Semantic Middleware for Context Services Composition in Ubiquitous Computing

Abdelghani Chibani
Citypassenger France
14 avenue du Quebec, KI1, 91945, Courtaboeuf, Villebon France
achibani@citypassenger.com

Karim Djouani
LISSI Lab. Paris XII University
120-122 rue Paul Armangot 94400 - VITRY SUR SEINE, France
djouani@univ-paris12.fr

Yacine Amirat
LISSI Lab. Paris XII University
120-122 rue Paul Armangot 94400 - VITRY SUR SEINE, France
amirat@univ-paris12.fr

ABSTRACT
In this paper, we describe a semantic middleware which aims to provide ubiquitous computing applications with high level contextual knowledge. The proposed middleware architecture is designed around service agent entities, offering transparent and reusable services for context semantic discovery, capture and aggregation. Aggregation of contextual knowledge is modeled using services composition mechanism along with dynamic discovery of available service agents enabled through hierarchical and distributed context directories organization. We introduce also a new ontological model to describe contextual knowledge and services interfaces. The proposed middleware is applied to the design of travel organization service scenario, based on the composition of some ad hoc context services.

Categories and Subject Descriptors
D.2.11 **[Software Architectures]** Middleware, services

D.I.2.11 [**Distributed Artificial Intelligence**] Multi-agent systems, ubiquitous computing

I.2.9 **[Artificial Intelligence]** Ontologies, Semantic web

Keywords
Ubiquitous computing, Service Agent, Middleware, Context Awareness, Semantic Web Ontology, Semantic Matching.

1. INTRODUCTION
With the rapid growth of IT and networking technologies, ubiquitous computing (ubicomp) environments are becoming increasingly complex systems. Theses environments are characterized by multiple smart cooperating entities. To execute their tasks, theses entities require an autonomic and cooperative organization. Our work target aims at building context aware, intelligent and autonomous service agents that have the capability to follow mobile person in their journey, adapt its behavior to user context, and providing them necessary services. These context aware services take their full signification in several applications like task execution assistance for mobile people, ambient intelligence spaces, leisure and gaming communities, etc.

Context Awareness involves acquisition and reasoning about contextual information and acting according to context changes. We retain an interesting definition of context given by A. K. Dey: "*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*" [3]. Several research projects proposed semantic middleware approaches based on central context server entities eg; E-Wallet [4], SOCAM [5] and CoBrA [6]. Theses entities, which are based on ontological model of context, are dedicated to process context raw data, to store and provide these data in the form of high level and semantic contextual knowledge. Unfortunately, some of these projects do not take into account the semantic description and the dynamic availability of context sources. For example, in SOCAM a simple service discovery mechanism is proposed to create dynamic links between applications and context provider components.

To support a seamless, transparent access and processing of context knowledge in ubicomp environment we need to design a middleware layer mainly consisting of intelligent context service agents. These service agents should be semantically and automatically interoperable with heterogeneous services, devices, sensors and actuators of the environment rather than statically preprogrammed functionalities or drivers.

In this paper, we propose a semantic middleware for context aware computing. We used an agent-based architecture to provide in a flexible manner automatic mechanisms for discovery and composition of services. To ensure a semantic interoperability of the objects composing the ubiquitous space, this Framework is based on ontological knowledge model by using semantic Web ontology language OWL. The article is organized in the following way: section 2 describes context service agent architecture. Section 3 presents in detail the proposed context modeling approach. It presents also how context aggregation and personal service agents fulfils context services discovery and composition. Section 4 describes a validation scenario of a travel organization service for nomadic people.

MOBILWARE 2008, February 13-15, Innsbruck, Austria

DOI 10.4108/ICST.MOBILWARE2008.2486

2. SERVICE AGENT ARCHITECTURE

A service agent exhibits two main capabilities (autonomy and cooperation). A service agent provides autonomic services, which does not depend on an external controller. For example, accepting or refusing a service request from another agent if any of the privacy constraints is not matched or carrying out actions without needing any external stimulus. The other example could be advertising services in a given services directory, or sending notifications to another agent about context changes. Moreover, when the service agent cooperates, it is necessary to share with other agents an understandable semantic description of its identity, its capabilities, contextual knowledge, interaction protocol, execution parameters, security, privacy, service quality and cost.

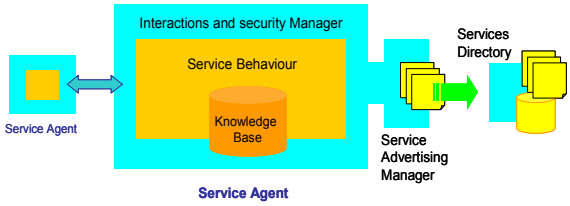

Figure 1. Context aware service agent architecture

A service agent is characterized mainly by the following behavioral architecture, (see Figure 1):

- Service Advertising Manager: It maintains the up-to-date semantic description of agent service. It allows the agent to publish its service description or remove it from service directories.
- Service Behavior Manager: It encapsulates the functional behavior (or the execution logic) of the service agent. We consider two types of functional behaviors. A reactive behavior is dedicated to agent's requests processing. We consider only requests for context access and requests for actuators control. Context capture agents and actuator agent implement this behavior, through specific drivers and API. The second type is a cognitive behavior which particularly concerns personal agents and aggregation agents. This behavior is a composition of existent reactive behavior which makes it possible for the agent to discover and invocate available services agents necessary to its tasks execution. The service behavior module consists also of a knowledge base, which allows reasoning and decision on context.
- Interactions and Security Manager: It is in charge of processing the messages received from other agents. These messages can be of three types: context information queries (queries requiring an immediate answer), subscription to contextual notifications, request to execute an action with an actuator or agent life cycle signaling messages.

2.1 Context service agent

A context service agent can be a simple agent acquiring contextual raw data from a given sensor. We call them in this case context capture agents (context sensor agent). Each capture agent is associated with a single context source type, while other context service agent can have a role of incorporating knowledge resulting from the fusion of various context sources, to provide contextual knowledge of higher level of abstraction. We call them in this case aggregation agents. Context knowledge security and privacy are handled inside each context service agent behavior through preconfigured inference rules. On top of context service agent infrastructure we find personal service agents. Theses agents are dedicated to fulfill tasks related to a specific application domain. The behavior of Personal agent is governed by a feedback loop: perception of the context, reasoning and acting to respond to context changes, through actuator service agents, example sending an alarm, switching light on/off, moving a robot wild sending an email, etc.). The services directory agents are used to provide the service agents with services discovery and advertising capabilities. Ontology server is proposed to manage ontology sharing and access to reasoning engines. Gateway agents enable service agents to interact with heterogeneous systems using protocols like web services.

2.2 Context aggregation service behavior

Context capture agents are dedicated to capture context from raw data sensors while context aggregation agent are dedicated to fusion of contextual knowledge collected form various context service agents. Aggregation agents differ from context capture agents in their complete independence to context resources. Consequently, when any context provider becomes unavailable, it is possible for the aggregation agent to substitute the unavailable service with other similar service, without canceling or suspending its execution. In our architecture, context aggregation agent exhibits two independent sub behaviors. The first sub behavior concerns contextual knowledge capture and processing, and the second sub behavior concerns the delivery of contextual knowledge to other agents.

In the first sub-behavior, the agent sends an advertisement to its associated directory agent in order to register its semantic description. Simultaneously, agent sends a discovery request to the directory agent to get appropriate context service agents. If discovery fails, the agent switches to active standby state until it finds available context sources. When the agent gets a selection of available context providers, it sends invocation request to acquire contextual knowledge. Invocation request is implemented according to the context access protocol provided by the middleware.

Parallel to context capture and processing behavior, the aggregation agent executes a second sub behavior to manage interrogation and subscription requests coming from other agents. This sub behavior is initiated just after a successful registration of the agent service description. It consists of a time delay for messages. Message content could be one of the following: (i) subscription to periodic notifications, (ii) interrogations with immediate answers, (iii) or messages of signalization. For each received message, the context agent starts by checking requestor access rights according to its own service policy. If the requester policy rights match with agent policy and required knowledge is available in its internal database, agent responds with the appropriate notification according to request parameters. Requests parameters and agent description are provided by our context awareness model.

Agent's context access requests processing, completed in the same manner by the sensor and aggregation agents, can be detailed according to two service invocation modes (subscription to notifications, interrogation requests). In the subscription to

notifications mode, request parameters, and the requestor agent identifier, are registered in context service agent subscribers list. The request parameter contains notification conditions (time or context value threshold or preconditions). Context service agent checks continuously for each subscriber if its request parameters match to the actual context before sending the required contextual knowledge notification. Conversely, in the interrogation mode, the context service agent checks the availability of the contextual knowledge required in its internal data base then sends to the requestor agent the appropriate answer.

When all context service agents related to aggregation agent are out of service, the aggregation agent switches to standby state until the providers become available again. When standby period is expired, the aggregation agent can consider that computing environment had a major breakdown and decides to end its service execution by sending an unsubscribe-me message to its directory agent.

3. CONTEXT MODELLING

In the state of the art we can find different modeling approaches [2]. These models present context as a set of attributes (data) that could be captured from different disseminated sensors using, for example, key value representation or semantic knowledge representation through RDF predicates or ontologies. For us, an efficient context modeling technique should exhibit characteristics like flexibility, extensibility, expressiveness, and reasoning which are very necessary to enable context awareness. These characteristics will make the system able to identify, to describe and to share any complex contextual information with a homogeneous representation.

1. Table of context modeling approaches in the state of the art

Approach	Extensibility	homogeneity	Expressive -ness	inference
key-value	-	-	-	-
Graphical	+	-	-	-
Predicates	+	+	-	-
Ontologies	++	++	++	++

Table 1 provides a summary of the required characteristics of context modeling approach and their pertinence for our architecture. Extensibility and expressiveness are important in the way that they make it possible to enrich the description and the representation of any new context information during run time without review of the representation model. For example, a person activity which is described by title, start and end time, location name, could be extended to integrate full description of location, participants profiles and activity type. According to the other modeling in approaches, denoted in table 1, ontologies are so far interesting because they provide an extensible and expressive model associated to reasoning capabilities. Ontologies provide also with high capability to take into account new contextual attributes during system life cycle. In our work, we intend to enable semantic interoperability of context service agents. For that purpose, we propose an OWL ontology called UCOUS (User Context Ontology for Ubiquitous Services), which can be distinguished from the existing ontologies by its capability to provide in the same representation both contextual knowledge description and context services providing access to these knowledge. In this way, we can get a homogeneous model based on common classes that allows describing in the same manner contextual knowledge and access services. Compared to LARKS Frame model (for Agents capabilities description) and OWL-S ontology (for semantic web services description), our model offers more expressiveness in the description of context service agent's capabilities. The contextual knowledge discovery and access is carried out in seamless manner through semantic matching of desired context services with context services being published in the environment.

3.1 Context services description

Figures 3 and 4 illustrate part of UCOUS context services Ontology main classes. This ontology is considered as a semantic meta layer on top contextual knowledge, which allows context agents to advertise semantic description of the contextual knowledge they provide, in a format understandable by any service agent of the ubicomp environment. Service agents which are continuously seeking for contextual knowledge will use this published descriptions to identify and to select the best context provider and how to interact with it.

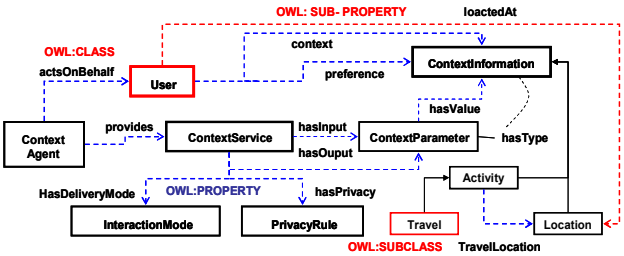

Figure 2. Context services ontology OWL classes and properties

Context services ontology core classes are "*ContextService*" and "Context Agent", (see Figure 3). These two classes are related by an owl object property (denoted by "provides" in line #01#) which means context agent provides a context service. ContextService class is described using five owl object properties (hasInput, hasOutput, hasType, hasDeliveryMode and hasPrivacy) which refer to the following ontology owl classes: "ContextParameter", "InteractionMode", and "PrivacyRule". ContextParameter class defines either input our output knowledge of the context service, eg; line #08# "hasOuput" denotes an object property defining an output Context Parameter, while Context Information type is modeled by a Data Type property, which refers to classes being inserted in the context knowledge ontology.

Interaction mode is modeled by owl class, denoted in line #06# of Figure 4. In this class, we took into account two types of interaction modes, closely related to context sources update frequency. The first type is notifications mode, denoted, in Figure 4, by Notifications owl class in line #05#, which refers to knowledge requiring a high frequency of notification, e.g.; temperature, location of a mobile robot or a user, weather, user activity, etc. We distinguish two sub types of notifications: (i) periodic notification, denoted by Periodic Class (Line #08# of Figure 4), which is carried out at regular time intervals and (ii) notification on change of state, which is carried out according to two cases. The first case, every time when a context change occurs (when the context provider captures a new context parameter value). The second case concerns a precondition, expressing if the

value of a context parameter reached a given threshold. The second type denoted QueryAnswer by is related to knowledge requiring low level notifications frequency, e.g. user profile, preferences, user contacts, user location description, user devices, etc.

```
01<owl:ObjectProperty rdf:ID="provides">
02<rdfs:range rdf:resource="#ContextService"/>
03<rdfs:domain rdf:resource="#ContextAgent"/>
04</owl:ObjectProperty>
05<owl:Class rdf:ID="ContextService">
06 <rdfs:subClassOf rdf:resource="../owl#Thing"/>
07</owl:Class>
08<owl:ObjectProperty rdf:ID="hasOuput">
09<rdfs:domain rdf:resource="#ContextService"/>
10<rdfs:range rdf:resource="#ContextParameter"/>
11</owl:ObjectProperty>
12<owl:Class rdf:ID="ContextParameter"/>
13<owl:DatatypeProperty rdf:ID="hasContextType">
14<rdfs:domain rdf:resource="#ContextParameter"/>
15<rdf:type rdf:resource=" owl#FunctionalProperty"/>
16</owl:DatatypeProperty>
17<owl:DatatypeProperty rdf:ID="hasPrivacy">
18<rdfs:domain rdf:resource="#ContextService"/>
19<rdf:type rdf:resource=" owl#PrivacyRule"/>
20</owl:DatatypeProperty>
```

Figure 3. Partial OWL/RDF representation of context services ontology

```
01<owl:ObjectProperty rdf:ID=" hasDeliveryMode ">
02<rdfs:domain rdf:resource="#ContextService"/>
03<rdfs:range rdf:resource="#ContextProvisionMode "/>
04</owl:ObjectProperty>
05<owl:Class rdf:about="#Notifications">
06<rdfs:subClassOf rdf:resource="#InteractionMode"/>
07</owl:Class>
08<owl:Class rdf:ID="Perdiodic">
09<rdfs:subClassOf>
10<owl:Class rdf:about="#Notifications"/>
11</rdfs:subClassOf>
12</owl:Class>
13<owl:ObjectProperty rdf:ID="onChangeContextParameter">
14<rdfs:range rdf:resource="#ContextParameter"/>
15<rdfs:domain rdf:resource="#Perdiodic"/>
16</owl:ObjectProperty>
17<owl:Class rdf:about="#QueryAnswer">
18<rdfs:subClassOf rdf:resource="#InteractionMode"/>
19</owl:Class>
```

Figure 4. Partial OWL/RDF representation of context services interaction modes

3.2 Context knowledge description

In our model, contextual knowledge is captured or inferred from several heterogeneous sensors, using different data structures, can be associated to inference rules language. This ontology makes it possible for context aware services to derive user situation in a specific time. In our Ontology, Contextual Knowledge, (denoted in Figure 2 by Context Parameter and Context Information classes) is modeled using three extensible concepts; *ContextInformation*, *User*, *Time*, represented in OWL/RDF, (see Figure 5). These concepts are sufficient to describe any user's contextual attribute semantics. In fact, these interrelated concepts offer three levels of user context abstraction:

- Relational level: It consists of set of predicates describing relations between user, structure of any concept characterizing user's context, time stamp of context capture or inference, and time validity of the capture or inferred contextual knowledge.
- Structural level: it provides extensible and complete class taxonomy, providing semantic description of context concept properties.
- User preferences level: It consists of predicates describing; what is the preferred (or desired) user's context.

```
01<owl:Class rdf:ID="User"/>
02<owl:Class rdf:ID="ContextInformation"/>
03<owl:ObjectProperty rdf:ID="createdAt">
04<rdfs:domain
rdf:resource="ContextInformation"/>
05   <rdfs:range   rdf:resource="   /damltime/
time-entry.owlInstantThing"/>
06</owl:ObjectProperty>
07<owl:ObjectProperty rdf:ID="trueAt">
08<rdfs:range rdf:resource=" /damltime/time-
entry.owl#InstantThing"/>
09<rdfs:domain
rdf:resource="ContextInformation"/>
10</owl:ObjectProperty>
11<owl:ObjectProperty rdf:ID="contextOf">
12<rdfs:domain
rdf:resource="ContextInformation"/>
13</owl:ObjectProperty>
14<owl:ObjectProperty rdf:ID="context">
15<rdfs:range
rdf:resource="ContextInformation"/>
16<rdfs:domain rdf:resource="User"/>
17</owl:ObjectProperty>
18<owl:ObjectProperty rdf:ID="preference">
19<rdfs:range
rdf:resource="ContextInformation"/>
20<rdfs:domain rdf:resource="User"/>
21</owl:ObjectProperty>
```

Figure 5. Partial OWL/RDF representation of contextual knowledge ontology

These three levels of description are represented in figure 5 using the following classes:

- Time: This class describes any timestamp based information, which enables temporal reasoning related to context information. In order to re-use the maximum of classes existing in OWL standard ontologies, Time class is modelled, in line #08#, by InstantThing; a class imported from OWL-Time Ontology [14].
- ContextInformation: This class makes it possible to describe any contextual attribute properties. It is modeled in our ontology by three object properties: createdAt and trueAt properties (denoted by <OWL:ObjectProperty> tag in lines #03# and #04#),

which value range (denoted by <rdfs:range> tag) refers to IntantThing class instances (denoted by #08#), while contextOf object property (denoted by line #11#) refers to User class Instances in the ontology.

- User: This class describes a system's user. User is defined in our ontology by two object properties: Context and Preference (denoted by <OWL:ObjectProperty> tag in lines #14# and #18#), which value range (denoted by <rdfs:range> tag) refers to ContextInformation class and subclasses instances. At this level of description we describe only the relationship between user and context while we avoided defining the relationship between system's user concept and physical person concept in order to leave user as an abstraction of any entity that could use system services for example a software agent, or a mobile robot.

3.3 Context services composition schema

In our Framework context aggregation (or fusion) diagram is represented by an acyclic directed graph. This graph makes it possible to model aggregation with arcs representing inference rules used to aggregate contextual facts. The arcs can be differentiated by their priority. While the starting nodes which corresponds to captured (or factual) contextual knowledge, (inference rule premises), the arrival and intermediate nodes correspond to the aggregated contextual knowledge (inference rule results).

Concretely, this directed acyclic graph corresponds to contextual service composition schema. In figure 6, we show an example of how to derive next user travel (C7 node) by aggregating current GPS location (C2 node) or address, activities schedules from her agenda (C1 node). C1 and C2 are differentiated using the level of priority where dashed arc has low level priority. At the initialization of agent context capture behavior, this last load the composition schema and translate it into services discovery requests to be sent to the associated directory agent. Each request contains a semantic description of required contextual knowledge which could satisfy an aggregation rule. Directory agent processes each request by executing a semantic matching operation between the request description and services description registered in its directory. The result of this operation is a selection of agent service identifiers available and their respective descriptions.

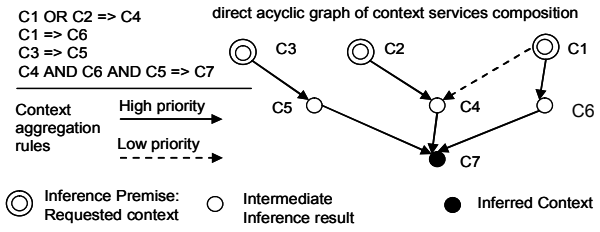

Figure 6. initial aggregation graph

At the reception of the response from agent directory, the aggregation agent updates its diagram of aggregation by adding to each node respectively it associated service identifier and description. For each starting node without corresponding service, the aggregation agent removes its corresponding arc (Figure 7). Automatically, after a successful services discovery, the agent starts to request service corresponding to each aggregation node of the diagram. For each node the agent of aggregation identifies service domain and request mode of the targeted service.

Once, the aggregation agent sends invocation request (subscription or interrogation) to the corresponding, it switches its state to wait for notification and/or answer. If the targeted service belongs to a private domain, the aggregation agent encloses to its request privacy policy (access rights).

Once an answer or notification is received, its content is added to the internal data base of the aggregation agent and supplied to the inference engine in order to derive new contextual knowledge and check its coherency to the overall model.

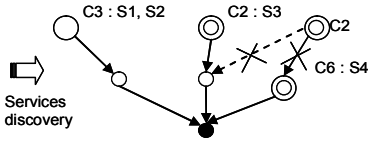

Figure 7. Updated composition schema after successful agent services discovery

Due to dynamic nature of ubicomp services, the capture tasks can fail at any time. For example, it is possible that one or more agents of context of the diagram of composition become out of service. Consequently, the aggregation agent reacts to these fault by carrying out a rediscovery request in order to update its composition schema (Figure 8).

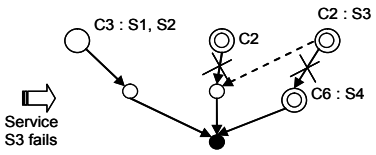

Figure 8. Composition schema modification after service failure

All failing or non authorized agent services are respectively marked as out of service or removed from the composition schema to avoid their rediscovery next time if directory agent is not up-to-date. Discovery loop could be restricted to avoid that aggregation agent indefinitely sends discovery requests.

3.4 Semantic discovery of context services.

OWL ontologies integrates Description Logics (DL) reasoning capabilities, which are useful for context processing and interpretation [10] [11]. An important reasoning capability in our work is semantic matching of ontologies [12]. It allows an agent to identify relevant classes from sets of example instances according to the class description (matching template). This function is useful to implement a semantic discovery of context services. When an agent needs a specific contextual knowledge, it sends a service discovery request to the closest directory agent available in the computing environment. This request includes a description of the required contextual knowledge in the form of a service description according to our contextual services ontology. Directory service agent, which is connected to OWL based reasoning engine, will process then the discovery request by executing a semantic matching between registered services description in its internal directory data base and the received request description. Compared to LARKS or DAML-S Matchmaker [12] [13], our discovery mechanism combines both contextual knowledge and services ontologies to process an

efficient matching among contextual services agents by using more than one service directory. Our ontologies are used here to extend the structure of services directory being held by a directory service agent. The later offers two types of services: services description inside the directory and semantic matching with agent requests.

4. SCENARIO

The scenario we studied concerns the implementation of personal agent dedicated to assist sales people when traveling by automatically discharging them from travel organization tasks. This agent which is integrated in the Citypassenger services platform, will use contextual knowledge to automate travel planning and booking. This agent makes it possible to offer the users two services. The first service consists of seeking and proposing best flights offers corresponding to user travel and activity planning. The second service consists of booking flights automatically selected. The execution of the behavior of the Travel Organizer agent requires the availability of the following context services infrastructure: time, diary, localization, current activity, next location, profile of the user and travel planning. The travel organizer interacts with user through his PDA web browser GUI interface. This interface makes it possible to associate the activities scheduled in the user agenda with contextual notifications corresponding to its localization and its travel planning as well as the flight offers. For each travel, a link "*to consult*" is displayed automatically in the PDA agenda to notify user about flights corresponding to his/her next travel. The user can then select a flight and launch booking process. The booking form is filed by the Travel Agent from contextual knowledge except for payment order which must be validated at the end by the user. The various tests with the users show that travel organizer personal service takes into account user context changes, and carries out context aware actions which are flight booking and user notification by a GUI interface, without requiring any focus from user. Compared to booking service portal like Opodo™ , which at least requires more then ten interactions and a complete focus of the user during more then 15 minutes, to search and book a flight for his next travel, our scenario requires only two interactions without any focus by the user, Figure 9.

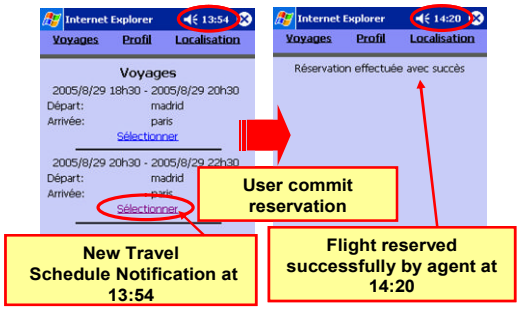

Figure 9. Semi automatic flight reservation after travel context change event.

5. CONCLUSION

In this paper, we have presented a service agent approach for context services composition in ubicomp environments. This approach is based on the use of ontologies to model context services description and discovery. Aggregation of contextual knowledge is modeled as a composition service schema through using direct acyclic graphs while dynamic discovery of services is used to check the availability of service during run time. We have used the proposed architecture in the implementation of a context aware travel organizer personal service. Various tests, in particular consistency check, have shown the efficiency of the proposed approach. The ongoing work concerns the integration of the middleware on top of OSGI a standard service platform for residential environments .

6. REFERENCES

[1] Baldauf, M., Dustdar, S. Rosenberg, F., "A Survey on Context-Aware Systems". *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4): 263 – 277, 2007.

[2] Strang, T., Linnhoff-Popien, C. "A Context Modeling Survey. *In Proceedings of Workshop on Advanced Context Modelling, Reasoning and Management*, 2004.

[3] Dey, A. K. "Providing Architectural Support for Building Context-Aware Applications", PhD thesis, Dec. 2000

[4] Gandon, F., Sadeh, N., Semantic Web Technologies to Reconcile Privacy and Context Awareness. Web Semantics Journal, 1 (3): 241-260, 2004 .

[5] Wang, X.H., Zhang, D.Q., Gu,T., Pung, H.K., OWL encoded context ontology (CONON), Ontology Based Context Modeling and Reasoning using OWL, Second IEEE Conference on Pervasive Computing and Communications.

[6] CoBrA: Context Broker Architecture, an intelligent broker for context aware smart spaces, http://cobra.umbc.edu/

[7] Román, M., Hess, C.K., Cerqueira et al : A Middleware Infrastructure to Enable Active Spaces. IEEE Pervasive Computing, 1(4): 74-83, October-December 2002.

[8] Rey G., Coutaz, J., The Contextor Infrastructure for Context-Aware Computing Component-oriented Approaches to Context-aware Computing ECOOP'04, Oslo, 14 june 2004

[9] Chibani, A., Semantic web oriented agent's middleware for context aware ubicomp applications. Phd Thesis, University of Paris 12, 2006

[10] Dean, M., Schreiber, G., OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/

[11] Horrocks, I., Patel-Schneider,P.F., Boley, H., Tabet,S., Grosof,B., Dean, M., SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004,

[12] Paolucci, M., Kawamura, T., Payne T., and Sycara K.. Semantic Matching of Web Services Capabilities. Proceedings of ISWC Conference, June 2002.

[13] Sycara K., Lu J., Klusch, M., and Widoff, S., Matchmaking Among Heterogeneous Agents on the Internet., Proceedings AAAI Spring Symposium on Intelligent Agents in Cyberspace, Stanford, USA, 1999.

[14] OWL-Time Ontology, http://www.isi.edu/~pan/damltime