# Using Semantics to Bridge the Information and Knowledge Sharing Gaps in Virtual Engineering

Javier Vaquero[1], Carlos Toro[1], Carlos Palenzuela[2], Eneko Azpeitia[3]

[1] Vicomtech Research Centre, Mikeletegi Pasalekua 57, 20009 San Sebastian, Spain
{jvaquero, ctoro}@vicomtech.org
[2] Inge-Innova, Albert Einstein 44, 01510 Miñano, Spain
[3] Inertek 3D, Laga Bidea 804, 48160 Derio, Spain

**Abstract.** In a Product Life Cycle (PLC) scenario, different Virtual Engineering Applications (VEA) are used in order to design, calculate and, in general, to provide an application scenario for computation engineering. The diverse VEA are not necessarily available when information sharing is needed, a fact that represents a semantic loss as the knowledge gained by using one VEA can be lost if a data translation occurs (e.g. a Finite Element program is normally able to export only the geometry). In this paper we present an architecture and a system implementation based on semantic technologies which allows a seamless information and knowledge sharing between VEA in a PLC scenario. Our approach is validated through a Plant Layout Design application which is able to collect knowledge provided by different VEA available. This work presents our system leaving a statistical analysis for future work, as at the moment our system is being tested.

**Keywords:** Virtual Engineering, Ontologies, Product Life Cycle.

## 1 Introduction

Virtual Engineering (VE) is defined as the integration of geometric models and their related engineering tools such as analysis, simulation, optimization and decision making, within a computerized environment that facilitates multidisciplinary and collaborative product development [1]. Virtual Engineering Applications (VEA) are the software implementations of VE. Each VEA in turn contains a set of Virtual Engineering Tools (VET), which is the collection of features that the VEA offers, e.g. in a CAD-like VEA tools like: *makeLine(), drawCircle()*, should exist.

Nowadays, VEA barely exploit the capabilities of contextual facts, user requirements, user experience and in general, of factors that could be easily modelled and advantaged from a semantics point of view.

From a Product Life Cycle perspective, the use of different VEA in each one of the stages is a common practice, for example CAD in the Design stage or FEA in Analysis stage. Some VEA are even proved valuable in several stages. However in this scenario, a semantic loss is reported [2] as in the outmost cases the geometry is the only feature that prevails. Information and knowledge sharing between VEA has

become an important gap for several reasons which include commercial interests of the manufacturer, the nature of legacy products and language incompatibilities. To approach a solution to this problem, we propose an architecture and a system implementation where all the knowledge generated by VEA in a PLC through its underlying VET is stored in a central knowledge repository. This repository is accessible to the aforementioned Virtual Engineering Applications in a persistent manner, providing a semantic support for the PLC and hence dropping redundancy whilst reducing important costs associated with typical format healing problems and information loss.

This paper is organized as follows: In section 2, we present briefly the related concepts necessary for our approach. In section 3, we present our proposed architecture for the semantic approximation to the information loss problem in PLC. In section 4, we present a system based in the proposed architecture for a plant design layout task. Finally, in section 5 we present our conclusions and future work.

## 2 Related Concepts

In this section we present a short overview of some concepts relevant to this paper. An interested reader is invited to review [3], [4], [5] for a wider explanation of the concepts presented.

### 2.1 Virtual Engineering Applications (VEA)

As stated in the introduction, VEA as implementations of the VE concept, are arguably the main facilitators of a new product development. In Fig. 1, the components of a VEA are presented following the sub-division introduced in [3].

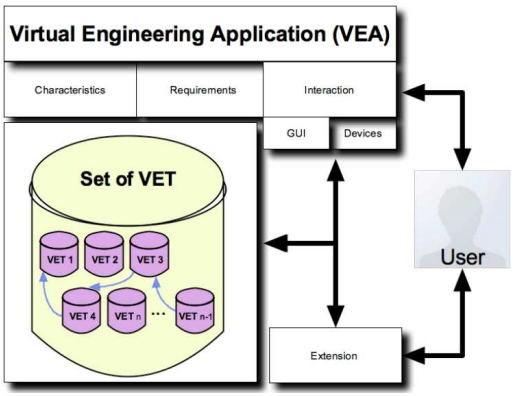


**Fig. 1.** According to [3], VEA are divided into 5 different parts: characteristics, requirements, interaction paradigms, underlying VET and extension capabilities.

In general, a VEA is composed by:

*(i)* A *Set of Characteristics*, which is the expected benefits of the VEA in terms of capabilities, features, accepted formats for input/output interactions.

*(ii)* A *Set of Requirements*, which is the minimum requisites that must hold the computer system where the VEA will be used.

*(iii)* A *Set of Interaction paradigms*, which is the different GUIs, input/output device characteristics (e.g. mouse, screen), etc.

*(iv)* A *Set of VET*, which is the collection of underlying tools which allow the fulfilment of the characteristics of the VEA.

*(v)* An *Extension Capabilities Mechanism*, which is generally provided via an API or scripting languages, allowing the programmatic extension of the VEA.

### 2.2 Classical Approaches in Interoperability between Virtual Engineering Applications

The interoperability between VEA is approached classically through data exchange. As each VEA support its own native format for the serialization of its internal data, it is necessary to use a translator between native formats. Translators are often specific for certain needs and no extrinsic knowledge is arguably translated from VEA to VEA, resulting in problems when a new functionality is available (a new VET for example) and producing inoperativeness until an updated version is presented [6].

Translators do not render a complete solution because suppliers provide only compatibility for a small set of features and in many cases the translation results will make no sense as different perspectives are not taken into account. One example of the aforementioned scenario could be the case of a pipe element designed in CAD and the same element considered in a cost handling VEA, although involving the same object, the translation is meaningless because the perspective changes. In some cases, the VEA supplier provides APIs for the development of new translators [7], [8], but this is arguably not a common practice due to marketing reasons or legacy systems.

Another solution reported, is related to the use of open standards for data exchange. Examples of these are IGES [9] for graphics or EDI [10] for ERP. These open standards provide a way to share information, but such information is far from the concept of knowledge as specialised relationships are lost in the translation, resulting in semantics loss as described in [2].

Semantic interoperability is one of the newest approaches in the state of the art. It specifically aims to the development of supporting applications with the ability to automatically interpret the information exchanged meaningfully and accurately producing useful results. Ontologies allow information exchange approaching this kind of interoperability [4]: the cases of e-Learning [11], Electronic Commerce [12] or Geographic Information Systems [13] are well documented examples of the success of this kind of interoperability.

### 2.3 Ontologies

Ontologies play a fundamental role in the Semantic Web paradigm [14]. In the Computer Science domain, the widely accepted definition (given by Gruber) states, "an ontology is the explicit specification of a conceptualization" [5]. In other words, it is a description of the concepts and their relationships in a domain of study. Fikes [15] identifies four top-level application areas where ontologies are applicable: (*i*) collaboration, (*ii*) interoperation, (*iii*) education and (*iv*) modelling. Within the VE domain, Mencke [6] considers three major areas where ontologies can be used: (*i*) Virtual Design, referred to the construction of virtual prototypes and their use with applications for controlling, monitoring and management, (*ii*) Test & Verification, referred to the simulation for checking the correctness and applicability of the Virtual Prototypes, and (*iii*) Visualization & Interaction, referred to the presentation of virtual objects and the user interaction.

The use of ontologies is validated in our proposal through the fact that in a PLC, each one of the involved VEA comprises a different level of specificity; a fact that leads to the need of a strong knowledge base paradigm if no semantic loss is desired. In our approach, we propose the use of a supporting knowledge base whose design language should be strong enough to contain the particularities of every VEA involved.

The aforementioned Supporting Knowledge base should provide mechanisms to reason and be queried about its contents, and at the same time it should be flexible enough to accept the inclusion of new relationships on the fly (in order to support the addition of new VET). Ontologies allow the representation of such Knowledge Base, including also stability checking after on-the-fly changes.

## 3 Proposed Architecture

A traditional PLC information flow is depicted in Fig. 2.a. We propose a supporting Knowledge Base that will store and handle the gathered knowledge following a centralized way as shown in Fig. 2.b. In our proposed Knowledge Base any supporting VEA will be able to contribute with its specific knowledge to the conceptual representation (for example, CAD provides the geometry, CAM provides manufacturing tool paths and FEA provides failure analysis).

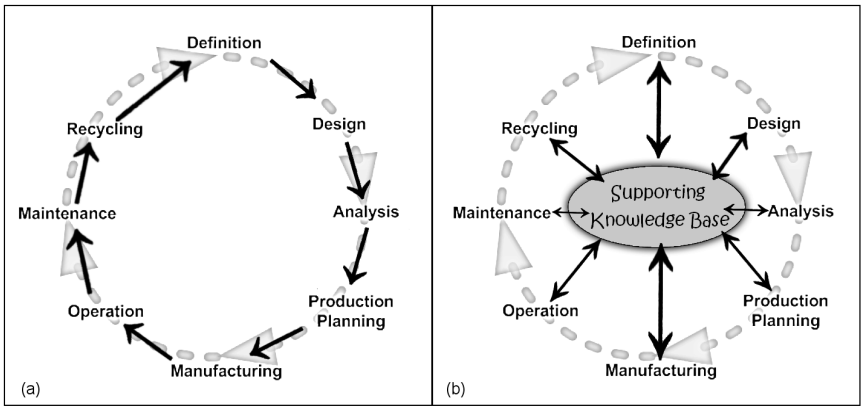


**Fig. 2.** Representation of data flow through PLC. (a) presents the classical approximation, and (b) shows the data flow using a support Knowledge Base.

In order to implement the supporting Knowledge Base, we propose the architecture depicted in Fig. 3.

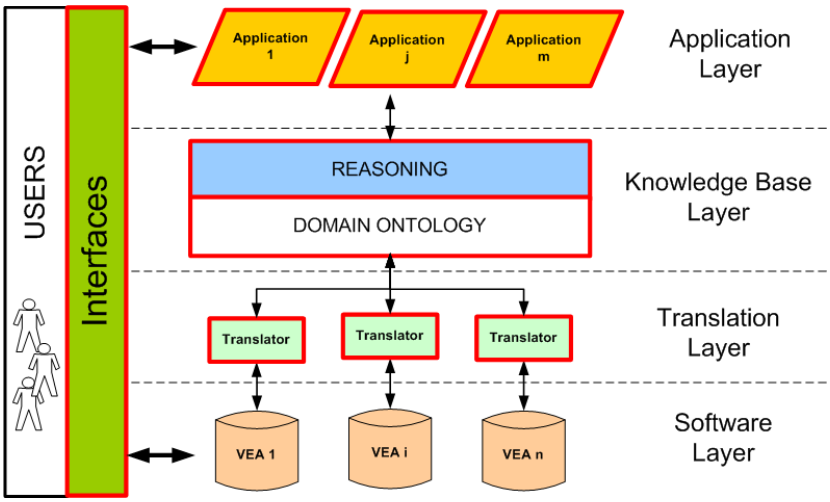


**Fig. 3.** Proposed architecture for the implementation of the supporting KB. Data from the Software layer is translated in order to feed the Knowledge Base.

In our architecture, the first layer is called the ***Software layer***; it contains the collection of VEA available in the different PLC stages. These VEA interact with users in a classical way (through their own interfaces). In general VEA in this layer possess extension capabilities that allow the possibility to access their embedded knowledge.

The next layer is called ***Translation layer***, in this part of the architecture takes place the alignment and matching between VEA generated knowledge and the Domain ontology located in the next layer. It can be argued that this translator provide the means of matching the VEA knowledge with the supporting Knowledge structure, for such reason, the importance of choosing a good model for the Domain is critical as problems like information incompleteness and multiple sources leading to redundancy should be considered.

Each one of the VEA in the Software layer is associated to its own translation module. Translators allow the bidirectional traffic of the knowledge: they are able to convert the knowledge generated in the VEA to the Domain ontology, and vice versa, from the ontology to the VEA format. Translator construction can be made in several ways, being the most usual using an API, or the parsing of the generated output files.

The following layer is called the ***Knowledge Base layer***; this part of the architecture contains the Domain ontology itself. All the gathered knowledge generated by each VEA is stored and managed here. The Semantic layer also contains a Reasoner for the exploitation of extrinsic knowledge [16] contained through a reasoning API [17] which is commonly a Descriptive Logics (DL) handler [18] who processes the Domain ontology.

At the top of the architecture is the ***Application layer***, where a series of applications capable to interact with the KB in a direct way can be produced. Such applications take advantage of all the stored knowledge while interact with thought their own interfaces.

## 4 Case Study

As case study, we developed within the frame of a research project; an application for the design of industrial plant layouts. In such scenario, many teams of engineers participate in the design of the plant with different VEA and in the different stages of the PLC producing a situation that is arguably prone to present a semantic loss.

In our scenario, the goal is the consideration of a new product that will be manufactured in the aforementioned facility. The Industrial Plant has to be adapted in one or several of the production lines already in existence in order to manufacture the new product.

Our visualization tool makes use of the presented architecture in order to provide an effective bridge over the gap of knowledge loss between the different VEA utilized. Fig. 4 depict a relation of our case of study and the generalized architecture presented in section 3. In order to simplify the example, we will consider only three VEA in the ***Software layer***: AutoCAD, used for the static geometries design (located in the design stage of the PLC mainly), RobCAD, used for the kinematics calculation and moving objects simulation(located in the analysis and operation stages of the PLC), and a XML serialized file which contains diverse database gathered information in the form of stored facts about the actual plant (located in the operation and maintenance stages of the PLC). We make a differentiation between static elements (e.g. walls, floor, fixed objects), and non-static elements (e.g. a robotic arm) in order to recognize easily the knowledge that must be involved in the re-calculation

duties during the simulation of the plant layout operation. Each VEA used possesses a translator in order to match the generated knowledge to the Domain ontology structure (***Translation layer***). AutoCAD provides an API called ObjectARX [19], which allows programmatically the parsing of the different 3D models, contained.

In the case of RobCAD, the vendor does not provide any public API, but instead a plug-in that allows exporting generated animated models into the VRML format. Even if such files have some non-standardized labels, we developed a VRML parser in order to extract and match the knowledge contained in those files into the ontology (for the extraction of the movements themselves).

In a similar way, we developed an XML parser to recover the information stored in the XML Generator output files, matching this information in the ontology.

In [20] the advantages of using Engineering Standards as a basis of ontology Domain modelling are detailed. Being one of these advantages the avoidance of semantic loss, the use of engineering standards is not only validated for our solution but highly desired.

Following the procedure to create Domain ontologies based on standards presented in [20], we have found in the domain of Plant Design reported success cases with the standard ISO-STEP (10303ap227) [21]. Our Domain Model in the ***Knowledge Based layer*** was hence serialized using the aforementioned approach adding also a second ontology for the extension of the STEP protocol to particularities of the problem at hands (again following the methodology presented in [20]).

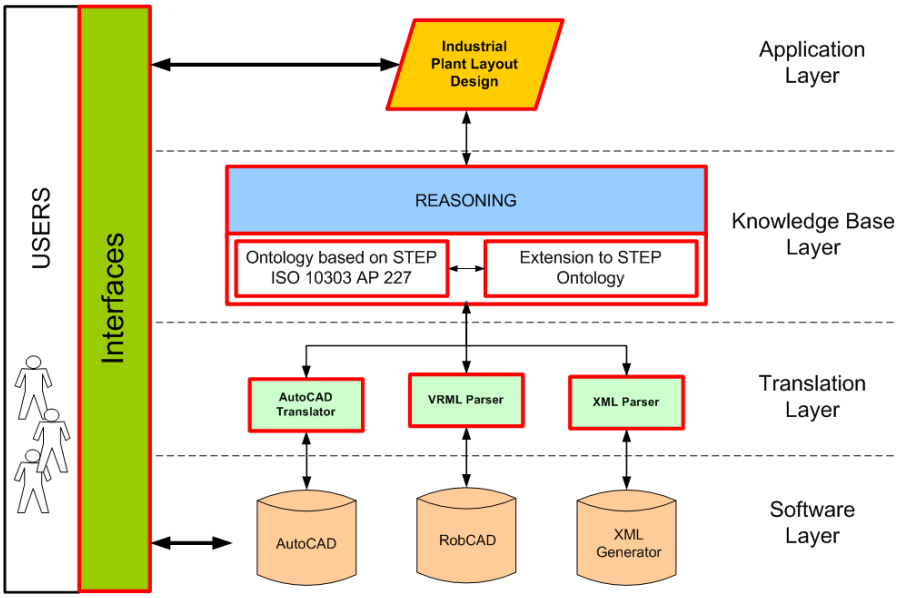


**Fig. 4.** Case study matched in the proposed architecture. Three different VEA are located in the software layer, each one with its own translator for KB feeding.

For the reasoning part, we created a simple interface that uses Pellet [22] via the Protégé OWL API [23].

At this point, we have developed our desired Industrial Plant Layout Design application in the ***Application layer***. The developed application allows the user the examination of different layouts for the same Industrial Plant in a 3D environment by navigating through the plant and modifying the mentioned layout in order to obtain the best possible configuration with the aid of the semantic engine that will not permit configurations that contradict design principles (using for that purpose the reasoner capabilities). The User can also obtain additional information from the elements contained in the active layout, for example the security area needed for the robot *KUKA KR 150-2* or the costs associated to the breakdown of painting process manufacture call.

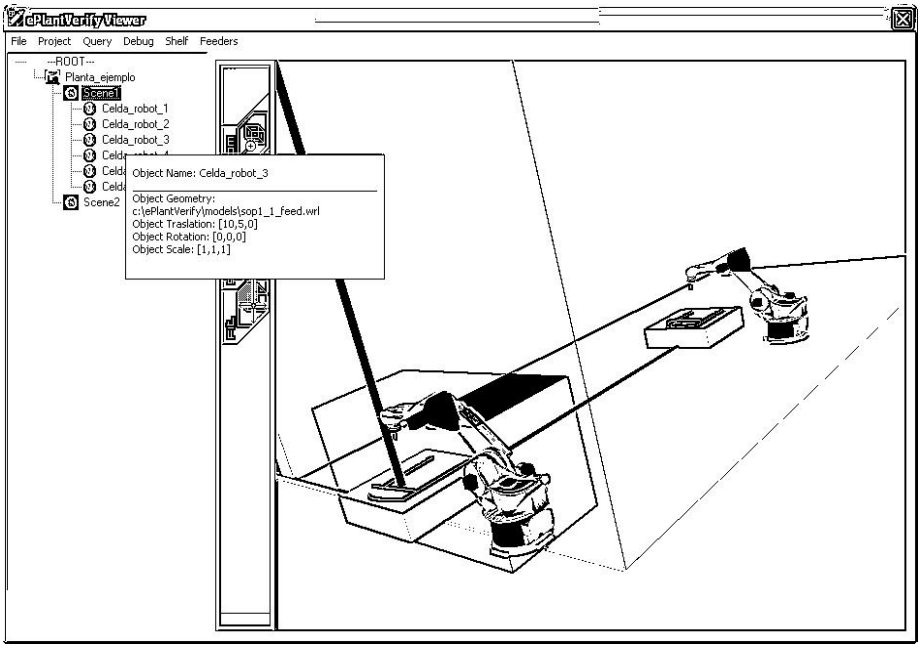


**Fig. 5.** Industrial Plant Layout Design application screenshot. The interface presents a 3D representation of the layout.

As can be seen in the Fig. 5, the application interface shows the three-dimensional representation of the active layout, and the different layouts (with their corresponding elements) in a tree mode. When the mouse is over an element in the tree, the system asks to the knowledge base for the position, translation and scale of the concrete instance of the 3D model and presents the information to user. If the user needs other information about any specific element or layout, he/she has possibility to launch a query to the knowledge base from query menu. Such query is handled by the reasoner over the instances of the ontology.

## 5 Conclusions and Future Work

In this paper, we presented a semantic based architecture that allows a seamless information and knowledge sharing between Virtual Engineering Applications in a Product Life Cycle scenario. We discussed the main highlights of the VEA interoperability problem and validated our presented approach with a test case consisting on the implementation of a Plant Layout Design application supported in our architecture. Our implementation is effective in collecting knowledge provided by different VEA available proving that our approach could be applicable within the industry. At this time, the presented system is being tested in order to offer a deep statistical analysis in subsequent works.

As future work, we intend to extend our implementation to more VEA in order to do an effective checking on the economical impact that having a centralized knowledge structure will provide. Also we are currently working on the use of the centralized Knowledge gathered within a Semantic Web application running as a service in a company web page in order to provide means for remote monitoring of an industrial plant.

**Acknowledgments.** We would like to express our gratitude to the Basque Government for the partial funding of the project ePlantVerify where the main techniques presented in this article were developed (under INNOTEK 2007-2010 grants call), and the partial funding of PCT/ES 2010/000009, where our system is in the process of a research patent concession. We also want to thank Inge-Innova and Inertek 3D as partners of the consortium of the aforementioned research initiatives.

## References

1. Jian, C.Q., McCorkle, D., Lorra, M.A., Bryden, K.M.: Applications of Virtual Engineering in Combustion Equipment Development and Engineering. In: ASME International Mechanical Engineering Congress and Expo, IMECE2006–14362, ASME, ASME Publishers. (2006)
2. Zorriassatine, F., Wykes, C., Parkin, R., Gindy, N.: A survey or virtual prototyping techniques for mechanical product development. Journal of Engineering manufacture 217, 513--530 (2003)
3. Toro, C.: Semantic Enhancement of Virtual Engineering Applications. PhD Thesis, University of the Basque Country, Spain. (2009)
4. Obrst, L.: Ontologies for Semantically Interoperable Systems. In: CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management, pp. 366--399. ACM, New York, NY, USA. (2003)
5. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies 43(5-6), 907--928 (1995)
6. Mencke, S., Vornholt, S. Dumke, R.: The Role of Ontologies in Virtual Engineering. In: 19th International Conference on Database and Expert Systems Application, pp. 95--98. IEEE Computer Society. (2008)
7. ObjectARX. URL: http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=773204 (Last visited March 18, 2010)
8. Solidworks 2003: API Fundamentals. Solidworks Corporation (2001)

9. Initial Graphics Exchange Specification (IGES). URL: http:// tx.nist.gov/standards/iges (Last visited March 18, 2010)
10. Electronic Data Interchange (EDI). Federal Information Processing Standards Publication 161-2. URL: http://www.itl.nist.gov/fipspubs/fip161-2.htm (Last visited March 18, 2010)
11. Busse, S.: Interoperable E-Learning Ontologies Using Model Correspondences. In: On the Move to Meaningful Internet Systems 2005: OTM Workshops. LNCS 3762, pp: 1179--1189. Springer Berlin, Heidelberg. (2005)
12. Obrst, L., Liu, H., Wray, R., Wilson, L.: Ontologies for Semantically Interoperable Electronic Commerce. In: nternational Conference on Enterprise Modelling and Enterprise Integration Technologies (ICEIMT), and the Conference of the EI3-IC Initiative (Enterprise Inter- and Intra-Organisational Integration – International Consensus), Valencia, Spain, April 24-26, 2002, pp. 366--399. Kluwer Academic Publishers. (2002)
13. Fonseca, F., Câmara, G., Monteiro, A. M.: A Framework for Measuring the Interoperability of Geo-Ontologies. Spatial Cognition and Computation 6 (4): 309– 331. (2006)
14. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, 284 (5), 34--44. (2001)
15. Fikes, R.: Multi-Use Ontologies, Stanford University, URL: http://www-ksl.stanford.edu/ people/fikes/cs222/1998/Ontologies/sld001.htm (Last visited March 18, 2010)
16. Sattler, U: Description Logic Reasoners. URL: http://www.cs.man.ac.uk/~sattler/ reasoners.html
17. Using the Protégé-OWL Reasoner API. URL: http://protege.stanford.edu/plugins/owl/api/ ReasonerAPIExamples.html
18. Baader, F., Horrocks, I., Sattler, U.: Description Logics. Handbook of Knowledge Representation. Elsevier. (2007)
19. Kramer, B.: ObjectARX Primer. Autodesk Press. (1999)
20. Toro, C., Graña, M., Posada, J., Vaquero, J., Sanin, C., Szczerbicki, E.: Domain Modeling Using engineering Standards. In: Knowledge-Based and Intelligent Information and Engineering Systems. 13th International Conference, KES 2009, Santiago, Chile, September 28-30, 2009, Proceedings, Part I. LNCS 5711, pp. 95--102. Springer Berlin, Heidelberg (2009)
21. SCRA: STEP Application Handbook ISO 10303, Version 3. (2006) Available in http://www.uspro.org/documents/STEP_application_hdbk_63006_BF.pdf (Last visited March 18, 2010)
22. Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 5 (2), 15--53. Elsevier. (2005)
23. Protégé OWL API. URL: http://protege.stanford.edu/plugins/owl/api/ (Last visited March 18, 2010)