications that we have equipped our environment to realize this demonstrator. They display on the network ontologies representing the context information they handle. These ontologies use other generic ontologies that we have made available on the network. That's why, we will dedicate a first time to present these generic ontologies.

2.3 Ontologies "useful" for the environment

We have designed six generic ontologies to allow different devices to describe their context model. In the interest of simplicity of realization and for the clarity of our explanation, we preferred to design ourselves these ontologies, rather than to use those available on Internet. However, we have taken particular care in the realization of these ontologies to reflect the heterogeneity of descriptions that can be found on the web.

2.4 Ontology: "Ski Conditions"

The first ontology (Figure 1) describes the meteorological conditions related to skiing, including sunshine, snow level and temperature.

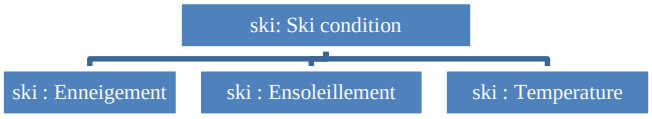

Figure 1: The ontology "ski condition".

2.5 Ontology: "Physical Context"

The second ontology (Figure 2) used in our demonstrator represents the physical context focusing on temperature, time and location

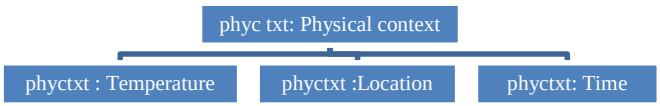

Figure 2: The ontology "ski condition".

2.6 Ontology: "Context"

The third ontology (Figure 3) describes the physical context and the social context as the subclass of the context. The second being "described by" relationships and "uses" activities.

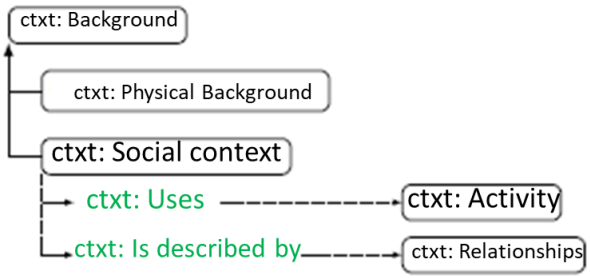

Figure 3 - The ontology "context".

2.7 Ontology: "PDA"

The ontology of Figure 4 is typically an ontology that could be used to describe the basic functionality of an electronic calendar. It tells us that "calendar", "notes" and "contacts" are subclasses of an "application". In addition, we learn that a "calendar" uses "appointments".

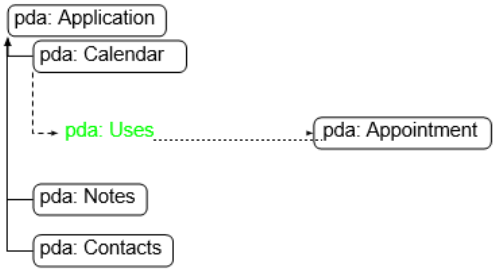

Figure 4 - The ontology "PDA".

2.8 Ontology "Weather"

The ontology "Weather" describes weather conditions including temperature, sunshine and the pressure (Figure 5).

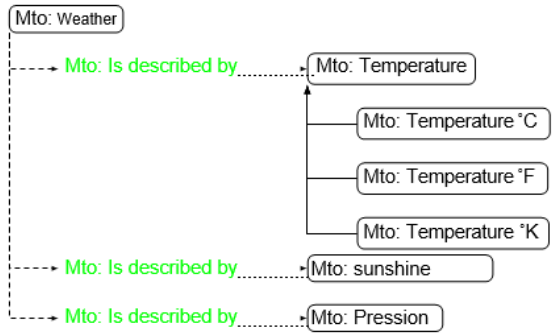

Figure 5: The ontology "Weather".

2.9 Ontology "City"

The last generic ontology of the environment is a hierarchy of cities (Figure 6).

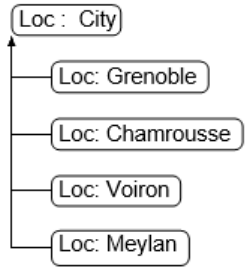

Figure 6 - The ontology "City".

2.11 LiveMountain

Tom uses Live Mountain to organize his ski days with Paul. The application allows him to monitor the occurrence of adequate conditions. It takes the form of a frame representing a table, which evolves according to the conditions. LiveMoutain queries web services specializing in weather and the ski conditions of the resort that interests Tom and Paul. If they are not available, the application will try to contact other sources of information taking into account their accuracy and confidence that it grants them to represent this information. So, Tom can realize in one fell swoop skiing conditions. When he sees that the weather looks good and that his friend Paul intends to go to the mountains, he can get more information through the TangibleZoom app. He just needs to approach his PDA table to have access to details of the conditions in different stations. For more comfort, he can move his PDA near a larger screen, on which the information appears so. If it is satisfied with the conditions and available for an output, the iRider application allows it to find a way to ride in the mountains with Paul. Depending on the interest of proposals and its current situation, Tom receives carpooling proposals on a device appropriate. When a more interesting proposal comes up, iRider triggers the AlertMe application, who seeks to alert Tom in a gradual manner, taking into account the timing of the proposal and the context of use. This is a carpooling offer proposed by a friend of Paul, who works in a nearby station, and Paul has already accepted this offer. Tom accepts him too and is getting ready for this day.

LiveMountain is a context information consumer, implements the contextConsumer class with the context model in Figure 7. To work, the LiveMountain application is interested in to the following environmental information:

- the weather defined by the Mto ontology of the Chamrousse localization defined in the Loc ontology in order to build the symbolic representation he displays,
- the user's activity as defined by the ctxt ontology, in order to transmit it to Remote LiveMoutain applications.
- the ski conditions defined by the ski ontology of the Chamrousse localization of the ontology Loc; to refine its symbolic representation.

To work, the TangibleZoom app looks at the following information about the environment:

- the precise location of the user in order to know which is the last interface device that has been approached,
- the location and activity of the other users present in the environment so as not to disrupt and be able to handle conflicts between applications.

To work, the AlertMe application looks at the following information about the environment:

- the location of the user to know which interface devices are available in this piece,
- the activity of the user to be able to adapt the level of alert and the means of the alert,
- the location and activity of the other users present in the environment so as not to disrupt and be able to handle conflicts between applications.

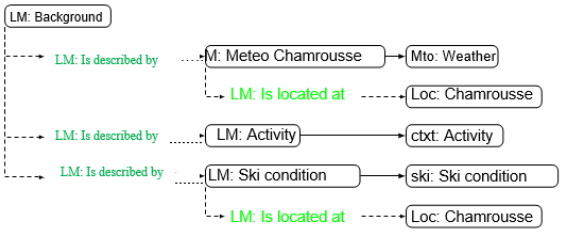

Figure 7 Ontology representing the context information sought by the "Live Mountain" application.

2.12 Service information "snow" Chamrousse

A web service accessible from the network provides information on ski conditions in Cham- rousse. For the purposes of experimentation, we have developed this web service that runs on a local server, as a replica of an existing web service. This allows us, through an administration console, to simulate the changes in ski conditions. This web service acts as a context information provider for our environment, especially for the LiveMountain application. It implements the ContextProducer class. It is able to provide the information of Figure 8, ie the snow as defined in the ski ontology for locating Chamrousse Loc.

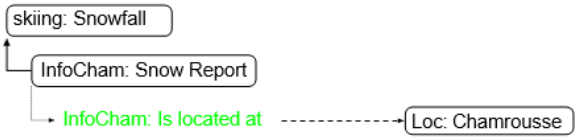

Figure 8 - Ontology representing contextual information provided by Chamrousse Snow Information Service.

2.13 Agenda on PDA

PDA is a provider of context information and therefore implements the Context-Producer class. It may, in particular, provide its timetable, defined as a subclass of PDA: Calendar, as shown in Figure 9. This information may give some indication of Tom's occupation and therefore the best way to send him a message.

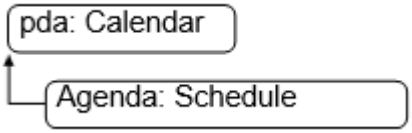

Figure 9 - Ontology used by the calendar application on the PDA to represent the context information it provides.

2.14 Grenoble Thermometer

The environment of the demonstrator is supposed to be located in Grenoble. This environment is equipped with a thermometer that provides the temperature, defined as subclass of phyctxt: Temperature, for the localization of Grenoble. A detail of the information provided by the thermometer can be seen in Figure 10. The thermometer, or rather its representative on the network, implements the ContextPro-class.

2.15 Balance sheet

A balance of the dependencies between the different models' context information of each of the devices is shown in figure 11.

Figure 10 - Ontology used to represent the context information produced by the device "Grenoble Thermometer".

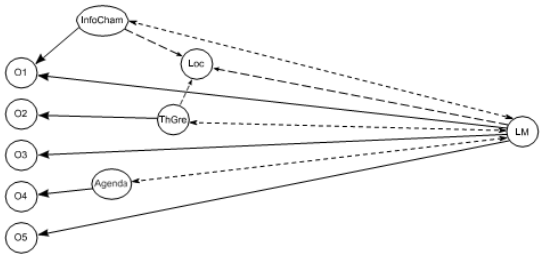

Figure 11 - Relationship between environmental ontologies.

3.0 AN INFRASTRUCTURE FOR AMBIENT INTELLIGENCE ENVIRONMENTS

This approach is to build a generic architecture for ambient intelligence environments. It is represented by Figure 12 which illustrates some elements of the scenario described in section 6.2. It is broken down into three elements, the service infrastructure, the context infrastructure and the composition infrastructure, which we will describe one by one.

3.1 Service Infrastructure

The service infrastructure provides the basic services for the proper functioning of our architecture in the environment. It allows discovery and communication between sensors and heterogeneous devices present in the environment. For example, we have shown in Figure 13 two of the services that can be used to notify the user: the Message Popup service triggers a popup window on Tom's PDA, and the Ambient Lamp service uses the light intensity of a environmental lamp to attract the attention of the user. The carpool service, iRider, allows you to notify the user each time a new proposal is available. Different notification systems will be used depending on the context of the users. Within this infrastructure, three different execution environments coexist: a J2EE platform, an OSGI platform and an Amigo.NET platform. Each of these environments hosts a runtime platform for JADE agents. The functional interoperability between the different services of the different execution environments is mainly ensured by the possible communication between these multi-agent platforms. In fact, the services provided by the context and composition infrastructures are implemented in the form of agents and each device and application has a representative on one of the multi-agent platforms.

Services can be a source of context, that is, they provide information about the environment that can be used by applications to adapt their behavior. In our scenario, we can consider that Tom's PDA has three different context sources. First, the Agenda provides information on Tom's occupations. Secondly, the RFID reader by reading RFID tags scattered in the environment, provides the location of the PDA and its carrier, Tom for example, and the likely interest of the device user of the environment carrying this tag. Finally, the MessagePopup application is also considered as a source of context. Indeed, when Tom clicks on a popup message, we can consider that it interests him.

An infrastructure management application provides the services needed by applications to take advantage of the ambient intelligence environment. It is divided into two parts: the context infrastructure that is of particular interest to us and the composition infrastructure.

3.2 Context Information Management Infrastructure

The context information management infrastructure is the multi-agent implementation of the solution we propose. Its purpose is to provide environmental information in the form of context information producers (CPs) to Context Information Consumers (CCs). The infrastructure tackles the problems of heterogeneity and dynamism by creating dynamic links between CCs and CPs that have not been designed to work together, and using ontology alignment technologies [4] to solve the problem of heterogeneity. A CC requiring the availability of Tom will first discover all the CPs in the environment (1) and obtain, by executing the protocol, the information they provide (2). This context information model is expressed with an OWL ontology [6]. When a context information model contains an unknown concept of a CC, it asks an ontology alignment server [5] to return all the matches between the model and its own model. If the CC finds compatibilities between the two models, it will send a SPARQL query [7], expressed with the vocabulary of the CC ontology and translated into the vocabulary of the CP ontology by the alignment server. This server will also translate the result of the

request, expressed as RDF triplets, from CP to CC. Applications, services and devices can be CP, CC or both.

3.3 Composition infrastructure

The composition infrastructure supports the assembly and reconfiguration of services into applications. It meets the requirements through flexible application management based on knowledge representation techniques and multi-agent systems [3]. An assistant agent, AA, triggers the composition by providing the description of the application to be made to the CA (1) dialing agent. This description expresses that the iRider service must be connected to a notification service. The latter is described abstractly without any reference to an existing service. To compose the application, the CA sends service requests to all SA supervisory agents (2). Each supervisor agent obtains descriptions of the available services of the infrastructure service (3), and evaluates their relevance for performing the desired task in the target application. In order to choose the appropriate notification service, a SA must know the availability of Tom. For this, it uses our context information management infrastructure. Once a correct composition is found, CA establishes a connection between the different services (4). Thus constructed, the system is completely adaptive: when the context changes or when a service appears or disappears, the assembly of the application is modified.

3.4 Matches between ontologies

To achieve running interoperability between the producers and consumers of context information present in the environment, the context infrastructure uses an alignment server to provide it with the matches between the ontologies describing the templates. background information manipulated by different devices. However, if two devices use models that refer to the same ontology, the context information consumer itself will calculate the degree of matching with the context information producer. This is the first case that we will detail. In the case where the devices exhibit models referring to different ontologies, the context information consumer will use an alignment to calculate its degree of compatibility with the producer. These alignments stored by the server are either calculated during execution or pre-filled. We will detail the alignments used for the proper functioning of our environment in a second time.

3.5 Without alignment

In order to calculate its degree of compatibility with a context information producer whose model uses a common ontology with those used to represent the consumer information model, the consumer will not need to use an alignment service but will look for the relations of subsumption or equivalence between the concepts of the different models. We illustrate this case with the correspondences between the ontology of the LiveMountain service and the ontology of the information service on Chamrousse (Figure 13).

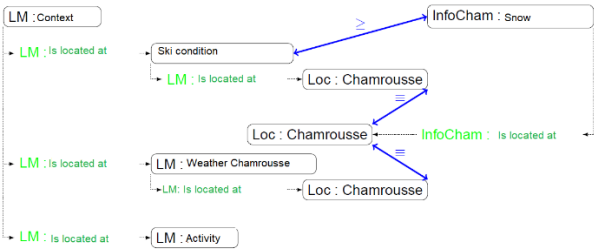

Figure 13: Correspondences between two ontologies with relations of subsumption and equivalence

3.6 Using alignments

In the case where one seeks matches between two concepts coming from two different ontologies, one uses an alignment. Alignments can be generated in two ways:

- Manually, that is, a user or an expert will construct all the matches between the two ontologies to be aligned.

- Automatically, that is to say that the correspondences between the two ontologies are calculated by a machine. There are different algorithms to perform this calculation which can be based for example on syntactic or semantic distances between the concepts.

In addition to these two types of alignment, we have distinguished geographical alignments that have the particularity of matching concepts representing places according to their geographical proximity.

3.7 Manual alignment

The first manual alignment is between the application ontology and the context ontology. It indicates that an appointment is a kind of activity (Figure 14). The second manual alignment is between the LiveMountain service context template and the Agenda context template. It indicates that an appointment is a kind of activity.

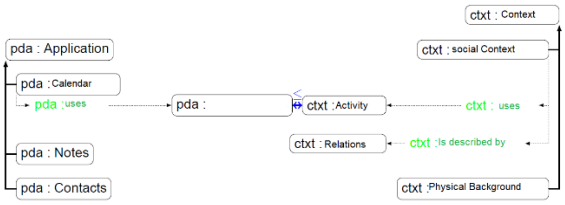

Figure 14: Manual alignment 1.

3.8 Automatic alignment

The automatic alignments we present are based on the syntactic distance between the concepts of the different ontologies. The first automatic alignment is between the weather ontology and the physical context ontology. It indicates an equivalence between the two temperature concepts of the two ontologies. The second automatic alignment is between the LiveMountain service model and the Grenoble thermometer service model. It indicates an equivalence between the two temperature concepts of the two ontologies(Figure 15).

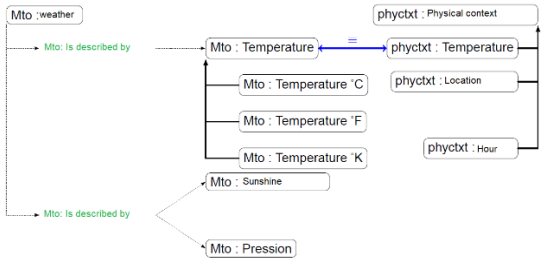

FIGURE 15 - Syntax Automatic Alignment 1

3.9 "Geographic" alignment

Using the geographical alignment of Figure 48, which represents the proximity of different cities around Grenoble, the LiveMountain service, a consumer of context information, can evaluate the confidence it gives to information from the Grenoble temperature service..

3.10 Demonstrator

This demonstrator was presented at the Ubicomp 2007 conference where we installed a light version for practical reasons related to transport, but which still offered many means of interaction for visitors. Its primary purpose was to show an infrastructure that could easily accommodate independent sensors, devices and applications and combine them to achieve the desired behaviors.

4.0 Installation

We will present the installation of our demonstrator by first describing its spatial organization, then the devices and sensors used. Finally, we will present the simulation tools that we have included in our demonstrator to make it richer; as well as administrative tools

4.1 Space organization

Figure 16 shows the spatial arrangement of the demonstrator. It was designed to form a space close and as close as possible to a domestic environment. On the one hand, we had a video projection surface. On the other side, we had four tables to install the different elements of the demonstrator. We grouped them into three groups as follows:

- Sensors and household devices are supposed to be present in the environment and visible by the user. Visitors can see, manipulate and interact with them. Those are elements (S1), (S2), (L), (T) and (PDA).

- The simulation consoles represent the services present outside the environment, present outside or on the Internet. Visitors outside the environment use it to simulate a change impacting the environment. These are the elements (L1) and (L2).

- The management consoles are accessible from outside the environment and allow visitors to monitor what is happening in details in the infrastructure (L3).

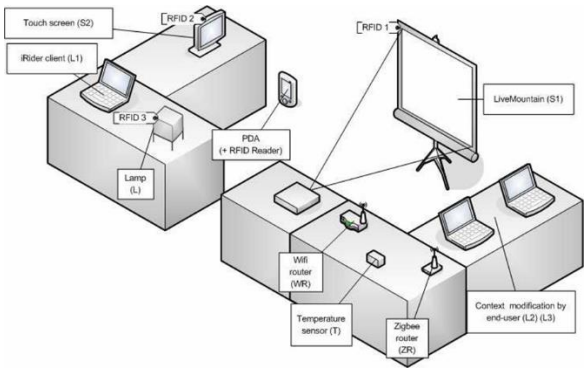

Figure 16 - Space organization of the demonstrator.

We will now detail the elements that make up each of the three groups

4.2 Sensors and home devices

The demonstrator consists of a set of intelligent devices that together form an intelligent environment. We distinguish three types of devices in our environment: Environment-integrated displays, small sensors and devices and wearable devices.

4.3 Screens integrated into the environment

The environment is equipped with two screens:

A wall screen (S1) displays the symbolic representation proposed by the LiveMountain application. The user will see it as a classic painting that has the particularity to evolve over time. As part of this demonstrator, we projected this image using a projector connected to a computer. The latter, connected to the network, executed only one application that collected the context information of the environment and displayed the corresponding image.

A touch panel (S2) used to display messages to the user. The messages are displayed as a pop-up and then they can be deleted or detailed if the user touch them. This screen can also be used to view an Internet page associated with a message. When there is no message, the screen displays pictures. For the realization of the demonstrator, the table is made with a touch screen connected to a computer that receives messages and displays

4.4 Small devices and sensors

The environment contains several small devices and sensors. These communicating objects use the ZigBee wireless communication standard. They are connected to the local network and other devices thanks to a ZigBee router (ZR) developed in our laboratory. Two types of communicating objects are used in our demonstrator:

The ambient lamp (L) is a lamp whose light and brightness can be controlled from a distance. It is used to transmit information to the user in a "soft" way. In our demonstrator, we use it to notify the user of the arrival of a message by modifying the lighting of the lamp according to the importance of the message.

The temperature sensors (T) periodically send the values collected on the network. We use them to indicate the temperature of the ski resort. In our environment, users can manipulate them and act on this parameter.

4.5 Portable personal devices

The PDA is the only device that is not an integral part of the environment since it is attached to a user. It has, for our demonstrator, three utilities:

- Store the user's personal information such as his agenda or preferences. The user can view or modify this information during the demonstration.

- Provide an interface to control or interact with the system. Unlike other screens of the environment dedicated to a single application, the PDA can be used as a computer classic to start an application or open a web page for example.

- Connect the user to objects in the environment using RFID tags. RFID tags (RFID1, RFID2, RFID3) are attached to objects in the environment. Thus, the user can interact with these objects by approaching the PDA. The user accesses the information relating to the object (a carpooling proposal by approaching the PDA of the lit lamp for example), moreover, this action is interpreted by the infrastructure as context information that can be the location of the user or his interest.

5.0 Simulation consoles

The weather service web provides the weather for the ski resort. This web service runs on the local network on the L2 computer, so visitors can simulate

a change in weather conditions through a simulation console.

The service, iRider, carpool allows the user to record his carpool search and to be notified of the corresponding proposals. This service runs on the L1 computer and is accessible through our local network. By using a web page, the visitor can register a new carpooling proposal and realize that the notification it generates will be different depending on the context of the user in the environment and its compatibility with the search of the user.

5.1 Consoles of administration

The main purpose of this demonstrator is to present the role of the infrastructure used. To provide the opportunity for visitors to understand how it works, we used several management consoles that mainly allow:

- Browse the different semantic descriptions used to represent the context information and the composition infrastructure.
- Show the discovery and selection of different producers of information by the consumer of information.
- To show the behavior of the composition infrastructure during the discovery and composition of services.

Table 1 - Characteristics of demonstrator devices

Designation	Type	OS	Processor	Memory	connectivity
PDA	PDA	Windows CE	500 Mhz	128 MB	Wifi+RFID
L1	Portable	Windows XP	1,6Ghz	1 GB	Wifi
L2	Portable	Windows XP	1,6Ghz	1GB	Wifi
L3	Portable	Windows XP	1,6Ghz	1GB	Wifi
S1	Portable	Windows XP	1,6Ghz	512MB	Wifi
S2	Portable	Windows XO	1,6Ghz	512MB	Wifi

5.3 Interaction with the demonstrator

This demonstrator was designed to offer the visitors maximum interactivity with the environment (figure 16) without guiding it in a scripted use to test the dynamic aspect of the infrastructures implemented. Our environment is divided into three spaces where visitors can interact with part of the demonstrator.

5.4 The LiveMountain space

The LiveMountain app continually displays information in the periphery of the user's attention. It consists of an information presentation service and three sources of context information. It displays a painting depicting a mountain landscape and symbolizing the conditions ski. It also displays the desire and availability of the local user that it infers from context information of the user it interprets. Context sources are potentially redundant: a web service providing weather conditions and snow cover, sensors for temperature and outdoor brightness, and shared calendars for users. . .

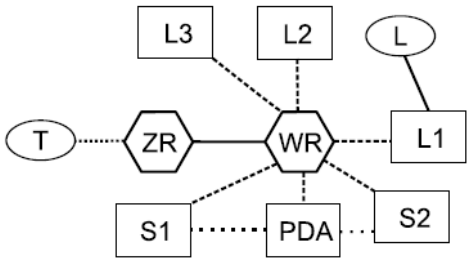

Figure 17 - Connectivity of the elements of the demonstrators

Figure 18 shows the table displayed by LiveMoutain and the simulation console corresponding to this situation.

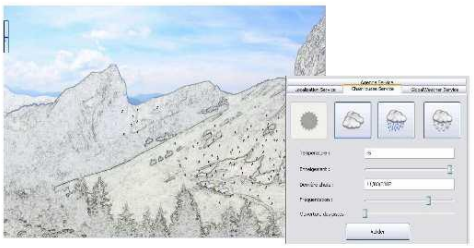

Figure 18 - Live Mountain Table and Simulation Console for Weather Information

5.5 The iRiderNotification Space

The iRiderNotification app uses a peer-to-peer carpooling app called iRider and a diffuse notification service for carpooling proposals. iRider allows you to save ridesharing offers. The user is informed by different notification devices disseminated in the environment that are dynamically selected by the user-based composition infrastructure and the relevance of the proposals to the user's activities and preferences. .

To illustrate the role of context and composition infrastructure in the iRiderNotification application, visitors have access to a local version of the iRider application that allows them to:

- Register their carpool preferences and wishes.
- Provide a carpooling proposal, and observe how the environment notifies the user of its relevance.

Visitors can also act on Tom's context, such as his availability, to influence the choice of the notification device.

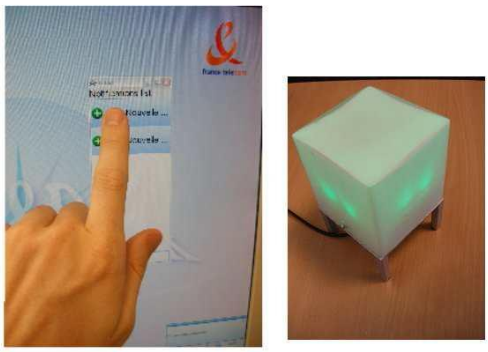

FIGURE 19 - Two notification devices: A notification pop-up (left) and the ambient lamp (right).

5.6 The TangibleZoom space

The TangibleZoom app offers widespread manipulation of information. Using a mobile device that the user moves from a notification device to a display device on which

he can look at the detail of the information that is notified to him. This application requires proximity sensing, which we do with RFID tags placed near devices in the environment and PDA with a tag reader, to ensure dynamic composition between devices, notification services and device. mobile.

Visitors experience the TangibleZoom app as if they lived in the environment. They carry the PDA and scan the associated RFID tag with the LiveMountain table (RFID1), the touch screen (RFID2) or the ambient lamp (RFID3). Thus, the PDA displays a web page relating to the scanned device and the application that uses this device. For example, Figure 54 shows us the web page that appears on the PDA after scanning the tag associated with the ambient lamp.

5.7 Balance sheet

At first, the realization of this demonstrator made it possible to evaluate the difficulty for a manufacturer to use the implementation of our solution and to embed it in a sensor, a service or an application. It appears that for devices limited in computing power as are the temperature sensors of our demonstrator, they need to have a representative in a platform multi-agent accessible on the network. However, this is not an insurmountable obstacle, and despite the multitude of execution environments used in our demonstrator (J2EE, OSGi, migo.NET and ZigBee), we have easily managed to implement a context management component for all consumers and producers of environmental context information since the development of the entire demonstrator required only a little over a month of work (for a reduced team of three developers).

In a second step, this experiment allowed to test the functioning and the robustness of our proposal in a real environment very little constrained. We did not find any major problem during the use of the demonstrator which proved to be fully functional. At the start of the demonstrator, the first execution of the protocol by the consumers of context information ends with their planned connection with the intended producer (s). Context information producers can be "hot" disabled and consumers discover new ones without any intervention on our part. The demonstrator has always provided all of its features. However, we have not been able to test the arrival of new devices and applications in the environment. The preparatory work of providing each consumer and producer with a context model

representing a different worldview for each of them, confirms that the infrastructure supports discovery and considers devices and applications. heterogeneous.

6.0 Conclusion

In this Article researcher proposed a representation and context information management for diffuse computing environments. The purpose of our work is to specifically consider the dynamic aspect of context management in this type of open and changing environment (users with devices enter the environment, move around ...). The dynamic aspect also requires reaching goals of openness to build our infrastructure, that is to say the capacity to enrich the environment with new features, typically when new devices with previously unknown features are available. In addition, we have endeavored to propose an infrastructure that requires minimal effort in its use by using standard technologies and requiring few adaptations. We attacked this problem by providing an infrastructure to help developers with the building devices and applications sensitive to context information. This infrastructure is based on: A distributed, component-based architecture that allows the addition of new devices that provide context information to applications at any point in the execution; A representation of the context that, using semantic web technologies (RDF, OWL and SPARQL), ensures interoperability between independently developed components due to the open nature of these technologies and their ability to incorporate new descriptions; A minimal protocol for acquiring context information using SPARQL, a query language for the semantic web; Correspondences between ontologies to deal with heterogeneous semantic descriptions. We have shown how these technologies interact in order to deal with problems of heterogeneity and dynamism. This proposal has been implemented using technologies all already available. The first research perspective is to test and show how our infrastructure meets the goal of scaling up. The main strength of our proposal lies in the correct assembly of these elements.

There are some limitations to this approach. Thus, like most solutions using ontologies, it is more flexible but certainly slower. For example, a semantic web-based application that uses our infrastructure will require reasoning, finding matches between ontologies, and calculating query responses, which can be very complex operations. Fortunately, the complexity of such a solution is not an insurmountable obstacle, especially since there are specific solutions to these problems. For example, [8] proposes a technique for compiling ontologies to speed up the process of responding to queries, which is called "matching in context". The use of approximate reasoning methods is another solution that can also be used in diffuse computing environments: that is to say, to propose a behavior to help the user in his task, but not necessarily the optimal behavior.

References

[1] M.Weiser. The computer for the twenty-first century. Scientific American, 256(3) :94 – 104, Sept. 1991.

[2] R. Aipperspach, B. Hooker, and A.Woodruff. The heterogeneous home. In UbiComp '08 : Proceedings of the 10th international conference on Ubiquitous computing, pages 222–231, New York, NY, USA, 2008. ACM.

[3] M. Vallée, F. Ramparany, and L. Vercouter. Flexible composition of smart device services. In Pervasive 2006, Dublin, Ireland, May 2006

[4] J. Euzenat. An API for ontology alignment. In S.McIlraith, D. Plexousakis, and F. van Harmelen, editors, Proc. 3rd International Semantic Web Conference (ISWC), volume 3298 of Lecture Notes in Computer Science, pages 698–712, Hiroshima (JP), 2004

[5] J. Euzenat. Alignment infrastructure for ontology mediation and other applications. In M. Hepp, A. Polleres, F. van Harmelen, and M. Genesereth, editors, Proc. 1st ICSOC international workshop on Mediation in semantic web services, Amsterdam (NL), pages 81–95, 2005.

[6] M. Dean and G. Schreiber. OWLWeb Ontology Language Reference. Recommendation, W3C, February 10 2004.

[7] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. Working draft, W3C, 2006.

[8] S. Ben Mokhtar, A. Kaul, N. Georgantas, and V. Issarny. Efficient semantic service discovery in pervasive computing environments. In Proc. 7th Middleware conference, volume 4290 of Lecture notes in computer science, pages 240–259, Melbourne (AU), 2006.