Context Management in Mobile Environments: a Semantic Approach

Alejandro Cadenas
Telefónica I+D
Emilio Vargas, 6
28043 Madrid, Spain
cadenas@tid.es

Carlos Ruiz
iSOCO
Pedro Valdivia, 10
28006 Madrid, Spain
cruiz@isoco.com

Iker Larizgoitia
Universidad de Deusto
Av. Universidades, 24
48007 Bilbao, Spain
ilarizgo@tecnologico.deusto.es

Raúl García-Castro
Universidad Politécnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte, Spain
rgarcia@fi.upm.es

Carlos Lamsfus
CICtourGUNE
Paseo Mikeletegi, 56 - 201
20009 San Sebastián, Spain
carloslamsfus@tourgune.org

Iñaki Vázquez
Universidad de Deusto
Av. Universidades, 24
48007 Bilbao, Spain
ivazquez@eside.deusto.es

Marta González
Robotiker-Tecnalia
Parque Tecnológico de Vizcaya, #202
48170 Zamudio, Spain
marta@robotiker.es

David Martín
CICtourGUNE
Paseo Mikeletegi, 56 - 201
20009 San Sebastián, Spain
davidmartin@tourgune.org

María Poveda
Universidad Politécnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte, Spain
mpoveda@delicias.dia.fi.upm.es

ABSTRACT
This paper presents a first draft of a context management model and architecture in the scope of mobile end-user services, paying special attention to mobile scenarios and specific mobility environments. The paper describes our notion of context and presents the process followed for developing the ontologies for representing context, providing an overview of the first version of these ontologies. Besides, we propose an architecture for managing context in mobile environments, including high-level descriptions of the components that compose it, as well as some low level details regarding the contextual interfaces involved. To show the way all these components interoperate and the advantages of the defined architecture and semantic model, we describe a use case with specific implementation details.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Software Architectures – *domain-specific architectures*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *semantic networks*; C.2.2 [**Computer Communication Networks**]: Network Architecture and Design – *Wireless communication.*

General Terms
Design, Experimentation, Standardization, Languages, Theory.

Proceedings of the 1st Workshop on Context, Information and Ontologies, June 1, 2009, Heraklion, Greece

Keywords
Context, context management, context ontology, context-aware services.

1. INTRODUCTION
Context-aware applications are one of the coming paradigms in telecommunication environments, going from the strictly research environment to the commercial deployment arena. The amount of devices around the user that can capture information about his/her context is progressively increasing due to the enhanced hardware and software functionalities of the mobile communication terminals as well as to specific sensor devices. And, on the other hand, the diversity of the ways that the user context can be used by different services or context consumers is growing fast. This is due to the increasing number of service delivery or provider entities that can be accessed by the user.

Such environment makes the perfect timing to conduct a rigorous multidisciplinary research project, focused both on the modelling area as well as on the architectural analysis, such as the directions presented in this paper, that outlines the work in progress of the project.

This work specifically focuses on mobile environments, where the user will own a mobile terminal that will behave both as source of context information and consumer of such information processed, via the services that may be executed on the mobile terminal itself. The global use case is the *prosumer* scenario, in which the user both generates and consumes services and context information. In this environment, the user will move from one specific situation to another, immersed in his/her own changing context.

An important research work has been performed in the area of services' context, due to the fact that services directly provided to

the user has been labelled as "services mIO!" and defined as "*a context-aware software entity with an user interface that is executed in a network or user's terminal server, and that provides value to the user*"

Such user's context will need to be modelled to be able to process and use it to configure, discover, executed and enhance different services that the user may request or that the user may even publish from his/her own device for others to consume. Such model structure needs to be fully defined as a first step to build such ambitious final objective and, to this end, we will follow a semantic approach because of the advantages provided by ontologies. A fully methodological iterative stepwise procedure is followed, involving a wide set of experts not only on ontological engineering but also on very different domains of the mobility environment (such as experts in mobility infrastructure, service development, device connectivity, interface design, etc.). This collaborative approach will produce a reliable and consensual model.

Once the model is defined, an architecture is required to obtain, aggregate, process, and progress context information as well as to execute the corresponding services based on such processing results of aggregated context. Such architecture will include all the necessary modules to perform such actions in a modular and interoperable way, taking as a premise the computing and storing limitations of mobile terminals. The objective is to propose a contextual framework that is flexible and fully future-proof to support any requirement, service or scenario that may be defined in the future. The state-of-the art is analysed to make use of it, and the specific enhancements and designs are performed based on the particular characteristics of the mobile environment.

And, finally, in order to show the execution procedures and the actual capabilities and flexibility of both the model and the architecture proposed, a use case is described and presented at a technical level. The interaction between the captured user's context and the management infrastructure is described, including interactions among the different entities described in previous sections.

The following sections of the paper are structured as follows. Section 2 describes the definition of context and the first version of the context ontologies produced. Section 3 provides the description of the context management architecture, and section 4 provides the details of the use case to show the interaction between elements. Finally, the conclusions and future work lines are provided.

2. THE mIO! ONTOLOGY NETWORK

In our work, we adapt Dey's definition of context [1] to cover our needs for modelling user-centric context. This way, we define context as any information that can be used to characterise the situation of an end user with the goal of selecting the services that are relevant to this situation and to adapt their functionality. This information will include the user characteristics in terms of profiles and preferences, near and remote environments, devices and services available, as well as any knowledge based on user interaction.

We aim to reuse as much as possible existing contextual models that are supported by a significant number of practitioners and to cover the evolution of these models as well as of our ontologies. Therefore, we will not develop a single ontology but an ontology network, that is, a collection of ontologies related together via a variety of different relationships such as mapping, modularization, version, and dependency relationships [2].

Currently, ontologies for modelling context are still in an experimental phase; they have been defined for different specific uses and cover different domains. Hence, no consensual model exists that can be broadly reused for modelling context in applications. Furthermore, even if there have been plenty of efforts for developing context ontologies, only few of them are available to be studied in detail and reused; these are the CoDAMoS [3], GUMO [4] and SOUPA/COBRA-ONT [5] ontologies.

2.1 Methodological Approach

In this section we describe the methodological approach adopted to build the mIO! ontology network. This approach follows the NeOn methodology for developing ontology networks [6]. The mIO! ontology network will be implemented in three consecutive iterations, each of them providing a working prototype of the ontology network. Clearly, modelling the complex domain of context will involve modelling the different subdomains that compose it.

At the moment of writing this paper, we have completed the first iteration of the ontology network development. In this iteration, due to time restrictions, our goal was to obtain a first set of ontology requirements and a first prototype of the ontology network that could be used in early stages of the project. In the next iterations this prototype will evolve as well as the ontology requirements, producing improved versions of the ontology network. The main activities that we carried out in this first iteration of the ontology development process are the following:

- **Ontology specification.** We defined the scope of the ontology network, its intended users, and the ontology requirements. For extracting these requirements we involved end users and experts in each of the subdomains covered by our definition of context; this way, for each subdomain we obtained as requirements a mix of domain characteristics in Natural Language and competency questions.
- **Scheduling.** We identified the different activities to carry out during the development process and organized them in time according to the existing requirements and restrictions. In our case, the development will comprise three iterations where incremental prototypes of the ontology will be produced.
- **Ontological resource reuse.** We searched existing ontologies and selected those that a) covered parts of our requirements and b) had been developed consensually by a group of people. In some cases, we pruned the ontologies to remove specific class hierarchies that were not relevant.

  Ontology reuse was not straightforward because, as mentioned above, most of the context ontologies described in the literature are not available on the Web. In addition, some of the candidate ontologies to be reused had reasoning problems. Thus, we had to take a decision between reusing a different ontology, developing a new ontology from scratch, or repairing the inconsistent ontology.
- **Ontology implementation.** In the first iteration of ontology development, because of time restrictions, we limited the implementation of the ontology network to the concepts and properties needed to link the ontologies to be reused.

- **Ontology evaluation.** Finally, we performed a verification of the first prototype of the ontology network according to the ontology requirements.

The next sections present an overview of the ontology network developed and of the evaluation performed according to the ontology requirements.

2.2 Overview of the mIO! Ontology Network

The goal of the mIO! ontology network is to represent knowledge related to context. Because the domain of context is quite broad, the mIO! ontology network consists of a core ontology that interlinks different ontology modules that describe the different subdomains needed for modelling context. Moreover, modularization allows using only the modules that are involved in a given use case, instead of using the whole ontology network.

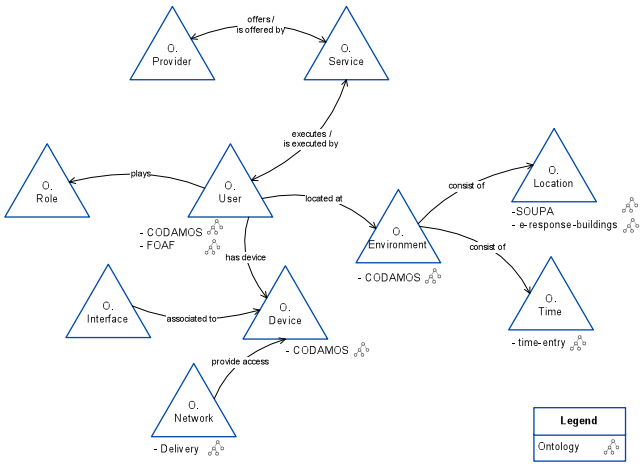

Figure 1. Overview of the mIO! ontology network.

Figure 1 presents the high-level conceptual model of the ontology network that contains ten modular ontologies: User, Role, Environment, Location, Time, Service, Provider, Device, Interface, and Network. The figure also includes those ontologies that were reused for building some ontologies as well as the connections between the ontologies by means of properties.

The first prototype of the mIO! ontology network has been implemented in the OWL ontology language [7] and contains 277 classes, 130 object properties, 116 datatype properties, and 83 instances.

Next, we present a brief description of each of the ontology modules including the ontologies that were reused in each module that satisfied our requirements:

- **User ontology**. It models knowledge about users, groups, organizations, etc. It reuses the CoDAMoS[1] and FOAF[2] ontologies.
- **Role ontology**. It models knowledge about roles, profiles, preferences, etc.
- **Environment ontology**. It models knowledge about environments including their humidity, luminosity, noise, etc. It reuses the CoDAMoS ontology.
- **Location ontology**. It models knowledge about locations such as buildings, location coordinates, spatial entities, distance, etc. It reuses the space module of the SOUPA[3] ontology.
- **Time ontology**. It models knowledge about time such as temporal units, temporal entities, instants, intervals, etc. It reuses the ontology time-entry of OWL-Time[4].
- **Service ontology**. This ontology models knowledge about services.
- **Provider ontology**. This ontology models knowledge about service providers.
- **Device ontology**. It models knowledge about devices, including hardware information, software and platform, and reuses the CoDAMoS ontology.
- **Interface ontology.** This ontology models knowledge about the user interfaces that the different devices can provide.
- **Network ontology**. It models knowledge about communication networks.

The FOAF ontology was completely reused but, in the other cases, the ontologies were pruned because they contain a lot of unnecessary concepts. This selection of concepts was carried out according to the ontology requirements.

2.3 Ontology verification

For performing a verification of the first prototype of the ontology network, we analysed up to what extent the ontology network developed covered our requirements. This verification was carried out manually, checking whether existing classes and properties cover user requirements.

Table 1 shows, for each of the subdomains covered in the ontology network, the number of requirements specified (in form of domain characteristics in Natural Language and competency questions) and the number of requirements covered by the first prototype of the ontology network.

As mentioned before, in this first iteration of the ontology development process, because of time restrictions, the focus was on extracting consensual ontology requirements from the different experts and users and on providing a first version of the ontology network by reusing existing ontologies with minimal changes.

As a conclusion we can mention that, while we were able to obtain a first prototype of the ontology network in a short time by reusing existing ontologies, this prototype only very partially covers our ontological needs, as can be seen in the table, because our ontology requirements are specific to our domain and use cases.

[1] http://www2.cs.kuleuven.be/~distrinet/projects/CoDAMoS/ontology/context.owl

[2] http://xmlns.com/foaf/spec/

[3] http://cobra.umbc.edu/ont/soupa-ont.tar.gz

[4] http://www.w3.org/TR/owl-time

Table 1. Analysis of the requirements covered by the ontology network

	Domain characteristics		Competency questions	
Subdomain	Specified	Covered	Specified	Covered
User	7	0	20	0
Role	20	5	0	0
Environment	4	2	17	4
Location	7	1	14	2
Time	2	1	8	6
Service	13	0	12	0
Provider	13	6	15	9
Device	2	1	6	2
Interface	3	0	8	0
Network	12	1	19	0
TOTAL	**83**	**17**	**119**	**23**

3. CONTEXT MANAGEMENT ARCHITECTURE

The context management architecture plays an important role in the definition of any context-aware platform. Context-awareness is related to using context to provide relevant information and/or services to the user, where relevancy depends on the user's task [8]. The context architecture presented in this paper provides the foundations for the different entities to deal with context (how to discover it, how to store it, how to access it and how to take advantage of the information it provides) in a mobile environment.

The first step to define a context-management architecture is to have a common and shared context model definition for all the entities of the system. This aspect has already been covered in the definition of the mIO! ontology network in the previous section. Unfortunately, this shared model is not enough to materialise a real context-aware application. There is usually a need for architectural services and abstractions whose primary objective is to provide suitable mechanisms for managing the context by the correspondent participants (e.g., sensors, actuators and applications).

The spread of mobile devices in this kind of architectures makes it necessary to consider some extra issues regarding this mobile world. Surprisingly, the physical limitations of these devices are not an impediment by themselves but because of the additional issues that have to be considered (e.g., distribution, scalability, modularity, mobility, privacy, fault tolerance, and even battery life and network connections).

Due to these aspects, many existent platforms choose to develop middleware for context-aware systems, such as [9] and [10] among others. A middleware infrastructure enables the definition of standard abstractions and common services that facilitates the interaction of the different context-aware entities.

Our work deals with these kinds of aspects and therefore we have pre-defined a context management architecture that gathers the different entities our context-management architecture is going to handle. This logical architecture is a first step towards the definition of a comprehensive context-management architecture. Figure 2 shows this logical architecture, where the specific deployment and internals of each component are yet to be defined.

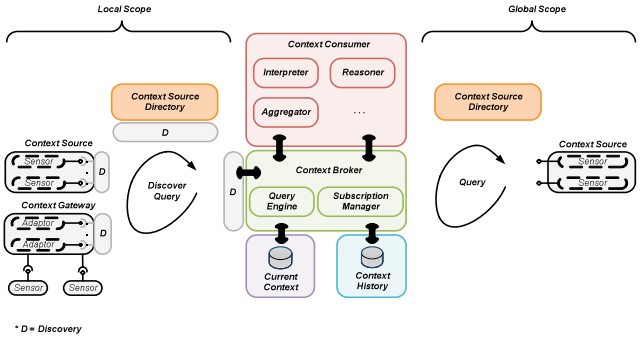

Figure 2. Overview of the mIO! Context-Management Architecture.

Next, we present a brief description of each of the components of the architecture.

3.1 Context Providers

The context provider (or source) is a computational entity able to provide certain context information following an ontological model. Context Providers consider are very diverse such as sensor data in objects located in the user's surroundings, context information provided by the mobile terminal (e.g. SW & HW capabilities, temporal data, etc), user's emotional state (gathered by ad-hoc devices), user's instantaneous privacy statements, user's social network/s, mIO! Services' functionalities, mIO! services' popularity, etc.

3.2 Context Consumers

The context consumer is a component that uses context information (instantaneous and historical) to carry out its functionality. Context consumers, for instance, are the intelligent agents that provide at every moment the most appropriate mIO! services to the user taking as input data the context of the user and the mIO! services.

Once a user decides to execute a mIO! service, such service access the context to self-personalisation and to achieve its target.

3.3 Context Manager

The context manager is a central element that will manage the context information generated by diverse context providers and will handle the requests from the context consumers. This element will have the following submodules:

- **Context Gateway.** The previous definition of context source implies that every provider is able to provide its information using a semantic definition. Bearing in mind that for limited mobile devices or platforms this is not always possible, a context gateway figure has been defined. This gateway acts as a semantic enabler for limited devices, transparently acting as a context source for the rest of the components of the architecture.
- **Context Broker.** It defines a common access interface to the current context acting as a façade for the context consumers. This element will coordinate the underlying mechanisms for retrieving the context inside the architecture. It will have two main components, a query engine and a subscription manager.

- **Context Source Directory.** Because of the characteristics of the mobile environments, not all the context information will be locally available to the context-consumer. This is the reason why a directory element has been included. For global context sources (those available on the Internet) a context source directory will host the definition and access mechanism of these sources. In local environments, a directory will make the discovery of context sources easier.

  When adding a directory element to a context-management architecture, there is usually a trade-off between the flexibility and the complexity of the system. Besides, it is unlikely that a directory element can be deployed in any real scenario. Therefore it is advisable to make this component optional in highly dynamic environments such as in the scope of our work.
- **Current Context.** This component stores the instantaneous context in a suitable semantic representation as defined earlier.
- **Context History or Life-flow.** As important as the current context information is the history or flow of that information. Often underestimated, context history has proven to be priceless information in order to analyze patterns, behaviours or trends. Therefore, we need a mechanism to define that information, to generate, maintain and access it.

3.4 Basic Interactions

Once the main elements of the context-management architecture and their functionalities have been presented, the basic interactions must be defined in order to obtain a general overview of how the system is expected to work. The operation of the mIO! context architecture can be summarized in four main aspects: discovery mechanism, context access, context processing, and context history.

3.4.1 Discovery Mechanism

Any dynamic context-aware environment needs a discovery mechanism for the different components to start interactions among them. In our proposed architecture, context-sources have to be accessible by the context-consumers through the context-broker element. This dynamic discovery will be carried out in the local environment of the context-consumers. Global discovery does not apply, apart from the access to a preconfigured directory which has the definition of globally accessible sources. Many discovery mechanisms have been proposed so far for context-aware systems (see [11] or [12]). Regardless the concrete protocol or selected implementation, the discovery protocol in our work will take into account the following aspects:

- **Context source definition language.** Context sources need to be defined using a common language to announce themselves to the rest of the components. This language must at least support primitives for defining the characteristics of the context source, the information it provides, and the endpoint from where the information can be retrieved.
- **Context source publication mechanism.** Context sources will announce themselves periodically and those announcements are going to be processed by the interested entities (e.g., the local directories or the context brokers). The context sources might be very heterogeneous and dynamic, so they just announce themselves and provide an endpoint for the communication with the rest of the entities.
- **Context source directories.** Context sources can be automatically discovered using the publication mechanism. In addition to this discovery method, the mIO! system will support the definition of context source directories. These directories can be either local or global. When a context source needs to be discovered, a context-broker can use either this announcements or query a local directory (if it exists) for an appropriate context-source. This local directory will store the locally discoverable context-sources (locally meaning in the physical surroundings of the context-broker). Global directories will be accessible through a global access network and its address will be previously known. The directory owner or administrator will register global context sources that fulfil certain characteristics or predefined policies. Local and remote directories have the same structure and philosophy; they just differ in the way they are discovered. The former is discovered dynamically and the latter's address is preconfigured.

3.4.2 Context Access

Another important interaction in a context-aware system is how the context is accessed by the different consumers. This access is going to be independent of the different underlying concrete mechanisms, that is, we have chosen to follow a data-centric approach where the consumers just ask for information, regardless the source the information came from. This kind of approach leverages the practical knowledge that a consumer needs to know in order to access the context, simplifying the communication interfaces. In contrast, we need to define a context-query language descriptive enough for the consumers to define their context needs.

This query interface is complemented by a subscription mechanism where consumers can register for changes or updates in certain context-values or expressions. This subscription mechanism will also be built over the context query language used in the query interface.

3.4.3 Context Processing

Context is useful not only for the information it directly provides but also for the information that can be deduced from it. Deducing information from context can be done in several ways, the most common of which are semantic reasoning, interpretation of context, and aggregation of context. Regardless of how context is processed, the objective is to generate new relevant information that is useful for the consumers. The context-processing in mIO! will be based on (but not be limited to) semantic reasoning, taking advantage of the possibilities this representation provides. Virtually any processor can be added to the platform because of the interfaces it will provide.

The main challenge we are facing with context-processing is that any complex reasoning process is bound to be computationally hard and seldom do mobile devices have the needed capabilities to support them. However, we will try to explore the possibilities of adapting this reasoning process to the mobile world and its feasibility within currently available mobile devices.

3.4.4 Context History Analysis

Context can be seen as something immediate that is constantly changing. Many systems focus on that immediacy for the development of context-aware applications. However, the evolution of the context is valuable information that might be

fruitfully analysed, using data mining techniques. The only problem with context history is the information explosion it leads to, so special attention has to be drawn to what the history is going to be used for and where it is going to be stored. We are exploring the use of context history to detect behavioural patterns of the users, her/his profile and the high-level situation the user is in, e.g., using machine learning techniques drove by scenarios requirements.

4. Context Execution Use Case

The following use case scenario has been designed in order to validate both the proposed context models obtained in the modelling phase as well as the architecture that shall support context-aware services. As shown in previous sections, the notion of context itself along with the contextual architecture have been designed considering the general goal of providing services to people in mobility scenarios through their mobile devices, being those services provided from another user's device (prosumer mode), or from an application server platform (hosted mode).

The proposed architectural framework can support a wide variety of applications. But, among such diversity, a single use case is presented in this paper, although many others are currently being investigated.

The tourism domain is widely considered to be one of the emerging industrial sectors where mobile services are highly demanded. In fact, in 2015 there will be more than 3 billion travellers around the globe and they will demand more ubiquitous services, specific to the situation of each individual, as well as to their personal preferences in specific circumstances. Surveys reveal that over 90% of travellers carry a mobile device with them. Time will be a very scarce resource and connectivity to all kinds of services in mobility will be highly demanded and required.

Accordingly, tourism turns out to be a very adequate application domain for the kind of services that will be developed within the ongoing research paper, given the dynamic changing situations that tourists experience, that can be followed by an appropriate context management model. In this use case scenario, service is defined as follows: service is a context-aware software entity that provides added value assistance to the user. The service has a user interface and is executed in a server hosted either at another user's mobile device or at the network.

The selected service that can potentially enhance the tourism experience is related to the selection of tourist information depending on the tourist's mobility and preferences. That means, among other activities, providing directions to locations within an unknown city and/or descriptions of the points that could be of interest for the tourist according to user's profile and role with respect to his/her context at one particular moment. Based on contextual modelling of the user, recommendations of services, commercial offers or even adaptation of the interface used to present the results to the user in the mobile device can be adapted. Such a service can be driven to support the user's mobility while an individual is in a particular city.

4.1 Use Case Description

Let us consider a particular individual that has arrived in a city for the first time and that is travelling along with his wife. That information can be obtained from the location of the mobile devices and querying the context history database. Both devices have been located in the same bearings at the same time (the system concludes through reasoning that these two individuals are located together in the same place). Moreover, based upon the information stored in the context history database, the system finds out that this is the first time this couple visits the city, as the coordinates found in the mobile devices have not been found in the database.

In addition, based on the time and date, as well as on the fact that the individual is with his wife, the contextual infrastructure assigns to these users the role of "tourists". Following, the tourist recommendation service is automatically informed about such a decision and considers the previous information to provide sightseeing alternatives. As the individuals are not familiar with the city, different possible places to visit are selected by the tourist recommender service based on the user's combined preferences (topics that the users were interested in previous similar situations, i.e., while visiting new cities in the past) taken from the context history.

Therefore, a first place to visit is presented on the mobile device, along with the public transport options available (given that the users arrived to the city by train, so it is clear that they do not have a car to use to move in the city). While on the bus, the users might not have a clear idea of the best bus stop to take. The service will keep the users informed about such topics, specific to route events while visiting the city. The users may also get information about nearby museums compatible with the user's preferences or hobbies. Specifically, the users may get special last-minute offers, based on the fact that they can be very close to the museum. For instance, a museum that might be interesting for the users is displayed on the mobile phone. In ten minutes time, a visit group is available with two free places to complete the group. Given that the museum is interested in completing the visit group, the users subscribed to the contextual recommendation service get special lat-minute discounts if they are close to the museum.

As it was previously stated, the individual has a context history, obtained from previous actions of the user. Apart from other types of information, the system has stored the types of actions or preferences of this specific user in similar situations. Eventually, the actions taken by friends or relatives are also considered, through a reputation-based validation process. Such information is stored in the context history element, presented in the section 3.

4.2 Use Case Contextual Interactions

Figure 3 presents all different entities described in the architecture section as well as the transactions among them.

The mobile device has a Context Provider agent running, as it was introduced in the section 3. Such an agent is capturing information about the user and progressing that to the Context Manager element that includes several of the entities presented in section 3. The User device will also run the User Application (that can be a simple browser to present the information to the user).

Such context information is progressed via standard transport protocols. In the case of a mobile device with PS (Packet Switched) connectivity through a GPRS access node, that transport can be implemented over IP protocol, in the case when the implementation is a web service implementation (HTTP/SOAP), or directly through standard operator network protocols, like SIP (Session Initiation Protocol [13]). The specific

API used to progress that information is still under discussion, but given the nature of the context information and the allowed latency, a protocol based on XML schema (XSD) fits perfectly with such purpose. Although only one Context Provider is shown on figure 3, several providers or sources could exist, both at the mobile device as well as on the infrastructure. Each one of them can progress specific information that will be aggregated.

The Context Manager gets the information from all context providers and aggregates that based on the modelling, reasoning and inference principles presented in previous sections. High-level context information of the situation of the user is obtained. Such context is stored at the context history log to be used later for machine learning procedures that will optimize a user model. Such area means an active research area, but such details fall out of the scope of this paper.

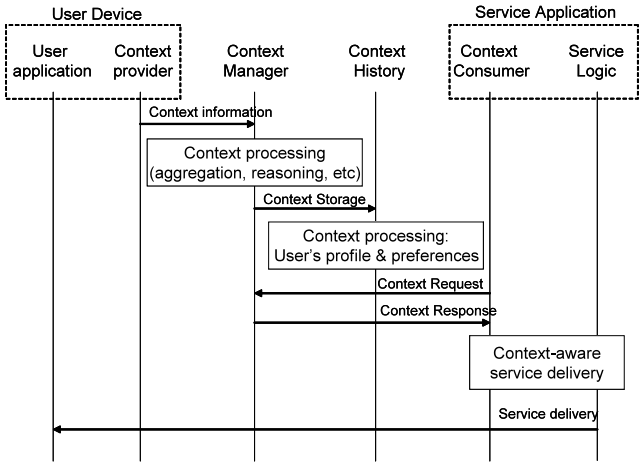

Figure 3. Interactions among contextual entities.

At a given moment, and following again network protocols and basically the same contextual API, the context consumer running at the service platform request to be informed about specific events or contextual situations for a given user. Such request, although can be transported over same options as with the Context Provider, shall be potentially much richer on the nature of the information requested. That is due to the fact that the Context Consumer requests to be informed about specific situations of users, and accordingly those shall be described in the request message generated by the Context Consumer. A semantic approach based on RDFS/OWL semantics can be followed, being fully compatible with the transport options presented in the previous paragraph.

Finally, when the context manager gets low-level context information from the context providers and gets the context of the user that matches the conditions for which the context consumer has requested to be reported, the context manager generates a notification to the consumer. That notification will include the user identifier for which such notification is generated, along with any additional data that may be considered necessary. Given that the context consumer is running at the application server itself (although other architectures of the application server modules can be studied), the service business logic is informed about the situation of the user, and the service is delivered to the user device.

This way the context provider may report the specific location of the terminal, local agenda information, etc. The Context manager can process such low level information into context information that may match the context consumer conditions ("User A is on tourist mode, is near museum M, and has at least 1 hour available ahead before lunch time to visit the museum"). The contextual recommendation service can then generate a notification of special offers, etc, directly to the user device, as part of the business logic, specific to the service.

This basic context-aware service delivery can be enhanced in different ways. For example, the contextual service is able to automatically detect tourist comments and services generated by other people with similar profile and characteristics with respect to the exhibition in the museum. As the original route of the traveller has changed due to the last minute decision to visit the museum, a new route of buses has automatically been reconfigured and new options are also shown in the display: other lines and location of bus stops as well as schedules in which the individual can take other buses to get to his original point of interest.

This same philosophy can be reused to mostly any service that may be proposed, just by implementing a Context Consumer API at the service platform to integrate that with the contextual architecture. Through such API, the service will get information relevant for the service itself that will enhance the service delivery, as shown in this use case.

5. CONCLUSIONS AND FUTURE WORK

This paper presents a complete context management framework, focused on user mobility that consists of a semantic model for representing context and a complete architecture to implement such model and enhance end-user services, by defining specific modules to do so. Finally, a use case is presented, in which both the previous model and architecture are used to implement a specific service in a mobility tourist environment.

While this is a first step in the development of the framework, our future challenge is to obtain a set of resources that can be easily reused in general application environments to manage context. Being our main focus mobile users, we require coping with a continuously evolving environment in all aspects, from end-user applications to network or device capabilities. Therefore, our work (ontologies and architecture) tries to be as extensible and maintainable as possible.

The next steps are twofold. On the one hand, the development of the mIO! ontology network will continue with a second iteration of the development process where the existing requirements will be validated and refined as well as the ontology network, which will be further developed and evaluated not only according to ontology requirements but also from a user perspective. On the other hand, the development of the mIO! context-management architecture is going to be focused on the low-level definition of each element of the architecture, extracting a common API for the main functionalities of the system. As part of such definition, the generation of demonstrator setups where the principles presented in this paper are verified will be a key step, in order to explore the possibilities of implementing that functionality for mobile environments, analysing its benefits and its drawbacks.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] Dey, A. K. 2001. Understanding and using context, Personal and Ubiquitous Computing Journal, 5(1), 5–7.

[2] Haase, P., Rudolph, S., Wang, Y., Brockmans, S., Palma, R., Euzenat, J. and d'Aquin, M. November 2006. NeOn Deliverable D1.1.1 Networked Ontology Model.

[3] Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Connix, K., Jonckers, V. and de Bosschere, K. 2004. Towards an Extensible Context Ontology for Ambient Intelligence.

[4] Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M. and von Wilamowitz-Moellendorff, M. 2005. Gumo – The General User Model Ontology. In User Modeling, 428-432.

[5] Chen, H., Finin, T. and Josh, A. 2005. The SOUPA Ontology for Pervasive Computing. In Ontologies for Agents: Theory and Experiences, 233-258.

[6] Suárez-Figueroa, M. C., Aguado de Cea, G., Buil, C., Dellschaft, K., Fernández-López, M., García, A., Gómez-Pérez, A., Herrero, G., Montiel-Ponsoda, E., Sabou, M., Villazón-Terrazas, B. and Yufei, Z. February 2008. NeOn D5.4.1: NeOn Methodology for Building Contextualized Ontology Networks. NeOn project.

[7] McGuiness, D. and van Harmelen, F. February 2004. OWL Web Ontology Language overview. W3C Recommendation.

[8] Dey, A. K. and Abowd, G. D. 2000. Towards a better understanding of context and context-awareness. Proceedings of the Workshop on the What, Who, Where, When and How of Context-Awareness, ACM Press, New York.

[9] Manuel, R., Christopher, H., Renato, C., et al. 2002. A Middleware Infrastructure for Active Spaces. IEEE Pervasive Computing, vol. 1, no. 4, pp. 74-83.

[10] Hofer, T., Schwinger, W., Pichler, M. et al. 2003. Context-Awareness on Mobile Devices - the Hydrogen Approach. In Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9 - Volume 9.

[11] Marin-Perianu, R., Hartel, P., and Scholten, H. 2005. A Classification of Service Discovery Protocols, Centre for Telematics and Information Technology, University of Twente, Enschede.

[12] Edwards, W. K. 2006. Discovery Systems in Ubiquitous Computing. IEEE Pervasive Computing, vol. 5, no. 2, pp. 70-77.

[13] IETF RFC 3261. SIP: Session Initiation Protocol.