# OntoPathView: A Simple View Definition Language for the Collaborative Development of Ontologies

E. JIMENEZ[1], R. BERLANGA, I. SANZ, M. J. ARAMBURU and R. DANGER
*Departmento de Lenguajes y Sistemas Informáticos*
*Universidad Jaume I de Castellón*

**Abstract.** In this paper we introduce a new view definition language, named OntoPathView, which combines the simplicity of XPath with the high expressiveness of RDF/S based view languages. This language will allow us to work in a collaborative environment for the development of complex ontologies.

**Keywords.** Semantic Web, Ontologies, View Languages, Knowledge Engineering, Collaborative Environment, Semantic Groups

## 1. Introduction

Ontologies have gained popularity in the AI community as a means for establishing explicit formal vocabulary to share between software agents [1]. For an AI system, whatever exists is whatever can be represented and can be interpreted. The term *ontology* came to refer to a wide range of formal representations from taxonomies and hierarchical terminology vocabularies to detailed logical theories describing a domain [2].

By its nature, ontology development is a collaborative process that involves several domains (or subdomains) and several domain experts [3]. In this work we introduce a preliminary architecture, based on *semantic groups* (or modules/packages) [4], for the construction of a collaborative environment dedicated to the development of ontologies. We also present a view definition language that will allow us to operate in this collaborative environment in order to handle a large ontology (a global virtual ontology), and extract ontology portions (views) from it.

This paper is organized as follows. The motivation of the work is described in the following section, where we analyze previous works in collaborative development of ontologies. We discuss their limitations, and we present the proposed architecture. Then, we introduce the syntax of OntoPathView language along with several examples in Section 3. The formal definitions of an ontology view are introduced in Section 4. In Section 5 we present the main functions of the OntoPathView language in the collaborative environment. Finally, some conclusions and future work are given in Section 6.

[1] Correspondence to: Ernesto Jimenez, Universitat Jaume I, E-12072, Castellon, Spain, ejimenez@uji.es

## 2. Motivation of the Work

Recent research on the Semantic Web and Knowledge Engineering has been mainly focused on aspects related to the following fields: languages for ontology definition (RDF/S, OWL), query and view languages (RQL, RVL), semantic integration of ontologies, and graphical ontology editors (OntoEdit [7], Protégé 2000[2], ODE[3]). Nevertheless, in the bibliography we can find only a few works related to the development of environments for the collaborative edition of ontologies. From them we emphasize the following ones: the Ontolingua Sever [6], OntoEdit and CODE [5], whose architectures are based on a centralized system with a server of ontologies. In contrast to them, Wiki@nt [8] proposes an architecture for a distributed environment where ontology modules are composed by several *wiki* pages. This last approach allows the edition of ontologies in a distributed way without a central server, but the system capabilities for the communication between developers are very limited.

Our approach assumes a distributed environment for the development of ontologies without a central server. The global ontology will be composed by several distributed and consistent portions, localized in different groups of computers. Moreover the adopted platform (grid or p2p networks) for implementing the collaborative environment will provide us with the necessary tools for notification, creation of groups, and exchange of ontology portions.

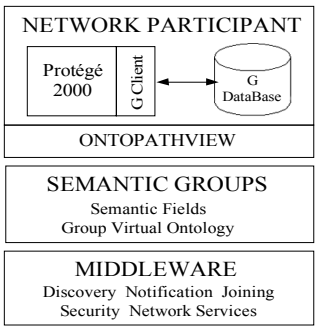


**Figure 1.** Preliminary Architecture

### *2.1. Preliminary architecture*

We have designed a preliminary architecture (see Figure 1) that has four layers; next we comment their main characteristics:

- *Middleware:* we need a platform to coordinate the network services. Let us emphasize that, at the moment, our approach is independent of the selected platform. However we will probably opt for a grid computing middleware[4] or p2p networks middleware[5].
- *Semantic Groups:* the architecture proposed is based on the work carried out by [4], which presents two approaches for the generation of semantic fields. In our approach a *Semantic Field* will be a group of interrelated views of a virtual

2 The Protégé Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu
3 ODE: Ontology-based software Development Environment: http://www.inf.ufes.br/~labes/ode/
4 Globus Toolkit: http://www.globus.org/toolkit/
5 JXTA Project: http://www.jxta.org/

ontology (a complex global ontology), which will also be an ontology. The union of the set of views, over the different *semantics groups*, will constitute the global (virtual) ontology. Figure 2 shows a virtual ontology, for the archaeological domain [14], with four hypothetical groups, which have their own application field. Notice that groups 3 and 4 are overlapping groups, since they share some concepts.

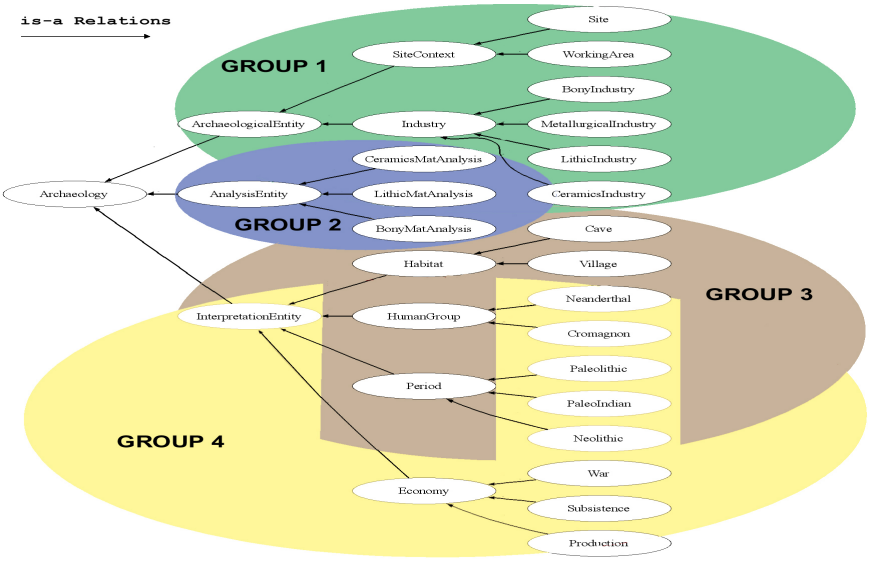


**Figure 2.** Groups in an ontology building environment

- *OntoPathView:* the main contribution of this paper is the design of a view definition language. In order to do the proposed architecture feasible we need a view definition language able to handle a large ontology (the virtual ontology), and to extract a portion from it. In following sections we will study thoroughly this language.
- *Network Participants:* At the moment, participants in the ontology development process are based on the open source ontology editor Protégé 2000 (due to the flexibility of its core) and the G platform[6]; however the final architecture should allow the connection with other ontology editors. Currently, we have extended Protégé 2000 with a plug-in[7] for the preliminary tests [9].

## 3. OntoPathView: A Simple View Definition Language

The main work presented in this paper is the design and implementation of a traversal-based view definition language, named OntoPathView. This language combines the simplicity of XPath [8] with the high expressiveness of other RDF/S based view languages [10, 11].

6 G, a Semistructured Database developed by Helide S.A: http://www.helide.com
7 The G Backend plugin for Protégé is available in the url: http://www3.uji.es/~ejimenez
8 XML Path Language (XPath) Version 1.0", W3C Recommendation, http://www.w3.org/TR/xpath

The OntoPathView language is based on OntoPath [12], a query language for ontologies. OntoPath queries are paths over the ontology graph that can contain operators over concepts, properties and instances. A query path always starts with a concept node, and follows with an alternate sequence of concepts and properties.

An OntoPath view over an ontology consists of the union of a set of queries in the OntoPath language, and a set of inference rules to guarantee the consistency and the completeness of the extracted objects, preserving their semantics. These rules will involve the reconstruction of the hierarchies of classes and properties, and the extraction of properties and concepts that are not explicitly indicated in the view but that are necessary to complete its definition.

### *3.1. Usual view expressions*

The expressions built with this language start from a concept to define the portion of the ontology that constitutes a view. Next, we explain the most usual expressions that can be included in a view specification:

- **"concept"**: The query extracts the initial concept and its direct slots of literal type (closure with reduction). This query doesn't extract instances, only concepts and properties.
- **["concept"]**: Square brackets indicate that instances of the extracted classes must be also retrieved.
- **"concept"+n**: The query also extracts subclasses (optionally, limiting the depth to *n* levels). With *'-'* instead of *'+'* we extract superclasses. And with *'*'* we extract both subclasses and superclasses.
- **"concept"/{"slot$_1$", …, "slot$_n$"}**: Extracts the initial concept with the specified slots. If some slot relates the initial concept to another class, we also extract that class and its direct slots of literal type (we apply the closure with reduction).
- **["concept"]/{"slot$_1$"="value", …}**: To filter instances from the result set, we associate a value and an operator to the slots.
- **"concept"/{"slot$_1$"+, …}**: Extracts recursively the concepts related to *concept* through the property "slot$_1$".
- **"concept"/{"slot$_1$"(n),…}**: The same as above but restricting the depth to *n* levels.
- **"concept"/{"slot$_1$", "slot$_2$"/{"slot$_{21}$", "slot$_{22}$"}, "slot$_3$"}**: In this query the *slot$_2$* expands the query graph. The range of the *slot$_2$* will be the new *initial* node.
- **"concept"/{"slot$_1$", "slot$_2$"/{?} "slot$_3$"}**: This query extracts the entire subgraph with range class of *slot$_2$* as the initial node.

### *3.2. Examples of views*

As earlier mentioned complex views can be built by stating a set of basic constructors. In this section we show the operation of the view language by means of some example views over the ontology presented in Figure 3. In this figure we have distinguished the conceptual schema from the instances by shading the latter. Let us emphasize that not all instances have been represented.

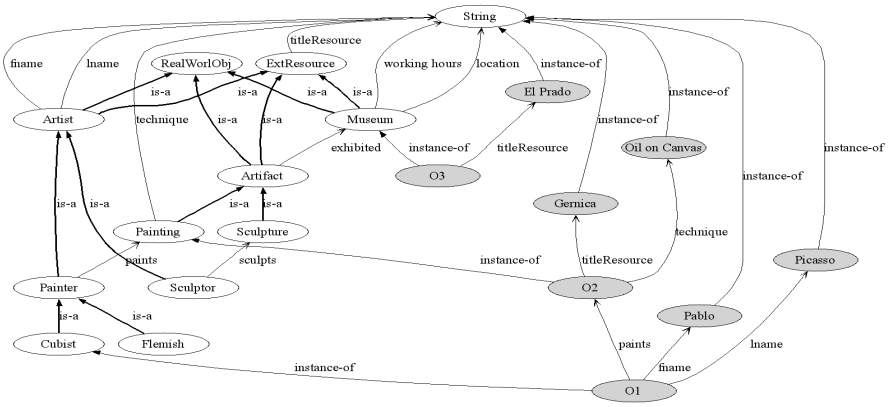


**Figure 3.** Example of an Ontology Graph

**Example 1 of an OntoPathView.** This view is specified by means of two queries:

OntoPath Query 1: ["Cubist"]/{"lname"="Picasso","fname"}
OntoPath Query 2: ["Museum"]/{"titleresource"="El Prado"}

As a result, this view definition returns the subgraph represented in Figure 4(a). Notice that, the class hierarchy has been restructured, since the concepts *Cubist*, *Artist* and *Museum* have a different group of ancestors. This ability to restructure the hierarchy of concepts in the ontology is formally defined in Definition 1 of the following section. Square brackets in both queries indicate that instances are also retrieved.

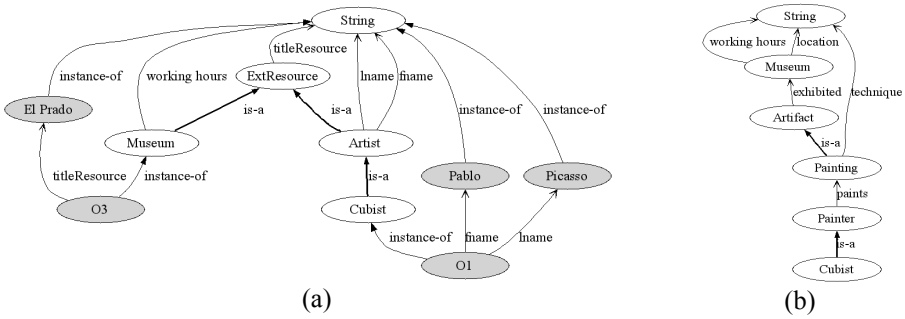


**Figure 4**. (a) Example of the View Subgraph for the Example 1. (b) Example of the View for the Example 2.

**Example 2 of an OntoPathView.** This view is formed by three queries, and does not include any instance object:

OntoPath Query 1: "Cubist"
OntoPath Query 2: "Painting"
OntoPath Query 3: "Museum"

Here in order to ensure completeness the view includes the relations *paints* and *exhibited*, and the concepts *Artifact* and *Painter*. This ability to infer relationships and concepts from the ontology is also formally defined in Definition 1 of the following section. The resulting view can be seen in Figure 4(b).

## 4. Formal Definition of an OntoPathView

In this section we present the formal conceptualization of our approach. We start from the definitions of Ontology ($\text{T-Box} = (C, \leq_C, R, \sigma, card, \leq_R, IR)$), Instance Set (also called A-Box) and Lexicon given in [13, 14]. In these definitions, *C* and *R* are the concepts and relations, respectively; $\leq_C$ is a partial order on *C*, called concept hierarchy; $\sigma$ is a function $\sigma$: $R \rightarrow C \times 2^{|C|}$, called signature; card: $R \rightarrow N^* \times N$ represents the minimal and maximal cardinality of a relation; $\leq_R$ is a partial order on *R* defined by transitive relations; finally, *IR* is the set of inference rules expressed in a logical language.

Moreover, we will denote an instance with name *o* of the concept *c* as $I|_o^c$, and the set of instances of a concept *c* as $I|^c$. Each instance is represented as a set of triples of the form (*subject*, *relation*, *object*).

Notice that the examples of the formal concepts are based on the ontology of Figure 3.

**Definition 1 (Terminological Knowledge View).** We define a Terminological Knowledge View or T-BView, over the T-Box, as a structure $\text{T-BView} = (C_V, \leq_C^V, R_V, \sigma, card, \leq_R^V)$ where $C_V \subseteq C$, $R_V \subseteq R$, $\leq_R^V \subseteq \leq_R$, and $\leq_C^V \subseteq \leq_C$.

Notice that the partial orders $\leq_C^V$ and $\leq_R^V$ are subsets of $\leq_C$ and $\leq_R$, respectively. For this reason the application of the transitive closure to these partial orders will restructure the concept and property taxonomies, and the hierarchies defined by others transitive relations. (See examples in Section 3.2).

We also define a set of inference rules that will infer the necessary association relations in the above ontology view, where $\text{dom}: R \rightarrow C$ with $\text{dom}(r) = \prod_1(\sigma(r))$ gives the domain of a relation *r*, and $\text{range}: R \rightarrow 2^{|C|}$ with $\text{range}(r) = \prod_2(\sigma(r))$ gives its range:

- if $\text{range}(r), \text{dom}(r) \in C_V \rightarrow r \in R_V$ (*e.g.*: $\text{Painter}, \text{Painting} \in C_V \rightarrow \text{Painter.paints} \in R_V$)
- if $\text{dom}(r) \in C_V, (\exists c' \in C_V, \text{range}(r) \leq_C^V c') \rightarrow r \in R_V, \text{range}(r) \in C_V$
- if $\text{range}(r) \in C_V, (\exists c' \in C_V, \text{dom}(r) \leq_C^V c') \rightarrow r \in R_V, \text{dom}(r) \in C_V$
  (e.g.: Cubist, $\text{Painting} \in C_V \rightarrow \text{Painter.paints} \in R_V$, $\text{Painter} \in C_V$)
- if $(\exists c' \in C_V, \text{range}(r) \leq_C^V c'), (\exists c'' \in C_V, \text{dom}(r) \leq_C^V c'') \rightarrow r \in R_V, \text{dom}(r) \in C_V, \text{range}(r) \in C_V$

**Definition 2 (Assertional Knowledge View).** We need auxiliary definitions to denote the Assertional Knowledge View or A-BView: the set of valid predicates for a class, and the set of instances that satisfy these predicates.

- Let $\wp_C$ be the set of valid predicates for class *c*, which is defined as follows:
  $\wp_C = \{(r, op, t, v) / r \in R_V, c = \text{dom}(r), v \in \text{dom}_C(\text{range}(r)), op \in \{=, \geq, \leq, >, <, \neq\}, t \in \{\forall, \exists\}\}$
  Where *v* is the value to match with the relation *r* in the instances, and *op* is the equality operator. Additionally, we will denote with $P_\forall^C = \{(r, op, t, v) \in \wp_C / t = \forall\}$ the set of *for all values* predicates and with $P_\exists^C = \{(r, op, t, v) \in \wp_C / t = \exists\}$ the set of *some values from* predicates.

– We define the set of instances that satisfy the predicates for the class *c* by means of the set:

$$I|^{c}_{P_{\forall},P_{\exists}} = \left\{ I|^{c}_{o} \in I|^{c} \;/\; \forall (r,op,v,t) \in P^{c}_{\exists}, \exists (o,r,o') \in I|^{c}_{o},\ o'\ op\ v \right\}$$
$$\cup\ \left\{ I|^{c}_{o} \in I|^{c} \;/\; \forall (r,op,v,t) \in P^{c}_{\forall},\ \forall (o,r,o') \in I|^{c}_{o},\ o'\ op\ v \right\}$$

For example, the next query retrieves all instances of the class *Artist* whose *fname* is "Pablo":

$$I|^{\text{Artist}}_{(\text{fname},'=',\exists,\text{Pablo})} = \left\{ I|^{\text{Painter}}_{\text{PabloPicasso}},\ I|^{\text{Artist}}_{\text{PabloNeruda}} \right\}$$

Finally, the A-BView is the union of all instances for all classes in the corresponding T-BView, that is: $\text{A-BView} = \bigcup_{\forall c_i \in C_V} I|^{c_i}_{P_{\forall}P_{\exists}}$

We also define the closure of the A-BView, by means of the following rule. In this way, any instance referenced by an instance of the view must be also included in the view.

$$\text{if } (\exists I|^{c}_{o} \in \text{A-BView}, (o,r,o') \in I|^{c}_{o}) \rightarrow I|^{c'}_{o'} \in \text{A-BView}$$

**Definition 3 (OntoPathView).** An OntoPathView is a triple (T-BView, A-BView, Lexicon). Notice that an OntoPathView is also a formal ontology.

## 5. Use of Views in the Proposed Collaborative Environment

In previous sections we have seen the proposed view language by means of several examples over an ontology, as well as its formal definition. However, our main motivation is its use in a collaborative environment. Next we present the main functions of the OntoPathView language in the proposed collaborative environment:

- A view will describe the ontology portion of a network participant (node), that is, each node will describe its resources (portions of a virtual ontology) by means of OntoPath views.
- An OntoPathView will also allow us to search previously defined views. When a new node wants to join an existing group, this node will define, by means of a view, the portion of knowledge of interest.
- Detection of conflicting views. OntoPathView will allow us to detect overlapped views (between groups or between nodes in the same group) in a simple way, intersecting the set of concepts involved in the views.
- The union of the set of views over the different groups (*semantics groups*) will constitute the global (virtual) ontology. Notice that each group can be composed by one or more nodes with its corresponding views; and the set of the views inside a group will also constitute a virtual subontology. Let us emphasize that the alignment, in the union of views, won't be a problem in the proposed environment, since the nodes and the groups will be fully coordinated.

With regard to the consistency of the virtual ontology (or the virtual subontologies), the collaborative environment must control the individual consistency of views in the different groups. The global ontology will grow up dynamically in several groups, and the system should not allow inconsistent changes in any node.

## 6. Conclusions and Future Work

In this paper we have presented a new view definition language named OntoPathView, which is an extension of the query language presented in [12]. The main advantages of this language are its high expressiveness and its simplicity of use. Other novelties of this work have been the implementation of a set of inference rules for the definition of views, the design of a preliminary architecture for a collaborative environment, and the integration of the G Database System with Protégé 2000.

In a close future we will study further aspects related to the management of conflicts in the access, the notification of changes on the ontology to the rest of users working on it, generation of semantic groups, etc. Moreover we will implement the designed architecture with a specific middleware, based on peer-to-peer or grid technologies.

## Acknowledgements

This work has been partially funded by the CICYT Project, TIC2002-04586-C04-03

## References

[1] N. F. Noy, “Semantic integration: a survey of ontology-based approaches”, ACM SIGMOD Record, v.33 n.4, December 2004

[2] N. F. Noy, M. Klein, “Ontology evolution: Not the same as schema evolution” Knowledge and Information Systems, 2003.

[3] J. Bao, V.G. Honavar, “Ontology Language Extensions to Support Localized Semantics, Modular Reasoning, and Collaborative Ontology Design and Ontology Reuse”. Technical Report, Computer Sicence, Iowa State University (2004).

[4] I. Navas, I. Sanz, J. F. Aldana, R. Berlanga, “Automatic Generation of Semantic Fields for Resource Discovery in the Semantic Web”, accepted in the 16th International Conference on Database and Expert Systems Applications, DEXA 2005, Copenhagen, Denmark

[5] P. Hayes, R. Saavedra, T. Reichherzer, “A Collaborative Development Environment for Ontologies (CODE)”, Semantic Integration Workshop (ISWC), October 2003, Sanibel Island, Florida

[6] A. Farquhar, R. Fickas, J. Rice. “The Ontolingua Server: A tool for collaborative ontology construction”, Knowledge Acquisition for Knowledge Based System Workshop (KAW’95), Canada.

[7] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, “OntoEdit: Collaborative Ontology Development for the Semantic Web”. International Semantic Web Conference (ISWC02). Sardinia. Italy. June, 2002.

[8] J. Bao, V.G. Honavar, “Collaborative Ontology Building with Wiki@nt” Third International Workshop on Evaluation of Ontology Building Tools, Hiroshima 2004

[9] E. Jimenez, R. Berlanga, I. Sanz , “Integration of the Database G with the Ontology Editor Protégé 2000 (v.2.1.2)”, Technical Report DLSI-01/05/2005, Department of Languages and Computer Systems, Jaume I University (2005)

[10] R. Volz, D. Oberle, R. Studer, “Implementing Views for Light-weight Web Ontologies”, IEEE Database Engineering and Application Symposium 2003

[11] A. Magkanaraki, V Tannen, V. Christophides, D. Plexousakis, “Viewing the Semantic Web through RVL Lenses”, 2nd International Semantic Web Conference (ISWC2003)

[12] R. Berlanga, A. Scheppler, M. J. Aramburu Cabo, I. Sanz, R. Danger, “OntoPath: A Query Language for Ontologies”, IX Conference of Software Engineering and Data Bases, Malaga 2004

[13] R. Danger, R. Berlanga, J. Ruíz-Shulcloper, “CRISOL: An approach for automatically populating a Semantic Web from Unstructured Text Collections”. Database and Expert Systems. Ed. Springer-Verlag, 2004.

[14] R. Danger, I. Sanz, R. Berlanga, J. Ruíz-Shulcloper, "A proposal for the automatic generation of instances from unstructured text" CIARP 2004, Lecture Notes in Computer Science 3287, pp.462-469, Ed. Springer-Verlag