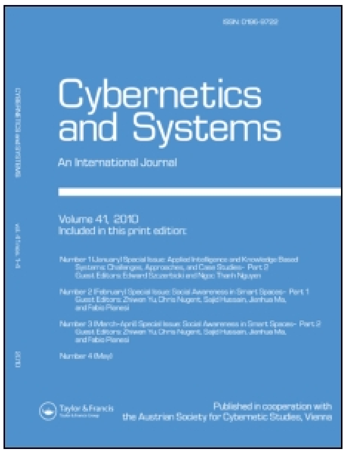


Cybernetics and Systems
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713722751

PRESERVING VIRTUAL ENGINEERING KNOWLEDGE THROUGH THE PRODUCT LIFE CYCLE

Javier Vaquero[a]; Carlos Toro[a]; Manuel Graña[b]
[a] Vicomtech Research Centre, San Sebastian, Spain [b] Computational Intelligence Group, University of the Basque Country (UPV-EHU), San Sebastian, Spain

Online publication date: 10 March 2011

To cite this Article Vaquero, Javier , Toro, Carlos and Graña, Manuel(2011) 'PRESERVING VIRTUAL ENGINEERING KNOWLEDGE THROUGH THE PRODUCT LIFE CYCLE', Cybernetics and Systems, 42: 2, 124 — 138
To link to this Article: DOI: 10.1080/01969722.2011.541212
URL: http://dx.doi.org/10.1080/01969722.2011.541212

PLEASE SCROLL DOWN FOR ARTICLE

Cybernetics and Systems: An International Journal, 42:124–138
Copyright © 2011 Taylor & Francis Group, LLC
ISSN: 0196-9722 print/1087-6553 online
DOI: 10.1080/01969722.2011.541212

Preserving Virtual Engineering Knowledge Through the Product Life Cycle

JAVIER VAQUERO[1], CARLOS TORO[1], and MANUEL GRAÑA[2]
[1]*Vicomtech Research Centre, San Sebastian, Spain*
[2]*Computational Intelligence Group, University of the Basque Country (UPV-EHU), San Sebastian, Spain*

Virtual engineering applications (VEA) are used through the whole product life cycle (PLC) process; their complexity and pervasiveness make them an ideal scenario for development and testing of software engineering innovations. More precisely, VEAs have problems when they need to share their specific knowledge. They suffer semantic loss situations, and they hardly use any semantic tools. This work proposes a new approach to solving these kinds of problems based on semantic technologies, allowing the seamless sharing of information and knowledge between VEAs involved in a PLC scenario. This approach is validated through a plant layout design application, where several VEA can cooperate and share their knowledge, accomplishing the design task.

KEYWORDS *ontologies, product life cycle, virtual engineering*

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 (Jian et al. 2006). Virtual engineering applications (VEA) are the software implementations supporting VE. Each VEA, in its turn, contains a set of virtual engineering

The authors express their gratitude to the Basque Government for the partial funding of the project ePlantVerify under the INNOTEK 2007–2010 grants call and Inge-Innova and Inertek 3D as partners of the consortium of the research initiatives.

Address correspondence to Carlos Toro, Vicomtech Research Centre, Mikeletegi Pasealekua 57, Donostia, San Sebastian 20009, Spain. E-mail: ctoro@vicomtech.es

tools (VET), which is the collection of features that the VEA offers; for example, in a computer-aided design (CAD)-like VEA the graphical tools to draw lines or circles.

Current VEAs barely make use of the information about contextual facts, user requirements, user experience, and, in general, any useful factor that could be easily modeled and exploited from a semantical point of view. For example, we have already studied how a VEA becomes friendlier if the graphic user interface is adapted to the individual users, showing them only the tools and windows they need depending on the role they are playing in the development cycle (Toro et al. 2007). In another case study, we developed an application that may suggest actions to be carried out by the users, taking into account previous experiences obtained from different experts in similar situations (Vaquero et al. 2009).

From a product life cycle (PLC) perspective, the use of different VEAs in each one of the stages is a common practice. Examples are CAD in the design stage or finite element algorithms (FEAs) in the analysis stage. Nevertheless, some VEAs may prove to be valuable at several PLC stages. However, in this scenario geometry is the only feature preserved when transferring information from one VEA to another. Losing ancillary information corresponding to the geometric features may incur a semantic loss (Zorriassatine et al. 2003); *semantic loss* is defined as the generated knowledge lost when transferring from the original VEA to another (or even from the original VEA to itself at a later PLC stage). Such loss is produced because knowledge is implicit to the generating VEA, and minor changes in the domain (for example, FEA and CAM (Computer Aided Manufacturing) do not understand the geometry generated by CAD in the same way) cause a poor translation to the VEA that is trying to obtain the knowledge. Sharing information and knowledge between VEAs has become an important handicap for system development due to several reasons, which include commercial interests of the manufacturer, the nature of legacy products, and language incompatibilities. To achieve a solution to this problem, we propose a system architecture and implementation where all the knowledge generated by a VEA during a PLC by way of its component VETs is stored in a centralized knowledge repository based on semantic technologies. This repository is persistently accessible to the VEAs, providing semantic support during the whole PLC, causing an information redundancy drop while reducing important costs associated with format repair and semantic loss.

In the following section we present the background concepts necessary for the understanding of the proposed approach. In the next section we describe the proposed ontology-based software architecture to solve the semantic loss problems during the PLC. Then we show a system designed following the proposed architecture that is applied to industrial plant design layout tasks. Finally, we present our conclusions and future work lines.

BACKGROUND CONCEPTS

In this section we present an overview of concepts relevant to this work. The reader will find more detailed explanations of the concepts presented in Toro (2009), Obrst (2003), and Gruber (1995).

Virtual Engineering Applications

As stated in the introduction, VEAs, as software implementations of VE concepts, are arguably the main facilitators of a new product development. In Figure 1, the components of a VEA are presented following the categorization introduced in Toro (2009).

In general, a VEA is composed of the following:

1. A *set of characteristics*, which is the expected benefits of the VEA in terms of capabilities, features, and accepted formats for input–output interactions. The set of characteristics defines a general overview of the affordances of the VEA.
2. A *set of requirements*, which is the minimum requisites that must fulfill the computer system where the VEA will be used. These requisites refer to both hardware and software.
3. A *set of interaction paradigms*, which are the places where the user executes the input and output interface actions, including the different GUIs, (Graphical User Interface), input–output device characteristics (e.g., mouse, screen), etc. Some examples are the use of multiple views in the same display or the capability to accept voice commands.

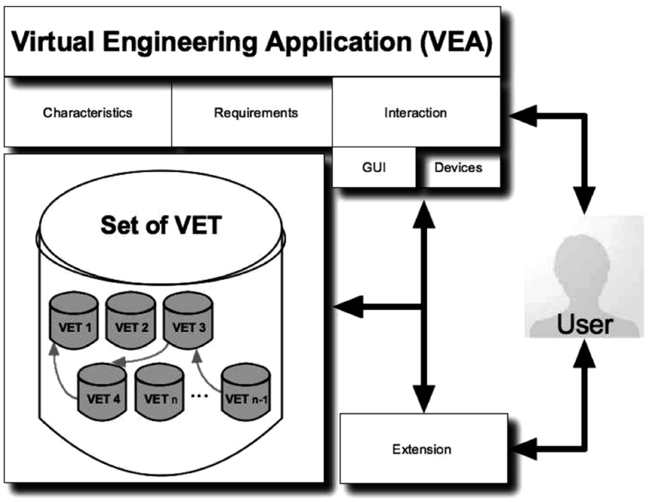


FIGURE 1 Generic VEA components (Toro 2009).

4. A *set of VETs*, which is the collection of computational tools that allow the fulfillment of the VEA's affordances. The set of VETs can be composed of elementary tools that can be used in conjunction with other tools or in isolation when necessary.
5. A *mechanism for the extension of the VEA's capabilities*, which allows the extension of the functionality of the VEA. The extension capability is generally provided via an API (Application Programming Interface) or scripting languages. As a general rule, it is easier to enhance the VEA when this API is more open.

In most cases, the main reason for acquiring or using a VEA is the comparison of the affordances defined by its set of characteristics against the set of user requirements. In addition, in many cases it is very important to have the possibility to extend VEA functionalities, and extension mechanisms must be taken into account in the VEA evaluation.

Conventional Approaches in Interoperability between Virtual Engineering Applications

Interoperability is defined by the Institute of Electrical and Electronics Engineers *IEEE Computer Dictionary* as the "ability of two or more systems or components to exchange information and to use the information that has been exchanged" (IEEE 1991, p. 114). From such a definition, interoperability can be decomposed into two distinct components: the ability to exchange information, denoted as *syntactic interoperability*, and the ability to use the information when received, denoted as *semantic interoperability*.

The syntactic interoperability between VEAs is a fundamental issue of the development of VE approaches to PLC. Because each VEA supports its own native format for the serial storage of its internal data, it is often necessary to use a translator between native formats. Translators are often specific to certain needs, and no additional knowledge is transferred from VEA to VEA, resulting in semantic loss when a new functionality is available (a new VET, for example), thus producing reduction in operation efficiency until an updated version of the translator is presented to the user (Mencke et al. 2008). In other words, generated ancillary information that could be used by another VEA is lost because it is not contemplated by an outdated translator.

There are a variety of reasons why commercial format translators do not offer a complete solution to this semantic loss. In most cases, VEA suppliers can only provide compatibility for a small set of features. It could be expected that using VEAs from the same company guarantees full compatibility. However, it is a common practice for large vendors to buy small companies and commercialize their products. This fact implies that, even though they come from the same provider, some VEAs may have only partial compatibility between them. Moreover, even when the supplier guarantees compatibility, we may find out that the translation is not correct and part of the VEA-generated knowledge is lost in translation. As an example, it is not unusual to export correctly the geometry of a model while losing its kinematics. In some cases, the VEA supplier provides APIs for the development of new custom-made translators (Autodesk 2010; Solidworks 2010), but this is not a common practice due to marketing reasons or legacy systems.

Another solution to the semantic loss problem reported in the literature is founded in the use of open standards for data exchange. Examples of these are the use of IGES (Initial Graphics Exchange Specification) (Kemmerer 2001) for graphics information transference or electronic data interchange (EDI; Kantor and Prabhakar 1996) for ERPs (Enterprise Resource Planning). These open standards provide a way to share information, but such information is far from the concept of knowledge as long as specialized relationships are lost in the translation, resulting in semantics loss as described in Zorriasatine et al. (2003).

Semantic interoperability is one of the newest approaches in the state of the art of VE for the PLC. It specifically aims for the development of supporting applications with the ability to automatically, meaningfully, and accurately interpret the information exchanged, producing useful results. Ontologies allow information exchange approaching this kind of interoperability (Obrst 2003): the cases of E-learning (Busse 2005), electronic commerce (Obrst et al. 2002), or geographic information systems (Fonseca et al. 2006) are well-documented examples of the success of this kind of interoperability.

Semantics and Ontologies

Semantics is a concept derived from the Greek term *semantikos*, which means *significant*, and it is the discipline that studies the meaning of things. The first appearance in its modern form was in Michel Bréal's book *Essai de Sémantique* in 1897 (Bréal 1897). In our work in this article, semantic technologies are used to improve interoperation among VEAs. Interoperability issues are formulated as a problem of semantic and information loss. There is a semantic loss when the ancillary information containing the meaning of the information objects is lost during the transference among VEAs. Semantic loss implies that some functionality of the VEA could not be applied to the data.

Modeling the application domain with ontologies, we can use their associated inference capabilities to allow the efficient design of the information transference channels, reducing the information and semantic loss.

Ontologies play a fundamental role in the semantic Web paradigm (Berners-Lee et al. 2001). In the computer science domain, the widely accepted definition states that "an ontology is the explicit specification of a conceptualization" (Gruber 1995, p. 908). In computer and information sciences, an ontology is a formal representation of knowledge as the set of concepts identified within a domain and their relationships. It is used to reason about the entities within that domain and may be used to describe the domain, providing a shared, agreed-upon vocabulary.

Fikes (1998) distinguishes four top-level application areas where ontologies are useful: (1) collaboration, (2) interoperation, (3) education, and (4) modeling. More specifically, within the VE domain, Mencke et al. (2008) considered three major areas where ontologies can be used: (1) virtual design, referring to the construction of virtual prototypes and their use with applications for controlling, monitoring, and management; (2) testing and verification, referring to the simulation for checking the correctness and applicability of the virtual prototypes; and (3) visualization and interaction, referring to the presentation of virtual objects and the user interaction.

The use of ontologies is motivated in our proposal by the fact that during a PLC, each one of the involved VEAs comprises a different level of specificity, a fact that leads to the need of a strong knowledge-based paradigm to avoid semantic loss. 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. This supporting knowledge base should provide mechanisms to reason and answer queries about its contents and at the same time it should be flexible enough to accept the inclusion of new relationships on the fly (i.e., in order to support the addition of new VET).

Ontologies, opposite to other modeling techniques, allow checking data properties after on-the-fly changes; that is, data integrity: a reasoner can verify the ontology structure automatically when new elements are added. Data integrity allows scalability, which is an important property for translation processes (for example, when a new version of the software is released, in many cases the translator needs to be updated). Therefore, ontologies are ideally suited for the representation of the proposed supporting knowledge base.

PROPOSED ARCHITECTURE

In the introduction section we presented some of the VEA interoperation problems that may result in semantic loss. In the knowledge/information flow of a traditional PLC scenario, depicted in Figure 2a, semantic loss sometimes arises when VEAs used in the same product development stage share their knowledge, but more often it happens when knowledge generated in one of the PLC stages flows to VEAs used in the next PLC stage. We propose a semantic-based centralized supporting knowledge base that will store and handle the gathered knowledge as shown in Figure 2b. Any VEA used during the PLC will be able to contribute with its specific knowledge to the conceptual representation in our proposed knowledge base (for example, CAD provides the geometry, CAM provides manufacturing tool paths, and FEA provides failure analysis).

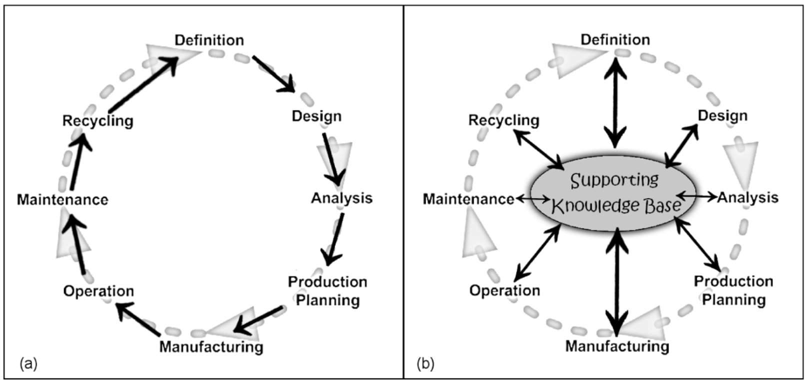


FIGURE 2 Representation of knowledge/information flow along the PLC: (a) conventional approximation and (b) using a supporting knowledge base.

More precisely, the supporting knowledge base is implemented through the architecture depicted in Figure 3.

In our architecture, the first layer is the *software layer*; it contains the collection of the different VEAs available in the different PLC stages. The user interacts with these VEAs in a classical way; that is, using the interfaces provided from each VEA, generating the diverse knowledge necessary in the PLC process. In general, VEAs in this layer possess extension capabilities that allow the possibility to access their embedded knowledge. This fact allows feeding the supporting knowledge base with the knowledge generated by the VEA or, conversely, increasing the VEA's knowledge with information coming from the supporting knowledge base when necessary.

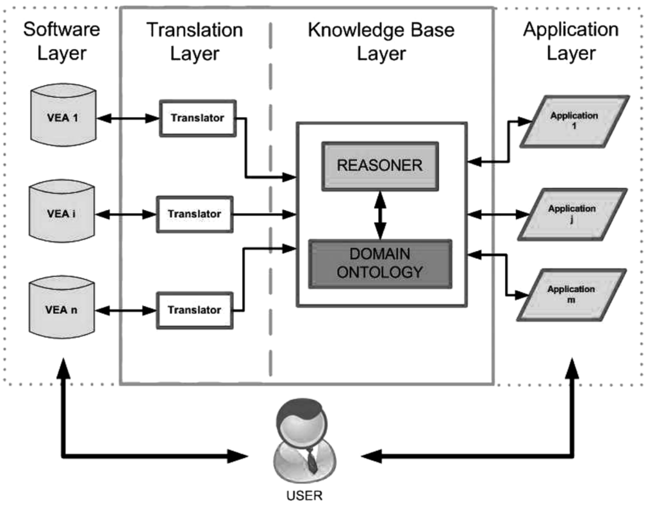


FIGURE 3 Proposed architecture for implementation of the supporting knowledge base.

The next layer is the *translation layer*; in this part of the architecture the alignment between VEA-generated knowledge and the domain ontology located in the next layer takes place. This translator provides the means of matching the VEA knowledge with the supporting knowledge structure; therefore, the importance of choosing a good domain's model is critical because problems like information incompleteness and multiple sources leading to redundancy should be considered.

Each VEA in the software layer is associated with its own translation module. Translators allow bidirectional traffic of the knowledge: they are able to convert the knowledge generated in the VEA into the knowledge base's domain ontology and, conversely, from the domain ontology into the VEA format. Translator construction can be made in several ways, the most usual being the use of an API, using a scripting language or the parsing of the generated output files.

The following layer is the *knowledge base* layer; this part of the architecture contains the domain ontology and the reasoner. All of the knowledge generated by each VEA is stored and managed at this stage. To achieve these goals it is important to have a deep knowledge of the specific problem domain as well as good ontology model building tools and methodology. For such reasons, it is arguably recommended the combined work of an ontology engineer and an expert on the specific domain of work and the use of engineering standards as presented in Toro et al. (2009).

The reasoner is responsible for three tasks:

1. Control of the ontology integrity; that is, check whether classes, attributes, and relationships fulfill the logic of the model. For example, for OWL-DL (McGuinness and van Harmelen 2004) models the logic used is description logics (DL).
2. Providing a query system, allowing the extraction of explicit knowledge stored in the ontology. The query system can be implemented with one of the existing query languages (e.g., SPARQL) or through the use of reflexive ontologies allowing the analysis of the queries and speed-up of the query process as presented in Toro et al. (2008).
3. Reasoning over the ontology; that is, the exploitation of implicit knowledge (Sattler 2010) contained in the ontology through a reasoning API (Protégé 2007), which is commonly a handler (Baader et al. 2007) who processes the domain ontology; for example, Pellet or RACER.

The last layer of the architecture is the *application layer*; it is reserved for custom applications capable of interaction with the supporting knowledge base in a direct way. Such interaction is bidirectional; applications have to be able to compose queries that are remitted to the reasoner of the knowledge base layer, and they also have to be capable of understanding the answer obtained by the reasoner to the given query. Bidirectional communication allows applications to get the stored knowledge and contribute to feeding the knowledge base with their new generated knowledge. Applications located in this layer take advantage of all of the stored knowledge. Nevertheless, this interaction is not necessarily visible to the user, who interacts with the application through his own interfaces in a transparent way.

CASE STUDY

Following the architecture presented in the previous section, we developed a solution to a case study posed within the framework of an applied industrial research project: the (re)design of industrial plant layouts. In such a scenario, many teams of engineers cooperate in the design of the plant using different VEAs at different stages of the PLC; hence many situations prone to semantic loss arise.

The goal is to produce a plant layout that will allow the manufacture of a new product. One or several of the production lines already in existence will need adaptation according to the manufacturing process needs of the new product.

The visualization tool developed to assist in the solution of the industrial plant layout design makes use of the architecture proposed in earlier to prevent semantic loss during interoperation between the different VEAs utilized. Figure 5 shows the instantiation of the general architecture described in the previous section to the details of our case study. In order to simplify the example, we will consider only three VEAs in the software layer, as can be seen in Figure 4: AutoCAD, used for the static geometric design (located in the design stage of the PLC mainly); RobCAD, used for the kinematics calculation and simulation of moving objects (located in the analysis and operation stages of the PLC); and an XML serialized file that contains a diverse database of 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 nonstatic elements (e.g., a robotic arm) in order to recognize the knowledge that must be involved in the recalculation duties during simulation of the plant layout operation. Nonstatic elements' geometries are modeled using AutoCAD in the design stage, and their kinematics are later added in the analysis stage using RobCAD. In this scenario, geometry is correctly exported, but additional knowledge generated in AutoCAD (for example, line thickness is related to the material) is lost, thus producing a semantic loss.

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 (Kramer 1999), which allows programatically parse the different 3d models contained.

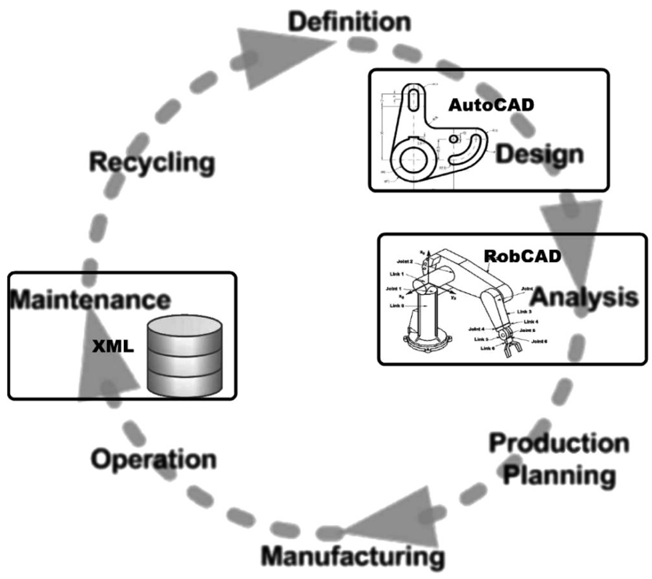


FIGURE 4 Localization of the case study VEAs in the PLC general process.

In the case of RobCAD, the vendor does not provide any public API but instead provides a plug-in that allows exporting generated animated models into the VRML (Virtual Reality Modeling Language) format. These files have some nonstandardized labels, but we developed a VRML parser in order to extract and match the knowledge contained in those files in 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.

Following the methodology to create domain ontologies based on engineering standards presented in Toro et al. (2009), it is necessary to choose an appropriate standard. For plant design domain, we have found reported successful approaches based on the standard ISO-STEP (10303ap227; SCRA 2006). Our domain model in the knowledge base layer was hence serialized using Toro et al.'s (2009) approach, adding a second ontology for the extension of the STEP protocol to particularities of the problem at hand (again following the methodology presented in Toro et al. [2009]).

As the knowledge base's reasoning component, we created a simple interface that uses Pellet (Sirin et al. 2005) via the Protégé OWL API (Protégé 2010).

At this point, we have developed our desired industrial plant layout design application in the application layer. The developed application allows the user to inspect different layouts of the same industrial plant in a 3D environment by navigating through the plant and modifying the layout in

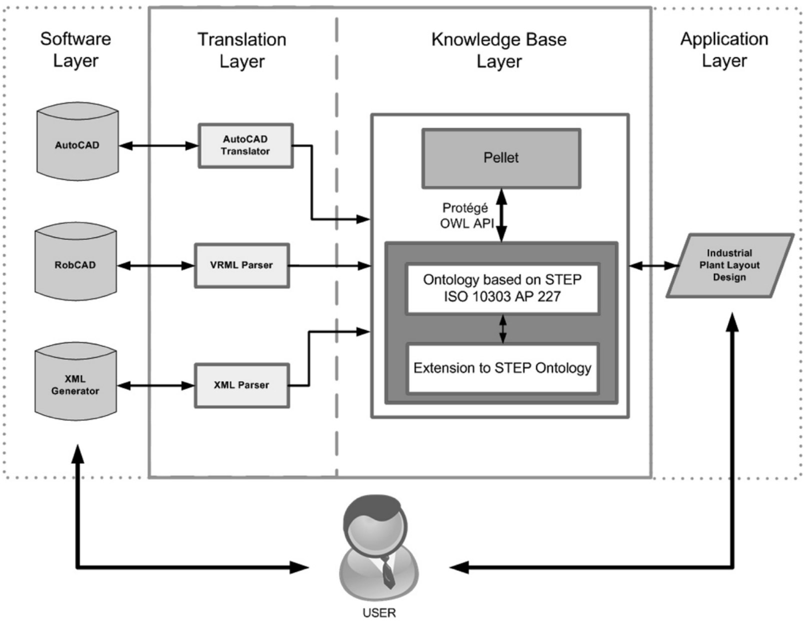
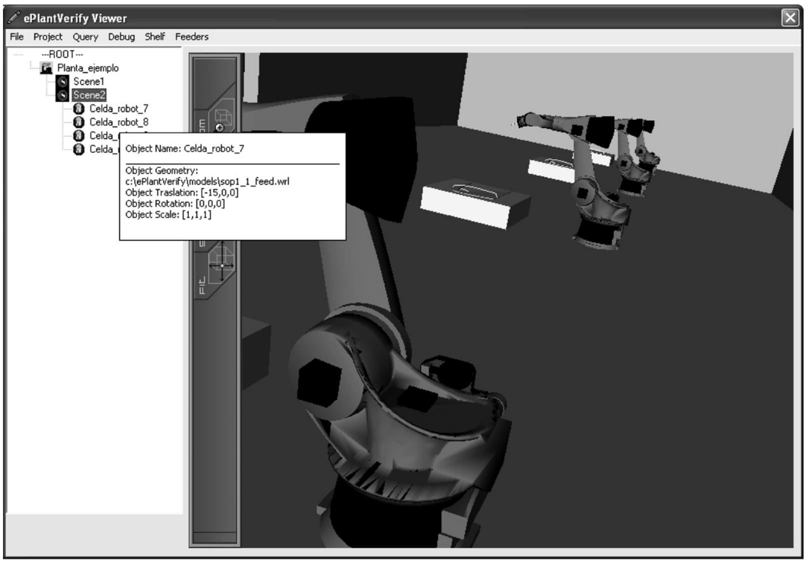


FIGURE 5 Case study matched in the proposed architecture.

order to obtain the best possible configuration with the aid of the semantic engine that will not permit configurations that contradict design principles (using the reasoner capabilities for that purpose). The user can also obtain



Object Name: Celda_robot_7

Object Geometry:
c:\ePlantVerify\models\sop1_1_feed.wrl
Object Traslation: [-15,0,0]
Object Rotation: [0,0,0]
Object Scale: [1,1,1]

FIGURE 6 Industrial plant layout design application screenshot: 3D representation of the layout.

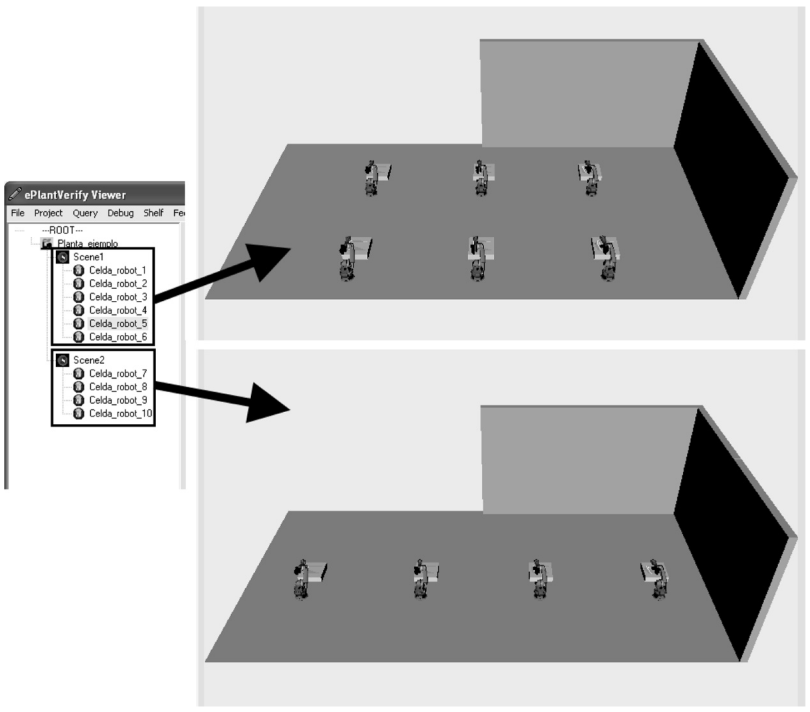


FIGURE 7 Industrial plant layout design application screenshot: multiple layouts.

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 with the breakdown of painting process manufacture cell.

As can be seen in Figure 6, the industrial plant layout design application interface shows the 3D representation of the active layout, and the different layouts (with their corresponding elements) are accessible from a tree-mode representation, as represented in Figure 7. When the pointer is over a tree element, the system asks the knowledge base for the position, translation, and scale of the concrete instance of the 3D model and displays the information to the user. If the user needs additional information about any specific element or layout, he or she has can launch a query to the knowledge base from the query menu. Such a query is handled by the reasoner over the instances of the ontology.

CONCLUSIONS AND FUTURE WORK

In this work, we faced the interoperability problem between VEAs through the product life cycle process and propose a semantic-based architecture that uses a supporting knowledge base to collect, store, and manage all of the knowledge generated in such VEAs, allowing a seamless information and knowledge sharing between the VEAs involved in a PLC scenario.

In order to validate our approach, we presented a test case where the knowledge base is modeled for an industrial plant layout design task. Three different VEAs feed the knowledge base, and each one of them has its own implemented translator. An industrial plant layout design application that launches queries to the knowledge base completes the test case.

Our implementation is effective in collecting knowledge provided by different VEAs available, proving that our approach could be applicable within the industry. At this time, the system presented is being tested in order to offer a deep statistical analysis in subsequent works.

As future work, our efforts are directed toward several different research lines. At the architectural level, we aim to enhance the supporting knowledge base of the architecture presented with decisional DNA as defined in Sanin et al. (2009). Current VEAs do not take into account the user's experience. From our point of view, experience could be a very useful information source for VEAs, not for making automatic decisions but to generate suggestions that the system can provide to users. According to Sanin et al. (2009), decisional DNA allows the storage of the day-to-day explicit experience in a single structure and predicting capabilities based on the collected experience (among other advantages such as transportability and shareability of the knowledge or versatility and dynamic nature of the knowledge structure).

On the other hand, we plan to extend the presented case study system with other VEAs and replace the Pellet reasoner by reflexive ontologies, a technology that presents several advantages like the speed-up of the query process or the addition of extra knowledge about the domain provided by previously computed queries and answers (Toro et al. 2008). With this change, we expect to improve the knowledge base efficiency, as reported in Cobos et al. (2008). After this change, we will study the economical impact that this plant layout design system could have in an engineering small-to-medium enterprise (SME).

Finally, we are also 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 the means for remote monitoring of an industrial plant.

REFERENCES

Autodesk. 2010. *Autodesk—Developer Center—ObjectARX*. Available at: http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=773204 (accessed September 26, 2010).

Baader, F., Horrocks, I., Sattler, U. 2007. Description logics. In *Handbook of knowledge representation*, edited by F. Van Harmelen, V. Lifschitz, and B. Porter. Amsterdam, Netherlands: Elsevier.

Berners-Lee, T., Hendler, J., Lassila, O. 2001. The semantic Web. *Scientific American* 284(5): 34–44.

Bréal, M. 1897. *Essai de sémantique*. Librairie Hachette, Paris, France.

Busse, S. 2005. Interoperable E-learning ontologies using model correspondences. *Lecture Notes in Computer Science* 3762: 1179–1189.

Cobos, Y., Toro, C., Sarasua, C. Vaquero, J., Linaza, M. T., Posada, J. 2008. An architecture for fast semantic retrieval in the film heritage domain. In *2008 International Workshop on Content-Based Multimedia Indexing, CBMI 2008, Conference Proceedings*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Fikes, R. 1998. Multi-use ontologies. Available at: http://www-ksl.stanford.edu/people/fikes/cs222/1998/Ontologies/sld001.htm (accessed September 26, 2010).

Fonseca, F., Câmara, G., Monteiro, A. M. 2006. A framework for measuring the interoperability of geo-ontologies. *Spatial Cognition and Computation* 6(4): 309–331.

Gruber, T. R. 1995. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* 43(5–6): 907–928.

Haarslev, V., Muller, R. 2003. Racir: An OWL reasoning agent for the semantic web. In Proceedings of the International Workshop on Application, Products, and Services of Web-Based Support Systems, in conjunction with 2003 IEEE/WIC International Conference on Web Intelligence, pp. 91–95, edited by J. T. Lingras. Department of Mathematics and Computing Science, Saint Mary's University, Halifax, Canada.

Institute of Electrical and Electronics Engineers. 1991. *IEEE Computer Dictionary—Compilation of IEEE Standard Computer Glossaries, 610-1990*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Jian, C. Q., McCorkle, D., Lorra, M. A., Bryden, K. M. 2006. Applications of virtual engineering in combustion equipment development and engineering. In *ASME International Mechanical Engineering Congress and Expo*, New York, NY: ASME Publishers.

Kantor, M., Prabhakar, A. 1996. *Electronic data interchange (EDI)*. Federal Information Processing Standards Publication 161-2. Available at: http://www.itl.nist.gov/fipspubs/fip161-2.htm (accessed September 25, 2010).

Kemmerer, S. J. 2001. Initial graphics exchange specifications. In *A century of excellence in measurements, standards, and technology*, edited by D. R. Lide. Boca Raton, FL: CRC Press.

Kramer, B. 1999. *ObjectARX Primer*. Albany, NY: Autodesk Press.

McGuinness, D. L., van Harmelen, F. 2004. OWL Web Ontology Language Overview. W3C Recommendation. Available at: http://www.w3.org/TR/owl-features/. (Accessed 8 February 2011.)

Mencke, S., Vornholt, S., Dumke, R. 2008. *The role of ontologies in virtual engineering*. In *19th International Conference on Database and Expert Systems Application*. Washington, D.C.: IEEE Computer Society.

Obrst, L., Liu, H., Wray, R., Wilson, L. 2002. Ontologies for semantically interoperable electronic commerce. In *Enterprise inter- and intra-organisational integration: Building international consensus*, edited by K. Kosanke, R. Jochem,

J. G. Nell and A. Ortiz Bas. Deventer, The Netherlands: Kluwer Academic Publishers.

Obrst, L. 2003. *Ontologies for semantically interoperable systems*. In *CIKM '03: Proceedings of the Twelfth International Conference on Information and Knowledge Management*. New York: Association for Computing Machinery.

Protégé. 2007. *Using the Protégé-OWL Reasoner API*. Available at: http://protege.stanford.edu/plugins/owl/api/ReasonerAPIExamples.html (accessed September 26, 2010).

Protégé. 2010. *Protégé OWL API*. Available at: http://protege.stanford.edu/plugins/owl/api/ (accessed September 25, 2010).

Sanin, C., Mancilla-Amaya, L., Szczerbicki, E., CayfordHowell, P. 2009. Application of a multi-domain knowledge structure: The decisional DNA. In *Intelligent systems for knowledge management*, edited by N. T. Nguyen and E. Szczerbicki. Heidelberg, Germany: Springer.

Sattler, U. 2010. Description logic reasoners. Available at: http://www.cs.man.ac.uk/~sattler/reasoners.html (accessed September 25, 2010).

SCRA. 2006. *STEP Application Handbook ISO 10303, Version 3*. Available at: http://www.uspro.org/documents/STEP_application_hdbk_63006_BF.pdf (accessed September 25, 2010).

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., Katz, Y. 2005. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* 5(2): 15–53.

Solidworks. 2010. 2010 Solidworks API Help. Available at: http://help.solidworks.com/2010/English/api/sldworksapiprogguide/Welcome.htm (accessed September 26, 2010).

Toro, C. 2009. Semantic enhancement of virtual engineering applications. Ph.D. Thesis, University of the Basque Country, Spain.

Toro, C., Graña, M., Posada, J., Vaquero, J., Sanin, C., Szczerbicki, E. 2009. Domain modeling using engineering standards. *Lecture Notes in Computer Science* 5711: 95–102.

Toro, C., Sanín, C., Szczerbicki, E., Posada, J. 2008. Reflexive ontologies: Enhancing ontologies with self-contained queries. *Cybernetics and Systems: An International Journal* 39(2): 171–189.

Toro, C., Termenon, M., Posada, J., Oyarzun, J., Falcon, J. 2007. Ontology supported adaptive user interfaces for structural CAD design. In *Digital enterprise technology*, edited by P. Cunha and P. Maropoulos. Heidelberg, Germany: Springer.

Vaquero, J., Toro, C., Martin, J., Aregita, A. 2009. Semantic enhancement of the course curriculum design process. *Lecture Notes in Computer Science* 5711: 269–276.

Zorriassatine, F., Wykes, C., Parkin, R., Gindy, N. 2003. A survey of virtual prototyping techniques for mechanical product development. *Journal of Engineering Manufacture* 217(4): 513–530.