A View-based Methodology for Collaborative Ontology Engineering: an Approach for Complex Applications (VIMethCOE)

E. Jiménez-Ruiz, R. Berlanga
Department of Computer Systems and Languages
Jaume I University of Castellon
{ejimenez, berlanga}@uji.es

Abstract

The development of very large ontologies may involve several domain experts, ontology engineers and final users, all with very different objectives. New scenarios like biomedicine, genomics and biology require methodologies that consider new dimensions in the development process, namely: dynamism, distribution and control. Existing works lack some important aspects with respect to these dimensions. In this paper we propose a view-based methodology for the collaborative, distributed and modular development of large ontologies, which achieves the identified requirements for the proper evolution and use of very large ontologies.

1. Introduction

Ontologies have gained importance in information processing, and consequently, nowadays an ever-increasing collection of ontologies describing real-world domains is available in the Web. We focus on the biomedical domain, where several large ontologies have been developed. For example: the NCI[1] ontology, the GENE[2] ontology, the FMA[3] ontology, and the set of Open Biomedical Ontologies (OBO)[4].

The ontologies above comprise thousands of concepts and relations, and they evolve in a dynamic way. Therefore, we consider necessary the design of a framework that allows a controlled evolution of these large ontologies, but without losing dynamism in their development and, of course, providing a high level of collaboration. In this work we propose a View-based Methodology for Collaborative Ontology Engineering, named VIMethCOE, that fulfils these requirements.

The remainder sections are organized as follows. Next section establishes the main characteristics of our methodology, and compares it to related work. The development phases are described in Section 3. Section 4 defines different knowledge spaces for our methodology. The Section 5 presents the view definition language used in the methodology. The view mechanism is analysed in Section 6. The application scenario proposed for testing VIMethCOE is presented in Section 7. Finally, some conclusions and future work are in Section 8.

2. Proposed Methodology

The work in [15] establishes the need for new methodologies that consider new dimensions in the development process, namely: dynamism (highly dynamic or static), team structure (centralized or distributed), and level of control (full or not controlled).

In the next paragraphs we present the main methodological aspects we consider relevant for the collaborative construction of ontologies, and how the proposed methodology faces a highly dynamic, distributed and partially controlled scenario.

2.1. Methodology Requirements

In the literature a good number of methodologies to carry out the collaborative construction of knowledge bases have been proposed. However neither of them enables, on the whole, the following aspects that we consider essential to achieve a good balance between the above dimensions.

Modularization (MO). Ontologies can involve a large amount of knowledge with several thousands of concepts. The definition of modules would facilitate the maintenance and validation of the ontology, the collaboration, the local reasoning, the visualization in ontology editors, and the reuse of knowledge.

[1] NCI: http://www.mindswap.org/2003/CancerOntology/
[2] GENE: http://www.geneontology.org
[3] FMA: http://sig.biostr.washington.edu/projects/fm/
[4] OBO: http://www.bioontology.org

Local Adaptation (LA). Each participant will be able to deal with knowledge in a local and private working space, making changes over it apart from the community's agreed knowledge.

Knowledge Abstraction (KA). The development of ontologies may involve a large group of experts from several areas. These experts may only have a partial knowledge of the domain, so they will be able to contribute in the development of only a portion of the ontology.

Personal Views (PV). Our methodology defines a view mechanism to hide and abstract the knowledge base for a specific group or application. Thus, views can be defined over one or more ontology modules, selecting the knowledge of interest. This mechanism provides participants with some freedom in the definition of their personal knowledge; at the same time it allows the system to control changes and divergences over the global knowledge.

Argumentation and Consensus (AC). Changes over the local knowledge will be published when participants consider them appropriate, and next, these changes should be evaluated by the community. To achieve a consensus, different developers must follow a formal or a semi-formal argumentation model like Ibis [9], or an extension of it. By means of this model, developers can argue the proposed changes and, optionally, they can propose alternatives.

2.2. Related Work

Among the different studied methodologies we can distinguish two groups. The first is composed by the following works: Cyc [1], Kactus [2], Uschold-King's method [3], METHONTOLOGY [4], On-To-Knowledge (OTK) [5], y UPON [6]. The works Co4 [7], DILIGENT [8], HCOME [10], Divergence Occurrences Methodology [12], (KA)^2 [13], and the OntoEdit system [14] would form the second group.

The first group represents a set of classic methodologies that propose a centralized approach to the ontology development. Although these methodologies describe several phases for the development of an ontology, they neglect collaboration issues. Works in the second group do not propose a complete methodology, but solutions to carry out an agreed definition of the knowledge.

Other ontology development systems, like Ontolingua Server [16], WebOnto [17], WebODE [18], Wiki@nt [19] and (KA)^2, rely completely on the WWW. Although they are not methodologies strictly speaking, they provide good frameworks for collaboration.

Table 1 summarizes the fulfilment of the proposed characteristics by the main systems and methodologies of the literature.

Table 1. Characteristics of Related Work

	MO	LA	KA	PV	AC
CO$_4$	√	√	X	X	√
DILIGENT	X	√	X	X	√
HCOME	-	√	X	X	√
Div. Occurr.	X	√	X	X	√
OntoEdit	X	√	X	X	X
Ontolingua	X	X	√	X	X
(KA)^2	√	X	√	X	X
WebOnto	X	√	X	X	√
WebODE	X	√	√	X	X
Wiki@nt	√	X	√	√	X
VIMethCOE	√	√	√	√	√

3. Development Phases

VIMethCOE is complementary to the existing centralized methodologies, and it is mainly focused on the collaborative development and the application tasks. Next, we describe the proposed phases in the ontology development process, pointing out that the last four are overlapped phases.

Requirements Phase. In this phase a reduced group of developers establish the initial requirements. They will define a top level knowledge of the domain either from scratch or by reusing existing ontologies. The initial modules will be defined from this top level knowledge by applying clustering algorithms like [20] or [21]. The creation of modules will allow us the definition of user groups that will work over the same knowledge fields.

Development Phase. In this phase the entire group of participants will take part in the development, namely: knowledge engineers, ontology engineers, domain experts and final users. They will work in a local and private environment, over a portion of the knowledge modules, in which they are experts, by defining *development views* over one or more modules, and over others views. In this paper we propose a view-based mechanism for this phase, which is described in Section 6.

Publication and Argumentation Phase. Local adaptations of the knowledge can be published, by means of views, in order to extend the community's knowledge. This published knowledge must be argued by others developers, and whenever a consensus is reached, the global ontology must be updated.

Evaluation and Maintenance Phase. In this phase, the initial modules may be redefined (division or union) if the growth of the ontology requires it.

Application Phase. In this phase we can define personal views over modules or other views, and extend them in a local and private environment. These views are oriented towards specific applications, and they can present divergences with other views and with the global agreed knowledge. Thus, these views represent a complementary knowledge with respect to the development views.

4. Knowledge Spaces

In VIMethCOE we distinguish several overlapped knowledge spaces (see Figure1), namely:

Private Space. This is the working space of developers. It is composed by the set of views in development and not published, and by any other knowledge that developers do not want to publish.

Public Space. It represents the shared knowledge, which can be used by the community.

Agreed Space. Inside the public space we can differentiate the knowledge that is in consensus. This space is composed by the ontology modules, as well as by the agreed views over them.

Development Views. This space is composed by the set of personal views that aims of extending the ontology.

Application Views. This space is composed by the set of personal views for a specific application. As earlier explained, this kind of views may have divergences with respect to public knowledge. In Figure 1 we can see that application views can also be used as development views.

Old Versions. Inside the public space old versions of the ontology are maintained in order to analyse the evolution of knowledge.

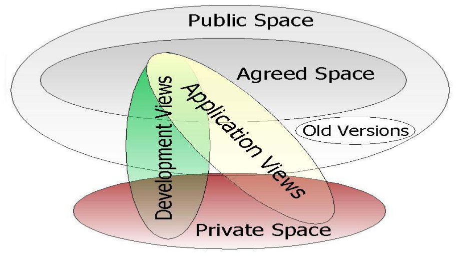

Figure 1. Knowledge Spaces in VIMethCOE.

5. The View Definition Language

One of the main characteristics of the proposed methodology is the ability to operate through views. We have designed and implemented a traversal-based view definition language, named OntoPathView [22]. In this language, views over an ontology consist of the union of a set of queries and a set of inference rules that guarantee the consistency and completeness of the view, preserving their semantics. Query definitions are paths over the ontology graph that can contain operators over concepts, properties and instances. On the other hand, inference rules will involve the reconstruction of the hierarchies of classes and properties, and the extraction of properties, concepts and instances that are not explicitly indicated in the view but are necessary to complete its definition.

OntoPathView is based on the methodology proposed in [23] for the definition of external schemata over object-oriented databases, by means of closure reduction. According to our point of view, the definition of views over ontology modules is similar to the definition of external schemata over several data sources. The work in [23] establishes a group of characteristics that an external schema, and a view in our case, must fulfil, namely:

Closed. An external schema is closed when it does not contain external references. That is, if a class A that belongs to the schema references a class B that does not belong to the schema, then class B, by closure, must be included into the schema. The problem appears when B also has references to other classes. The *closure with reduction* intends to mitigate this problem, so that the class B will be included into the schema, as a derived class, without external references, only with atomic attributes. Let emphasize that closure is also applied to instances.

Consistent. The relation between classes must be consistent. The external schema must not include axioms that can cause inconsistencies.

Completeness. All the relations between two classes A and B, which belong to the schema, must also be included. These relations can be: direct or indirect inheritance (reconstruction of class hierarchy), aggregation (reconstruction of transitive-relations hierarchy), or association.

This kind of views is mainly oriented to frame-based ontologies, and a deeper study is necessary to define a similar mechanism for other metamodels, such as description logics.

6. The View Mechanism

View mechanisms allow a collaborative evolution of the ontology with dynamism and distribution, that is, several developers extending the ontology in their local environments; but at the same time they enable the control of changes over the global knowledge. Next we comment the main characteristics of the proposed view mechanism for VIMethCOE.

6.1. View Hierarchy

Users can define views over the modules or over other views (agreed or not). These views are grouped inside a hierarchy depending on their definition and the changes made. Figure 2 shows an example of the view definition hierarchy, where agreed knowledge parts are shaded in the figure.

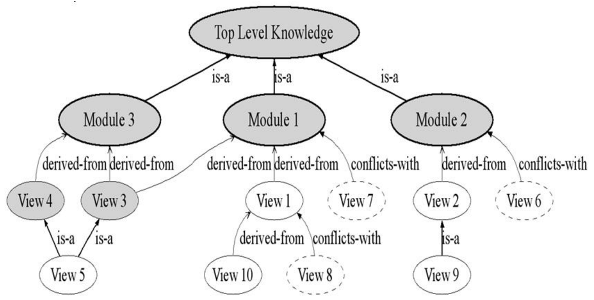

Figure 2. View Hierarchy

In Figure 2 we can appreciate three kinds of views depending on the nature of the transformations in the view definition:

Is-a views: these kind of views partially inherit the source knowledge, without doing any transformation on it. For example: selecting an entire hierarchy and including all external references, the union of disjoint views, filtering instances, etc.

Derived views carry out some simple and reversible transformations. These views are similar to those defined in [24] for defining external ontologies. For example: hiding classes and reconstructing the hierarchy, hiding properties with the closure reduction, renaming, intersecting views, etc.

Conflict views: these views imply transformations that, although reversible, may produce inconsistencies with respect to the community knowledge. For example, a user can define a view for analysis purposes where concepts and their instances are transformed into dimensions and measures.

6.2. Changes in Development Views

The evolution of modules is carried out by means of the definition of views, which are extended in a local and private way by users. These extensions may cause a loss of information and, therefore a conflict with the public knowledge. When users publish their extended views, their situations in the view hierarchy (Figure 2) are inferred depending on the source knowledge and the nature of changes over them. If an extended view only increases the knowledge or causes simple and reversible changes, the view is published like an *is-a view* or like a *derived view*. In other case, if the view causes some kind of loss of information or inconsistencies with respect the public knowledge, then this view is published like a *conflict view*.

The approach followed in VIMethCOE for the detection of change effects, is based on the work in [26], which proposes a set of operations over an ontology and analyses their implications over the conceptual schema, the instances and the consistency of the ontology.

6.3. Application Views in the Methodology

Application views come up due to the need to adapt the knowledge for a specific purpose, independently of the community's knowledge. These views may also work like development views, but in general they will have an objective and are evolution independent of the global ontology. That is, modifications in modules will not involve application views, and vice versa.

This kind of views will coexist with other development or application views and they probably have divergences with the public knowledge.

6.4. Argumentation Process

In order to achieve a consensus in the shared knowledge, a very important aspect of the methodology is the argumentation process.

When development views are published, other users involved in the development of the modified or extended knowledge can argue about them and propose alternatives. After the argumentation of a view there are two alternatives: achieve a consensus or not. In the first case, the new knowledge of the published view, or an alternative view, will be considered as an agreed view. Moreover the knowledge of the modules and the knowledge of other implied views will be updated. In the second case a conflict arises. The divergent alternatives can coexist and be extended independently, following the method proposed in [12], waiting for a future argumentation. Thus, the view hierarchy is essential to maintain this situation.

Let us emphasize that application views can also be published and argued by other users, but this will not affect to the ontology modules.

6.5. Modules and Semantic Groups

The ontology modules define an initial partition over the ontology in order to define a first set of user groups. These modules are static, but during the evolution process of the ontology may be necessary to make changes on them by means of divisions or unions.

Regarding the development views, the maintenance of a view hierarchy allows us to partially control the overlapped knowledge between the different views and their relations. However, the proposed definition mechanism is very flexible and it allows for the view definition over modules and/or other views. Thus, there may be several views with overlapped knowledge, but without a relation between them (is-a, derived-from or conflict-with). In order to control the set of overlapped views, we present the concept of *semantic group*, already introduced in [22]. The semantic group mechanism defines an incremental clustering algorithm, based on [11], and it allows us to group a set of views that share some knowledge. With these groups, communication, argumentation and actualization tasks will only imply the views belonging to a group. Let us emphasize that these groups are very dynamic, unlike ontology modules, and they may change, in an incremental way, when new views are defined and published.

7. Application Scenario

The biomedicine domain is an excellent scenario for applying this methodology, since in this domain there exist several large and very dynamic ontologies.

The specific application scenario proposed for testing the methodology is the Health-e-Child[5] project, which proposes the development of an integrated healthcare platform for European Pediatrics: decision support systems, knowledge discovery, etc.

In order to achieve a comprehensive view of a child's health the vertical integration of biomedical data, information, and knowledge is necessary. The identified biomedical information sources cover six groups, named vertical levels: molecular, cellular, tissue, organ, individual and population. Ontologies will define the formal conceptualisation of this domain knowledge, and will form the basis for the medical knowledge management system. In Figure 3 we present some vertical levels, as modules, and some possible views over them.

We consider that our view mechanism can be helpful for the objectives of the Health-e-Child project, specifically for the evolution, integration and application tasks over the set of ontologies of the knowledge management system.

Evolution. The definition of development views will allow users to select some specific portion of knowledge. They will be able to merge, when necessary, different levels of knowledge, for example: Molecular with Population (see Figure 3).

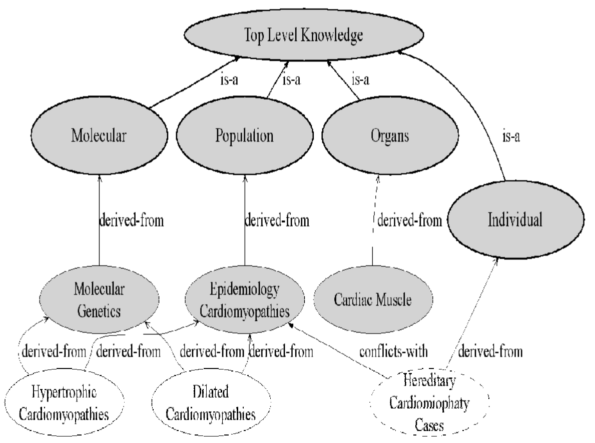

Figure 3. Example of Knowledge in Health-e-Child

Integration. The ontology modules, or vertical levels, may not have a direct relation between them. That is, there is not a concept-to-concept mapping due to differences in the abstraction of the knowledge. In order to integrate the different ontology levels it will be necessary to make several transformations, by means of views over the knowledge, similarly to [25].

Application. The access in Health-e-Child to the information sources of the hospitals will be carried out by means of the ontologies. For some specific applications it will be useful to select the required portion of knowledge since the global ontology may involve thousands of concepts. Moreover, the views will allow us to integrate the different levels of knowledge.

8. Conclusions and Future Work

In this work we have proposed an innovative methodology for the definition of both development views and application views. At present, some tests have been realized with a simple prototype where views, in OntoPathView language [22], are defined over small ontologies by means of a plugin[6] that connects the database G with the ontology editor Protégé.

In the short term we expect to create a more complex prototype that allows us to integrate several ontologies and to define views over them. In a longer term, we expect, in order to fulfil our objectives in the Health-e-Child project[7], to design and implement a system for the development, integration and application of ontologies, that collects all the requirements of the proposed methodology.

[5] Health-e-Child: http://www.health-e-child.org/

[6] G-Protégé Plugin: http://www3.uji.es/~ejimenez/GProtegePlugin/

[7] Project Objectives: http://www3.uji.es/~ejimenez/Health_e_Child/

Let us emphasize that VIMethCOE is independent of the underlying platform (grid, peer-to-peer, etc.), the ontology editor, and the ontology representation. Nevertheless, OntoPathView is based on the frame-based version of Protégé.

9. References

[1] D.B. Lenat, R.V. Guha, Building large knowledge-based systems. (Addison-Wesley Publising Company, Inc. 1990).

[2] G. Schreiber, B. Wielinga, W. Jansweijer. "The KACTUS view on the 'O' word". Proceedings of NAIC'95, Rotterdam, The Netherlands, pp. 159–168, 1995.

[3] M. Uschold, M. King. "Towards a Methodology for Building Ontologies". IJCAI-95, Montreal, Canada..

[4] M. Fernandez, A. Gomez-Perez, et al. "METHONTOLOGY: From ontological Art towards ontological engineering". Proc. AAAI-97, Stanford, USA.

[5] Y. Sure, S. Staab, R. Studer. "On-To-Knowledge Methodology", In Handbook on Ontologies". Edited by S. Staab and R. Studer (eds.). Springer (2003)

[6] A. De Nicola, M. Missikoff, R. Navigli. A proposal for a Unified Process for ONtology building: UPON, Proc. of DEXA 2005, Copenhagen, Denmark, August 22-26th, 2005.

[7] Euzenat, J. 1995. "Building consensual knowledge bases: context and architecture". In Mars, N. (Ed.). Building and sharing large knowledge bases. IOS Press. Amsterdam

[8] C. Tempich, H. S. Pinto, Y. Sure, S. Staab, "An Argumentation Ontology for DIstributed, Loosely-controlled and evolvInG Engineering processes of oNTologies (DILIGENT)", ESWC05, pp.241-256. Crete, Greece, 2005.

[9] Kunz W. and Rittel H. W. J.: Issues as elements of information systems. Technical Report S-78-2, Institut fur Gundlagen Der Planung I.A, Universitat Stuttgart (1970)

[10] K. Kotis, G. A. Vouros, J. P. Alonso. "HCOME: tool-supported methodology for collaboratively devising living ontologies", (SWDB'04, VLDB), 29-30 August 2004

[11] I. Sanz, J. M. Pérez, R. Berlanga and M. J. Aramburu: "XML Schemata Inference and Evolution". DEXA 2003. Prague, September 2003. LNCS 2736. ISBN 3-540-40806-1

[12] A. Díaz, G. Baldo, "CO-Protégé: A Groupware Tool for Supporting Collaborative Ontology Design with Divergence", 8th Protégé Conference 2005, Madrid, Spain

[13] V. R. Benjamins, D. Fensel, S. Decker, A. Gómez-Pérez. "(KA)2: Building Ontologies for the Internet: a Mid Term Report". IJHCS, 51:687-712. 1999.

[14] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke, "OntoEdit: Collaborative Ontology Development for the Semantic Web". ISWC02. Sardinia. Italy. June, 2002.

[15] H. S. Pinto, J. P. Martins. "Ontologies: How can They be Built?" In Knowledge and Information Systems, Springer-Verlag, Volume 6, Number 4, July 2004, pp. 441 – 464

[16] A. Farquhar, R. Fikes, J. Rice. "The Ontolingua Server: A Tool for Collaborative Ontology Construction". KAW'96.

[17] J. Domingue. "Tadzebao and Webonto: Discussing, Browsing and Editing Ontologies on the Web". KAW'98.

[18] J. C. Arpírez, O. Corcho, M. Fernández-López, A. Gómez-Pérez. "WebODE: A Scalable Workbench for Ontological Engineering". KCAP'01. ACM Press, 2001

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

[20] H. Stuckenschmidt, M. Klein. "Structure-based partitioning of large concept hierarchies", Proc. of ISWC, Lecture Notes in Computer Science, 2004.

[21] B. Cuenca-Grau, B. Parsia, E. Sirin, A. Kalyanpur. "Modularizing OWL Ontologies". KCAP-2005 Workshop on Ontology Management. Banff, Canada, October 2005

[22] E. Jiménez, R. Berlanga, I. Sanz, M. J. Aramburu, R. Danger: "OntoPathView: A Simple View Definition Language for the Collaborative Development of Ontologies". In B. López et al. (Eds.): Artificial Intelligence Research and Development, IOS Press, 2005. ISBN 1-58603-560-6

[23] M. Torres, J. Samos. "Closed External Schemas in Object-Oriented Databases". In Proc. of DEXA 2001.

[24] R. Volz, D. Oberle, R. Studer, S. Staab. "Views for light-weight web ontologies". ACM SAC, Melbourne, 2003.

[25] P. Bouquet, F. Giunchiglia, F. Van Harmelen, L. Serafini, H. Stuckenschmidt. "Contextualizing ontologies", Journal of Web Semantics, 2004

[26] N. F.Noy, M. Klein. "Ontology evolution: Not the same as schema evolution." Knowledge and Information Systems, 5, 2003.

Acknowledgements

This work has been partially funded by the UJI-Fundacio Bancaixa project P1B2004-30