# Ontology access patterns for pervasive computing environments

F.J. Villanueva, D.Villa, J. Barba, F. Rincón, F.Moya, J.C López
*Department of information Technologies and systems*
*University of Castilla-La Mancha*
*Paseo de la Universidad 4, Ciudad Real, Spain*
*Felix.Villanueva@uclm.es; Jesus.Barba@uclm.es; Fernando.Rincon@uclm.es;*
*David.Villa@uclm.es; Francisco.Moya@uclm.es; JuanCarlos.Lopez@uclm.es.*

**Abstract**. Lately, the use of ontologies for semantic description in pervasive systems has been researched and applied in numerous projects. The characteristic of this type of representation for data, services, interfaces, etc. allows the development of advanced services and evolving the interaction between entities in a pervasive environment. However, there is a characteristic of the pervasive systems related to the high degree of mobile devices present in them that are susceptible to access to several ontology-based pervasive systems with different representations, interfaces, etc. that, traditionally, the ontology based systems do not take into consideration when they are designed and implemented. This problem limits the capacity of systems to interact with mobile devices who do not have implemented the specific application for use that systems and interact with it. In this paper we define an access extension automatically generated for ontology based pervasive systems. Based on the general structure of ontology-based pervasive systems and the use of the applications specifically developed for interact with them we will extract several patterns in order to lookup information and allow a basic interaction with the system. In this way, we will define a set of access patterns based in commons structures and interactions in similar way to the concept of design patterns appeared in ninety decade and largely used in software engineering. Our patterns are designed with independence of the specific implementation and representation of the ontology. Finally, we will present a tool that automatically generates a specific implementation of the patterns defined and we will describe a prototype that uses this extension by mean of web services.

**Keywords:** Patterns, Pervasive Computing, Ontology, Web Services.

## 1 Introduction

Since definition of pervasive computing, the research community has been looking for methodologies and structures for semantic description information resident in pervasive computing sceneries. In fact, this has been one of the key problems in the implantation of pervasive environments.

Lately, the use of ontology it seems a good approach for this necessity of semantic description as we can see in [1], [2], [3], [4], [5], etc.

Additionally, if we have into account that a pervasive environment must provide its services and information to mobile clients that appears and disappears requiring both information and services it is seems clear that the model of use it is a key aspect in the real usability of any pervasive environment.

Since user's point of view, the installation of client applications for each one of candidate scenarios where it is possible to interact (home environment, office environment, airports, railway station, restaurants, etc.) it is, at least, disturbing.

The use of different languages for ontology specification (e.g. RDF, DAML+OIL and OWL which is lately coming dominant) and the development of interfaces for access and use of the information stored in the ontologies, present us a set of combinations where only the clients applications developed for those system can interoperate with them excluding the rest of devices and restricting the use of them for third parties applications.

In this article we propose an extension to these ontology based pervasive systems based in access patterns. We will define a set of access patterns which has been extracted of the observation and abstraction of how the information is accessed and used by users in a pervasive ontology-based environment in conjunction with a study of existent API's for ontology-based systems.

However, suggest to all systems to implement an extension is no realistic in similar way that it should be to suggest a standard API. Different research community opinions and competence between business should be a problem for achieve an agreement. To mitigate this problem, a tool which make possible generates a candidate implementation of the defined patterns has been developed.

| Project/Arquitecture | Ontology Language | Ref. | Project/Arquitecture | Ontology Language | Ref. |
|---|---|---|---|---|---|
| **Amigo** | **OWL** | **[5]** | **User Policies** | **DL** | **[7]** |
| **MoGATU** | **DAML+OIL** | **[6]** | **TAP** | **RDF** | **[9]** |
| **GAS Ontology** | **DAML+OIL** | **[13]** | **SOUPA** | **OWL** | **[4]** |

**Table 1.** Ontology based pervasive environments examples

## 2 State of the Art

We can find ontology based systems in many applications, since systems that classify general information like TAP[9] where different types of information is represented, for example music, geographic information etc. until specific systems where the target is express the knowledge in science specific topic like software patterns design [11]. Some examples of these systems are presented in Table 1.

Ontology engineering has acquired importance in last years due the increasing complexity of building general ontology-based systems. The abstraction pattern is a powerful tool in software engineering which it has been proposed in some areas of ontology engineering, for example in [10] a set of patterns is defined for the design of the ontology based systems.

If we study ontology based systems, in general we can find the system represented in the figure 1. Actually, the most considerable effort is the knowledge representation in a set of ontologies and the storage of these ontologies in a repository.

Generally this data base is called *Ontology Repository* and save the structure of the knowledge and its relation. Directly associated with this ontology repository we can find the knowledgebase or KB which, using the ontology repository, store all entities based in the ontology structure.

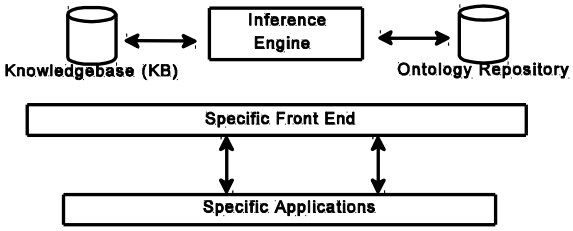


**Figure 1**.- General structure of an ontology-based pervarsive system.

Additionally, systems add to this structure an inference engine with the target of generating conclusions from KB and their relations expressed in the repository.

Finally, to use the structure described in Figure 1 *front-ends* are developed like middleware for specific applications. Such role can be assumed for works like Jena [12], XQUERY [14], RDFQ [15] increasing the variety of possible solutions at same time that aggravating the problem of the interoperability and heterogeneity.

## 3 Motivation

The main target of this project is providing a basic access to software clients resident in mobile hand-held devices to information of a pervasive system.

In this way, from the initial architecture developed in SENDA[17] project, where services are described by mean of ontologies, we propose a pervasive system for travellers assistant which it is destined to be implanted in airports, ports, railway stations, etc. with the target of assist to the persons in its mobile devices of all information required in their travels (timetables, tickets, renting, transfers, information about cities, hotels reservation, etc.) in an integrated way.

The structure of the project SENDA is out of the reach of this paper, however there were two main problems where the definition of the patterns described in this paper attends to resolve. At first place, we need to develop a minimal software component (resident in the mobile device) to interact with the system. The problem arises when this software component is used for interact with a pervasive system distinct which was developed. In the other hand, a key factor provided by ontology based systems is the possible reutilization of the knowledge and its structure stored in the ontology way. When a client wants to extract information of ontology based system it is necessary to be compatible with the system and find the system that have the information, and, in the system, the exact information that the user needs.

There are several approaches for solve the interoperability problem:

To define all information with the same structure an utilize the same API to access the information, this approach is taken in SOQA[16]. However, it is not likely that only one architecture will be utilized for all environments. Additionally, to develop a standard API for all applications is unrealistic.

To implement and to install in each mobile device all applications necessaries for each possible environment. This option again is unrealistic due to limitations in the mobile devices.

We think that an important work has to be done homogenizing, at least, a basic access to ontology-based systems and extending their use to applications resident in mobile devices by mean generic applications.

## 4 Access patterns

Propose a common API for all different types of ontology based systems is an unrealistic assumption. Nevertheless, to define extensions for specifics API's (originally developed for each system) in order to allow clients to get a basic information about the ontology based systems by mean of a basic interaction is most realistic. Additionally, if we provide a tool which makes possible get an automatically-generated service for such basic information, the costs of the implantation, and possible reservations to deploy such service, can be reduced right back.

If we study the different types of ontology representation languages we can extract a set of concepts (classes, relations, attributes, etc.) and its implementations in each language (Table 2).

| DAML | OWL | OIL |
|---|---|---|
| `<daml:Class rdf:ID="T">`<br>`<rdfs:label>T</rdfs:label>`<br>`</daml:Class>`<br>`<daml:Class rdf:ID="A">`<br>`<rdfs:subClassOf rdf:resource="#T"/>`<br>`</daml:Class>` | `<owl:Class rdf:ID="T" />`<br>`<owl:Class rdf:ID="A">`<br>`<rdfs:subClassof rdf:resource="#T" />`<br>`</owl:Class>` | `Class-def T`<br>`Class-def A`<br>`Subclass-of T` |

**Table 2**.- Ontology structure in DAML, OWL and OIL language.

We can see how *T* and *A* are like classes and its relation (*A* is a specialization of *T*) like subclass-of. If we study some of the API's developed for ontology based systems (e.g JENA, XQUERY or SOQA) we can see how most of the API's work with these general concepts. We are studding this common practices in existent API's for define a set of patterns in order to provide an starting point for develop automatically generated services which make possible to allow generic applications to interact with an ontology based system.

Traditionally, a software pattern has been defined like a probed solution for a recurrent problem. Lately, the use of patterns for resolve recurrent problems has been broadly used successfully in software engineering. So extending the concept of pattern to the problem described in this article we achieve abstracting the access to a pervasive ontology based system from implementation and platform detail.

## 5 Patterns definition

In a general way, when a client interact with an unknown ontology based system we identify three type of problems, at first time the client needs know between several ontologies, which one is the best for their proposes (e.g in a search of some type of information), after it selects the most appropriate it need get knowledge about the general structure of the system and finally, it uses the knowledge stored in the system. So, we define three sets of patterns:

*Suitable contents patterns:* In spite of with descriptions patterns resumed next a client can explore the structure of an ontology, one of the first steps in getting

information from an ontology based system is evaluating which system (between several candidates) can fit better with the client necessities. In general we need a set of metrics that allow us to choose the best candidate for explore it with *descriptions patterns*.

*Description patterns*: This set resolves the problem of getting information about the knowledge represented in an ontology-based system. So, this set allow to clients to know the concepts and its relations.

*Use patterns:* The last step after know the appropriate ontology and navigate for their structure is getting the information from KB. These patterns allow us to access the information itself.

The main patterns identified by now for the set of *suitable contents patterns* are based in the metrics defined in [8] for quality of ontology based systems. We use such metrics for getting an ordered list of systems related with the client concept target. Therefore, the set of *suitable contents patterns* are:

- *Class importance:* This pattern resolves the problem of getting an importance estimation of a concept. E.g in the implementation and following [8] we relate the importance ratio with the total number of instances associated in the KB to such concept or class.
- *Class connectivity*: This pattern resolves the problem of getting a depth-treated estimation of a class or concept in an ontology by mean of the relation with other classes or concepts. The more relations have a class, the more central role play in the ontology.
- *Attribute richness:* This pattern resolves the problem of getting a richness of the description of a class or concept by getting the number of attributes of given class.

With the three simple patterns defined before, we can get a quickly idea about the description of a concept in an ontology-based system so we can choose the most appropriate system for the client proposes. Once selected the most appropriate system, we need to ascertain the internal structure of the knowledge stored in it. Fur such proposes we defined *the descriptions patterns*:

- *Domain*: This pattern resolves the problem of getting all concepts contents in a system. Generally a list of classes described can give us an exact set of the main concepts described in such systems.
- *Family:* This pattern resolves the problem of getting all concepts related in a system. For example, allow us to know all relations (e.g “parent of” or “subclass of” “equivalentClass” “disjointWith”) for a given class.
- *Description:* this pattern resolves the problem of getting all attributes of a given class.

Finally, the *use patterns* allow us to getting the information from the KB:

- *Population*: this pattern resolves the problem of getting all instantes of a class in the KB.
- *Value-of*: For a class, instance and attribute, with this pattern is possible to get the value associated in the KB.

The set of patterns defined in this article is a minimum set in order to getting a basic interaction by mean of lightweight clients resident in mobile devices. All this patterns can be implemented in different ways by mean of specific API’s like Jena, some of them with query languages like XQUERY or RDFQ.

We are aware that with the patterns defined we cannot access to all knowledge that ontology can represent however they are powerful enough for our proposes.

## 6 Prototype

A prototype has been designed for guide the pattern extraction and design. Our target device is a PDA which has to be able to access the information stored in an ontology-based information assistant system for travelers.

We chose Web Services like technology implementation tool due its facility for are acceded by Internet. As we said in the introduction our main target is extending the number of systems which can be acceded by a generic application resident in the PDA. In spite that our actual prototype is based in our system, we are developing tools for third parties systems which can increase the information available for our generic PDA application, examples of this type of systems can be systems related with news, tourism information, restaurants, etc. like for example, the system described in [9].

Web Services also allow us to extend our generic information to other systems resident in Internet and facilitate developing for other devices (e.g cellular) our general application.

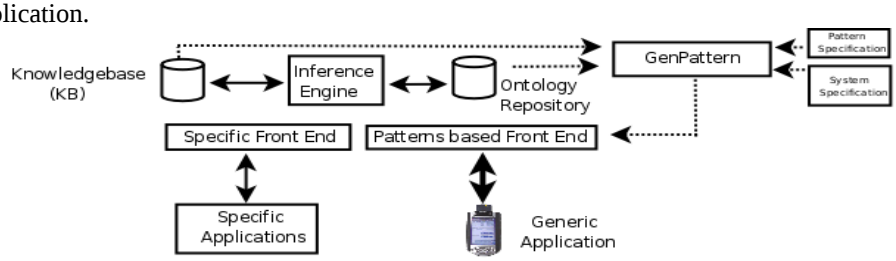


Figure 2.- GenPatterns infraestructure

Actually we are developing in python a plugin-based application called *GenPattern* which takes like input the pattern specification in MOF/XMI from OMG and a System Specification for a specific system and generates in automatically way a Web Service which represent our patterns based front end (fig. 2).

The *Pattern Specification* represents a metamodel which describes the concepts associated to the patterns and their representation in different ontology based languages. Also represents the way in which different KB can be acceded (by the moment, only mysql based KBs are supported).

The *System Specification* includes two main information, at first place the ontology-language used in system specification and second the language used for access to KB. Additionally, also has been necessary include other type of information like the operating system used in the system.

The use of patterns is independent of the final front-end generated. However the use of Web Services, firstly can be used for export the UDDI generated for service discovery of our system and secondly the WSDL description can help third parties developers to develop other applications for interact with the system.

The last part of our prototype is the application client for the web service automatically generated by the *GenPattern* tool. Resident in the mobile device, this application must be capable of interact with any system which have a web service generated over any ontology-based pervasive system.

A basic dialog of this application with a web-service is showed in figure 3. The application interact with a sub-set of patterns defined in this article in order to get a specific information.

In the actual implementation, the web service return in each invocation an XML structure which can be interpreted by the library libglade for generates a GUI for such answer. This GUI is specific for the mobile device (in our case a PDA) and simplifies the application itself building dynamically the GUI in function of the XML returned by the web service.

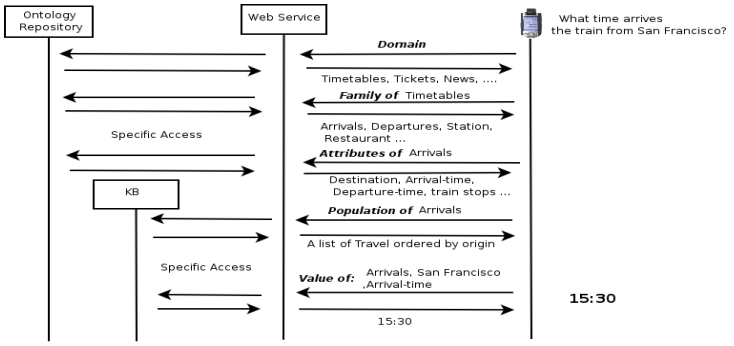


Figure 3: Basic dialog

The client application makes possible navigate by the structure of the ontology and access to the KB in a basic way. Actually the application is running in a PDA with GNU/Linux Familiar OS accessing to the web service by mean of *pywebsvcs* project (web services for python).

## 7 Conclusions

Interacting with pervasive ontology-based systems from a mobile device pass by unify the access methodology for the KB and the knowledge expressed in the ontology.

Faced with impossibility of unifying the access methodology or install an application for each system in a mobile device in this article we are analyzed the way how different system are used for develop its client applications.

From this study and the way which a user interact with a pervasive environment we are defined a set of access patterns for ontology-based system and group them into a metamodel.

Identifying well-known practices in accessing and using ontology-based systems we are identified three set patterns grouped in, *accurate contents*, *description* and *use* patterns that allow a basic interaction with any ontology-based pervasive system. An implementation example of these patterns is generated by mean of Web Services. The implementation export the knowledge resumed in the system to general applications.

The use of patterns have the same benefits than in software engineering, facilitating the develop and maintenance of access software for ontology-based systems.

Additionally a *GenPattern* tool is developed with the aim of automate the generation of the pattern-based front-end from a specification of the system (e.g language used for the ontology, the access-method for the KB, etc…). This tool expects

to extend the use of automatically generated front-ends for make possible the interaction of a general application without any modification of the existent application or front-end of the system.

Finally, we are developed an application for a PDA which interact with the web service generated by the *GenPattern* and create in a dynamically way a GUI from the XML returned by the Web Service.

Actually our efforts are centered in improve the *GenPattern* tool and extend its possibilities to enlarge the set of possible inputs of the system specification and extend the outputs to other possible interfaces like, for example, Jena. We also are working in testing our tools in third parties system in order to allow the interaction with the same PDA application developed for our project.

Lately , long terms efforts have two main lines, at first place study different inference engines in order to get patterns of use of such modules and secondly allow machine to machine interactions by mean of our front ends. This second approach could make possible to enrich a system with the information from third parties systems.

In general we think that an important work has to be done for homogenize the access to ontology-based system, our work is a first step in such way by mean of patterns. Our tools will free to specialist from technical implementation of the accessing layer to the system allowing them to centre their efforts in resuming the knowledge and information of any ontology-based pervasive system.

## References


[1] E. Jung, H. Jik, J. Woo. *Ontology-Based Context Modeling and Reasoning for U-HealthCare*. IEICE Transactions on information and Systems, Vol. E90-D, No. 8, 2007.

[2] X. Wang, J. Song, D., C. Chin, S. Hettiarachchi, D. Zhang. *Semantic Space: An Infraestructure for Smart Spaces*. Pervasive Computing. Vol. 3 No. 3, 2004.

[3] R. Masuoka, Y. Labrou, B. Parsia, E. Sirin. *Ontology-Enabled Pervasive Computing Applications*. IEEE Intelligent Systems, Vol. 18, No. 5, 2003.

[4] H. Chen, T. Finin, A. Joshi. *The SOUPA Ontology for Pervasive Computing*. Ontologies for Agents: Theory and Experiences. Springer, 2005.

[5] B. da Silva, P. Wijnen, P. Vink. *A Service-Oriented middleware for context-aware applications*. In Proc. of the 5th international workshop on middleware for pervasive and ad-hoc computing. ACM international conference proceedings series, 2007.

[6] F. Perich, A. Joshi, Y. Yesha, T. Finin. *Collaborative Joins in a Pervasive Computing Environment*. The international journal of very large databases, Vol. 14 No. 2, Springer Verlag, 2005.

[7] J. Weeds, B. Keller, D. Weir, T. Owen, I. Wakemna. *Natural Language Expression of User Policies in Pervasive Computing Environments*. In Proc. of OntoLex, 2005..

*[8]* S. Tartir, I. Budak, M. Moore, A. P. Sheth, B. Aleman-Meza. *OntoQA: Metric-Based Ontology Quality Analysis*. IEEE Conf. on Data Mining, 2005.

[9] R. Guha, R. McCool, E. Miller. *TAP: A Semantic Web Text-based*. J. of Web Semantics, 2003.

[10] A. Rector, G. Schereiber, N. F. Noy, H. Knublauch. *Ontology design patterns and problems: Practical Ontology Engineering using Protege-OWL*. 3th. International Semantic Web Conference, 2004.

[11] J. Rosengard , M. F. Ursu. *Ontological Representations of Software Patterns* . In Proc. of KES'04. LNCS. Vol. 3215. Springer-Verlag, 2004.

[12] HP Development Company. *Jena 2 Ontology API*, *http://jena.sourceforge.net/*, 2006.

[13] E. Chirstopopulou and A. Kameas. *GAS: an ontology for collaboration among ubiquitous computing devices*. I. J. of Human-Computer Studies. Vol. 62, 2005.

[14] W3C. *XQuery 1.0: An XML Query Language*. W3C *http://www.w3.org/TR/xquery/*

[15] A. Malhotra, N. Sundaresan. *RDF Query Specification.* W3C Query Languages Workshop, 1998.

[16] P. Ziegler, C. Sturm, K. R. Dittrich. *Unified Querying of Ontology Languages with the SIRUP Ontology Query API*. In 11st GI-Fachtagung für Datenbanksysteme in Business, Technologie und Web, 2005.

[17] F. Moya, J. C. López. *SENDA: An Alternative to OSGi for Large Scale Domotics Networks.* World Scientific Publishing, 2002.