

Towards the Integration of Ontologies in the Context of MDA at CIM level

Héctor J. Ruidías, María Laura Caliusco, and María R. Galli

¹ CIDISI, Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Argentina

² INGAR, Consejo Nacional de Investigaciones Científicas y Técnicas

{hruidias,mcaliusc}@frsf.utn.edu.ar,mrgalli@santafe-conicet.gov.ar

<http://www.frsf.utn.edu.ar/cidisi/>

Abstract. In recent years, model-driven engineering has been popularized by the Model-Driven Architecture (MDA) initiative. Essentially, in MDA three types of viewpoints on models are distinguished: the Computation Independent Model (CIM), the Platform Independent Model (PIM) and the Platform Specific Model (PSM). Many research works of MDA are primarily focusing on the PIM and PSM, and transformations between each other. On the other hand, the Semantic Web has popularized another notion of model: ontologies. MDA may benefit from ontologies in formal model of domain semantics and automated reasoning. In this paper an approach for generating an ontology from a *Language Extended Lexicon* (LeL) with the aim of facilitating the transformation between CIM and PIM is presented. In addition, a software application, called *OntoLEL Tool*, that implements this approach is described.

Keywords: ontology,cim, mda

1 Introduction

The Model Driven Architecture (MDA) is a framework for software development defined by the Object Management Group (OMG). Key to MDA is the importance of models in the software development process. The MDA approach is composed of: the Computation Independent Model (CIM), the Platform Independent Model (PIM), and the Platform Specific Model (PSM). Because these models represent a different abstraction of the same system, a transformation mechanism is required to establish how to go from one level to another. Thus, transformations are a core element in the MDA [12].

The most abstract, but at the same time most domain specific model level is the CIM, which may be modeled using any kind of domain specific description (e.g., narrative use cases). This fuzzy method to describe a software system on a very high semantic abstraction level might be regarded as advantage, as it offers much flexibility [15]. Otherwise, it might be regarded as disadvantage, as it avoids efficient tool support to specify and transform the CIM to the next model level. Thus, a CIM is extended towards a PIM by hand by enriching it with operational model elements which in general involves multiple human

interpretations of the imprecisely described CIM [19]. Then, using the MDA often results in a semantic gap between the CIM and the PIM [9]. The semantic gap is the difference between how completely a model represents reality and reality itself [20].

Recently, some proposals have appeared related to processes business model and MDA with the aim of facilitating the transformation into CIM to PIM [11][17]. However, the CIM contains, beside a business model describing a company's rules of business, a domain model, describing the concepts of a domain and their relations, and the requirements.

The Semantic Web has popularized another notion of model: ontologies. An ontology gives an explicit definition of the shared conceptualization of certain domain [6]. The integration between ontologies and MDA was primary focused on how the MDA technologies could improved the ontology modeling [3] [5] and how ontology could define the semantics of MDA [8]. Another use of ontologies is for verification checks of mapping models in the course of metamodel composition [1]. At CIM level, different approaches have been defined that use an ontology for representing the structure of some requirements engineering artefacts [4][7]. Based on the Language Extended Lexicon(LEL) there is a proposal of an ontology building process for representing the domain knowledge[2].

In [18] a framework that integrates ontologies in the context of MDA with the aim of facilitating the transformation between CIM and PIM was presented. This framework proposes to define a set of ontologies for describing the structure of the requirements artefacts, such as LEL, scenarios and use cases; and derives from them a domain and application ontologies for obtaining a CIM model. However, the requirements artefact ontologies nor the process for deriving the domain ontology were presented. Then, the main contribution of this paper is an heuristic for deriving a domain ontology $Onto^{Dom}$ from a LEL and an ontology that models the semantics of a LEL calls $Onto^{LeL}$. In addition, a software application called OntoLEL Tool is presented. This Tool facilitates the construction of CIM models which today are mainly done manually.

This paper is organized as follow. Section 2 defines the main concepts around this paper. Section 3 presents $Onto^{LeL}$ and the heuristic for obtaining a domain ontology from it. Section 4 shows the OntoLEL Tool. Finally, Section 5 discusses the conclusions and future trends.

2 Background

2.1 Ontological Engineering

A domain ontology gives an explicit definition of the shared conceptualization of a certain domain [6]. From a pragmatic perspective, an ontology can be defined as a representational artifact based on four kinds of modeling components: concepts, roles, restrictions and individuals. Concept represents classes of objects. Roles describe binary relations among concepts; hence they also allow the description of properties of concepts. Restrictions are used to express properties of roles, i.e.

cardinality. Individuals represent instances of classes, i.e. objects. Additionally, it is possible to use axioms and rules to infer new information. Axioms are logical sentences always true that express the properties of model paradigm. Rules are logical sentences that express characteristics of the domain, i.e. business rules. Formally,

Definition 1. *An ontology is a 6-uple $O := \{\mathcal{C}, \mathcal{R}, \mathcal{H}, rel, \mathcal{A}, \mathcal{T}\}$ where:*

- *Two disjoint sets, \mathcal{C} (concepts) and \mathcal{R} (relations).*
- *A concept hierarchy, a directed relation $\mathcal{H} \subseteq \mathcal{C} \times \mathcal{C}$ which is called concept hierarchy or taxonomy. So, $\mathcal{H}(C1, C2)$ means $C1$ is a sub-concept of $C2$.*
- *A function $rel: \mathcal{R} \rightarrow \mathcal{C} \times \mathcal{C}$ that relates the concepts non taxonomically.*
- *A set of ontology axioms \mathcal{A} expressed in appropriate logical language.*
- *A set of ontology rules \mathcal{T} expressed in appropriate logical language.*

For implementing an ontology, the most useful language for reasoning is OWL-DL that corresponds to a Description Logic. OWL ontology consists of Classes and their Properties (relations and attributes). The Class definition specifies the conditions for individuals to be members of a Class. A Class can therefore be viewed as a set. The set membership conditions are usually expressed as restrictions on the Properties of a Class. It is possible to further constrain the range of a property with property restrictions which always apply to a specific property and they come in several types: "allValuesFrom", "someValuesFrom", "cardinality" and "has-Value". A property restriction can be treated as an anonymous OWL class, which means that it is possible to define another OWL class as a subclass of a property restriction. Property restrictions must only hold in the context of their subclasses, which may only be a small part of the entire property domain. A key feature of OWL and other description logics is that classification (and subsumption relationships) can be automatically computed by a reasoner which is a piece of software able to infer logical consequences from a set of asserted facts or axioms. Considering that the OWL language is the standard for implementing an ontology and this is not always enough to do some deduction, then it is needed to combine OWL with other representation formalism as rules. One of the integration approaches is the Semantic Web Rule Language (SWRL), which provides the ability to express Horn-like rules in terms of OWL concepts [14]. In order to extract information from OWL ontologies a query language is needed. The most powerful language is SQWRL, which is based on the SWRL rule language and uses SWRLs strong semantic foundation as its formal underpinning. It also contains novel set operators that can be used to perform closure operations to allow limited forms of negation as fail-true, counting, and aggregation [13].

2.2 Language Extended Lexicon

The Language Extended Lexicon is a technique that facilitates analysis of system requirements based on the specific terms of the Universe of Discourse (UofD)

expressed in the way they are used in it [15]. This technique focuses on understanding the vocabulary of the problem, leaving in the background the problem as such. This vocabulary consists of symbols, words or phrases taken from the UofD. Symbols must be obtained from the terms used by those involved in the requirements analysis and the organization employing the vocabulary, as this leads to facilitate understanding between users, analysts and developers. In LeL each symbol is characterized by its name, concept and impact. The symbol name can be a word or phrase used repeatedly in the domain and must be recognizable and familiar to the stakeholders. The denotation of the symbol is given by the notion, while the effects of the symbol are established from the impact. The

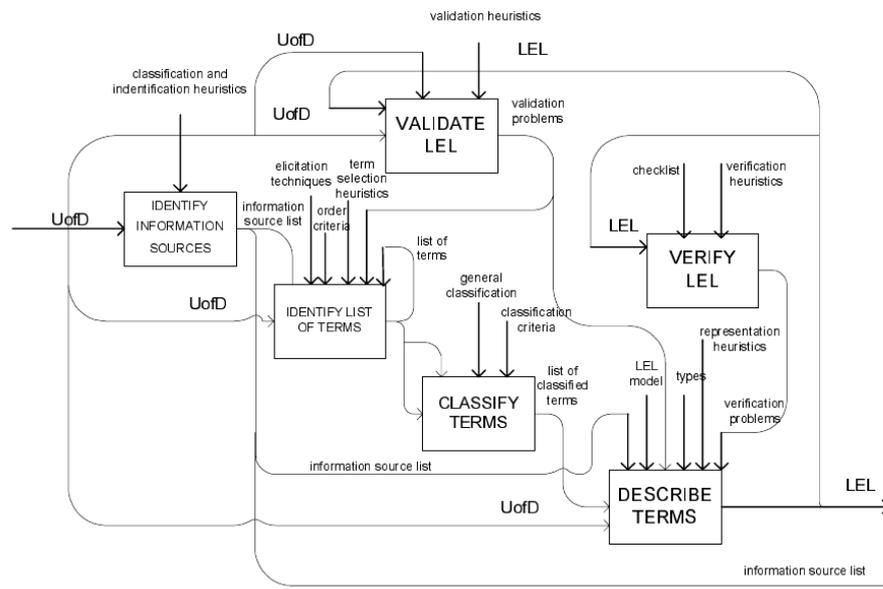


Fig. 1. A LeL Construction Process [10]

construction of the LEL must be oriented by the minimum vocabulary and the circularity principles. The circularity principle prescribes the maximization of the usage of LEL symbols when describing LEL entries while the minimal vocabulary principle prescribes the minimization of the usage of symbols exterior to the LEL when describing LEL entries. In order to define a LEL an iterative process has to be performed. This process, shown in Figure 1, is composed by six activities: 1) Identify Information Sources; 2) Identify List of Terms; 3) Classify Terms; 4) Describe Terms; 5) Verify LEL and 6) Validate LEL [10]. When a software development process required to define a lot of symbols, these two last activities are hard and time-consuming.

3 Derivation of a domain ontology from CIM

The purpose of this section is to present $Onto^{LeL}$, the ontology which describes the semantics of a LeL, and the process to generate a domain ontology called $Onto^{Dom}$, which in our approach constitute a CIM model.

3.1 $Onto^{LeL}$

In order to build the $Onto^{LeL}$ we followed the *Methontology* methodology and the ontology has been implemented in Protégé (<http://protege.stanford.edu>).

The main concepts of the $Onto^{LeL}$ (figure 2) are the ones of the model proposed by Leite[16]. Relations linking those concepts are: *hasNotion* that establish relations between instances of type *Symbol* and instances of type *Notion*, *hasBR* does the same between *Symbol* and *BehavioralResponse*, and the relationship between *Symbol* and *BehavioralResponse* and others instances of *Symbol* through relations *hasObject*, *hasSubject*, *hasVerb* and *hasState*. This ontology descriptions of LeL is expressed in DL notation in (1).

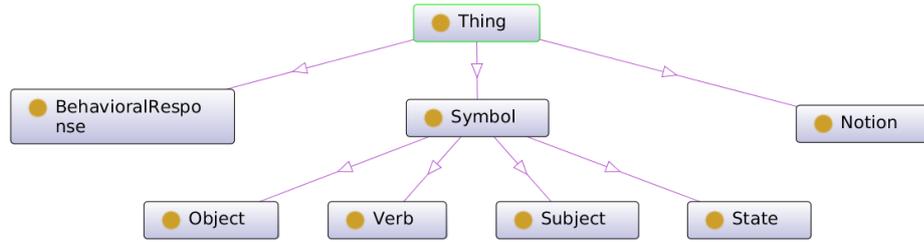


Fig. 2. The $Onto^{LeL}$ Ontology

$$\begin{aligned}
 Object &\equiv Symbol \sqcap \exists hasNotion.Notion \sqcap \exists hasBR.BehavioralResponse \\
 Subject &\equiv Symbol \sqcap \exists hasNotion.Notion \sqcap \exists hasBR.BehavioralResponse \\
 Verb &\equiv Symbol \sqcap \exists hasNotion.Notion \sqcap \exists hasBR.BehavioralResponse \\
 State &\equiv Symbol \sqcap \exists hasNotion.Notion \sqcap \exists hasBR.BehavioralResponse \\
 BehavioralResponse &\equiv \exists hasObject.Object \sqcap \exists hasSubject.Subject \\
 &\quad \sqcap \exists hasVerb.Verb \sqcap \exists hasState.State \\
 Notion &\equiv \exists hasObject.Object \sqcap \exists hasSubject.Subject \sqcap \exists hasVerb.Verb \\
 &\quad \sqcap \exists hasState.State \\
 Object \sqcap Subject \sqcap Verb \sqcap State &\equiv \perp
 \end{aligned}
 \tag{1}$$

The main advantages of using an ontology for defining a LeL is that the verification and validation processes can be done by using the reasoning capability of the ontology facilitating these activities. By using a reasoner, the consistency of the LeL can be checked. In addition, SWRL and SQWRL can be defined in order to validate the LeL. Furthermore, the $Onto^{LeL}$ ontology facilitates fulfillment of the principles of circularity and minimal vocabulary.

3.2 The process to obtain $Onto^{Dom}$

The process to obtain a $Onto^{Dom}$ directly from the instances of the $Onto^{LeL}$ can be resumed in three steps as is defined in the pseudocode of the algorithm 1. In the first step, for each instances of symbols $Object(L_o)$, $Subject(L_s)$ and $State(L_{st})$, a new class in $Onto^{Dom}$ is created and stored in $C(Onto^{Dom})$, taken the same name of the individual in $Onto^{LeL}$. The set of relationships ($R(Onto^{Dom})$) are obtained from instances of the symbol $Verb(L_v)$ in $Onto^{LeL}$, whose domain and range will be established further ahead. Furthermore, a mapping relationship τ_c is established between classes of $Onto^{Dom}$ and instances of $Onto^{LeL}$ as well as τ_r between relationships and instances of symbol $Verb$, these mapping functions have a two fold objective, one is vinculated to the process itself, to trace forward and backward between both ontologies, and the other to provides a very simple traceability mechanism which could be taken advantage in further stages of the global process.

The second step take all instances of *Notion* (L_{notion}) in order to check at first place for hierarchical relationship in case whether is the same kind of *Symbol*, and otherwise checking for another kind of relationship determined for the presence of a *hasVerb* property. These hierarchical relationships are established in $Onto^{Dom}$ between classes from a explicit relationship through *notions* of $Onto^{LeL}$ which relates *symbols of LeL* and have a *hasParent* relationship. In order to determinates that relationship, the τ_c are applies over instances of symbols either L_s , L_o or L_{st} filtered by *notion* (i_{notion}), obtaining in that way the corresponding classes which are related.

The last step complete the range and domain of each relations in $Onto^{Dom}$. To accomplish it the process check for each instance of *BehavioralResponse* in $Onto^{LeL}$ (L_{br}) searching for occurrences of instances of *Verb* meaning relationships between instances involves either of *Object*, *Subject* or *State*. In order to add a class into the *domain* of a relationship ($domain(\tau_r(r_{lel}))$), the class in question must have a behavioral response with a *hasVerb* property which matches with the relationship ($\{i \in L_{br} \mid \{r_{lel}\} \in i.hasVerb\}$). In the other hand, the range is defined by either *hasObject*, *hasSubject* or *hasState* of *behavioral response*, thus the *range* of relationship incorporates the classes involved.

4 The OntoLeL Tool

In order to test our approach, we have developed a software called *OntoLeL Tool*, which taking advantage of the *Protégé* API in the generation and manipulation

Algorithm 1 Domain Ontology generation

{Define I as Instances, C as Concepts and R as Relations of a Ontology}

Require: $Onto^{LeL}$

Ensure: $Onto^{Dom} \{I_{lel} \in Onto^{LeL}, C_{dom} \subset C(Onto^{Dom}), R_{dom} \subset R(Onto^{Dom}) \mid \tau_c :$

$I_{lel} \mapsto C_{dom} \wedge \tau_r : I_{lel} \mapsto R_{dom} \}$

$L_s \leftarrow \{I_s \mid I_s \in Subject^{Onto^{LeL}}\}$

$L_o \leftarrow \{I_o \mid I_o \in Object^{Onto^{LeL}}\}$

$L_v \leftarrow \{I_v \mid I_v \in Verb^{Onto^{LeL}}\}$

$L_{st} \leftarrow \{I_{st} \mid I_{st} \in State^{Onto^{LeL}}\}$

$L_{nt} \leftarrow \{I_{notion} \mid I_{notion} \in Notion^{Onto^{LeL}}\}$

$L_{br} \leftarrow \{I_{br} \mid I_{br} \in BehavioralResponse^{Onto^{LeL}}\}$

{STEP 1: Create Concepts and Relations of $Onto^{Dom}$ from $Onto^{LeL}$ }

for all $i \in \{L_o \sqcup L_s \sqcup L_{st}\}$ **do**

$C(Onto^{Dom}) \leftarrow i.hasName \subset C(Onto^{Dom})$

end for

for all $i \in L_v$ **do**

$R(Onto^{Dom}) \leftarrow i.hasName \subset R(Onto^{Dom})$

end for

{STEP 2: evaluate symbol's notions of the $Onto^{LeL}$ }

for all $i_{notion} \in L_{notion}$ **do**

{Check for hierarchical relationship}

if $i_{notion}.hasParent \in L_s$ **then**

$C(Onto^{Dom}) \leftarrow \tau_c(L_s.select(j \mid j.hasNotion \in \{i_{notion}\})) \subset$
 $\tau_c(i_{notion}.hasParent)$

end if

if $i_{notion}.hasParent \in L_o$ **then**

$C(Onto^{Dom}) \leftarrow \tau_c(L_o.select(j \mid j.hasNotion \in \{i_{notion}\})) \subset$
 $\tau_c(i_{notion}.hasParent)$

end if

if $i_{notion}.hasParent \in L_{st}$ **then**

$C(Onto^{Dom}) \leftarrow \tau_c(L_{st}.select(j \mid j.hasNotion \in \{i_{notion}\})) \subset$
 $\tau_c(i_{notion}.hasParent)$

end if

end for

{STEP 3: evaluate symbol's behavioral response of the $Onto^{LeL}$ }

for all $r_{lel} \in L_v$ **do**

for all $c_{lel} \in \{L_o \sqcup L_s \sqcup L_{st}\}$ **do**

if $\#(c_{lel}.hasBR \in \{i \in L_{br} \mid \{r_{lel}\} \in i.hasVerb\}) \neq \emptyset$ **then**

$domain(\tau_r(r_{lel})) \leftarrow domain(\tau_r(r_{lel})) \sqcup \tau_c(c_{lel})$

else

if $\#(\{i \in L_{br} \mid (\{c_{lel}\} \in i.hasObject \vee \{c_{lel}\} \in i.hasSubject \vee \{c_{lel}\} \in$
 $i.hasState) \wedge \{i\} \in c_{lel}.hasBR\}) \neq \emptyset$ **then**

$range(\tau_r(r_{lel})) \leftarrow range(\tau_r(r_{lel})) \sqcup \tau_c(c_{lel})$

end if

end if

end for

end for

of the ontologies involved. The main goal of this tool is to offer a comprehensive mechanism for building ontologies in very intuitive way for analysts, taking advantage of his knowledge about the analysis tools, in this case, is the LeL artefact. The main window of the software (figure 3) show us a clean interface

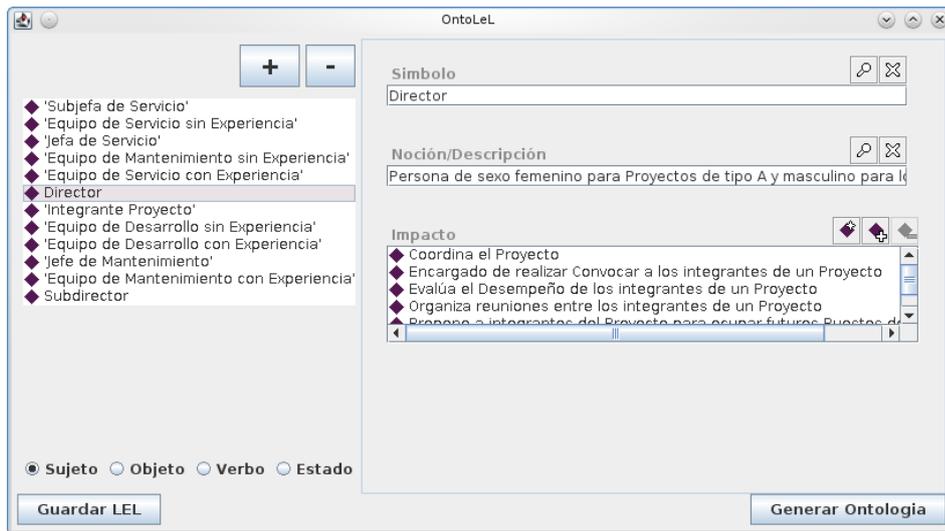


Fig. 3. Main Window on OntoLeL Tool

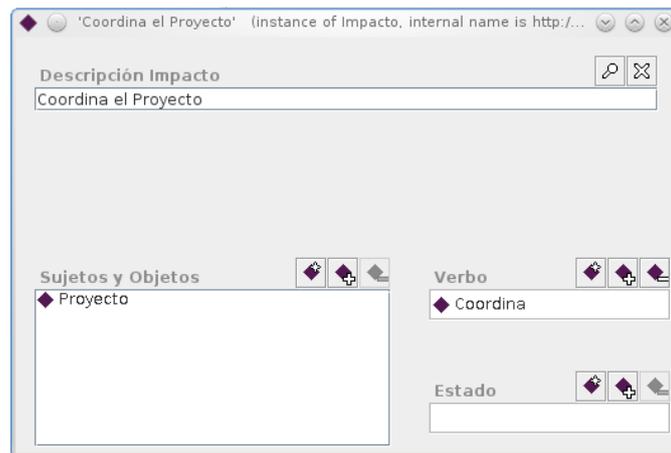


Fig. 4. Editing a Behavioral Response

where the analyst made a LeL from scratch. The panel at left presents in a intuitive way for either to add new symbols (plus label button), or to delete symbols of the list (minus label button). In the right panel we can see the *Description Name* of symbol, the *notions* and the *Behavioral Response* sections. Selecting in a items list the software deploy a window (figure 4) where we can link another symbols or add other on the fly. In the bottom of the main window we can see at the left a button to save the instances of $Onto^{LeL}$ wich has been created in the tool, and at the right the button to generate $Onto^{Dom}$. The $Onto^{Dom}$ is implemented in OWL 1.0 and it can be open with any tool that supports it.

5 Discussion and Future Work

In this paper the use of an ontology to describe the domain in term of LeL components was presented. Two goals has been covered with the proposed approach. The first one is to give an structure for the information of the LeL and also to offer a natural way to link symbols, such as Leite proposed it [16]. The second goal consists on the detection of inconsistencies in early stages of a software development project. In order to achieve it, we run consistence checking and clasification process over the ontology. This offer the possibility either of correct the ontology definitions or obtain a refined ontology over the base of their definitions.

In addition, the OntoLEL tool which implements our approach has been presented. Future work will be focused on improving the $Onto^{Dom}$ by including new ontologies for the rest of the requirements artefacts, and refining the heuristics of transformation according with thats inclusions. In addition, the framework [18] will be completed with transformation rules according to MDA approach tending to obtain a first conceptual model for the design stage of an information system reducing the semantics gap between CIM and PIM.

References

1. Arnarsdóttir, K., Berre, A.J., Hahn, A., Missikoff, M., Taglino, F.: Semantic mapping: ontology-based vs. model-based approach alternative or complementary approaches? In: Missikoff, M., 0001, A.D.N., D'Antonio, F. (eds.) EMOI-INTEROP. CEUR Workshop Proceedings, vol. 200. CEUR-WS.org (2006)
2. Breitman, K.K., Leite, J.C.S.d.P.: Ontology as a requirements engineering product. In: RE '03: Proceedings of the 11th IEEE International Conference on Requirements Engineering. p. 309319. IEEE Computer Society, Washington, DC, USA (2003)
3. Cranefield, S.: UML and the semantic web. In: Cruz, I.F., Decker, S., Euzenat, J., McGuinness, D.L. (eds.) The Emerging Semantic Web. Frontiers in Artificial Intelligence and Applications, vol. 75. IOS press (2001)
4. Decker, B., Ras, E., Rech, J., Klein, B., Hoecht, C.: Self-organized reuse of software engineering knowledge supported by semantic wikis. In: In Workshop on Semantic Web Enabled Software Engineering (SWESE) (2005)

5. Gasevic, D., Djuric, D., Devedzic, V.: Model Driven Engineering and Ontology Development. Springer Publishing Company, Incorporated, 2nd edn. (2009)
6. Gómez-Pérez, A., Fernández-López, M., Corcho, O.: Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer (2004)
7. Groza, T., Schutz, A., Handschuh, S.: SALT: a semantic approach for generating document representations. In: Proceedings of the 2007 ACM symposium on Document engineering. p. 171173. DocEng '07, ACM, New York, NY, USA (2007)
8. Guizzardi, G.: Ontological foundations for structural conceptual models. Ph.D. thesis, Centre for Telematics and Information Technology University of Twente, Enschede (2005)
9. Jablonski, S., Lay, R., Melier, C., Faerber, M., Volz, B., Dornstauder, S., Gotz, M., Muller, S.: Integrated process and data management for healthcare applications. International Journal of Healthcare Information Systems and Informatics 2(4), 1–21 (2007)
10. Kaplan, G.N., Hadad, G.D.S., Doorn, J.H., do Prado Leite, J.C.S.: Inspección del léxico extendido del lenguaje. In: Workshop on Requirements Engineering (WER). pp. 70–91 (2000)
11. Kardo, M., Drozdov, M.: Analytical method of CIM to PIM transformation in model driven architecture (MDA). Journal of Information and Organizational Sciences 34(1), 89–99 (2010)
12. Mellor, S.J.: Mda Distilled: Principles of Model-Driven Architecture. Addison-Wesley Professional (2004)
13. O'Connor, M., Das, A.: SQWRL: a query language for OWL. In: OWL: Experiences and Directions (OWLED), Fifth International Workshop. Chantilly, VA (2009)
14. Oconnor, M., Knublauch, H., Tu, S., Musen, M.: Writing rules for the semantic web using SWRL and jess. In: 8th International Protege Conference, Protege with Rules Workshop (2005)
15. Pohl, K.: Requirements Engineering: Fundamentals, Principles, and Techniques. Springer (2010)
16. do Prado Leite, J.C.S., Doorn, J., Hadad, G., Doorn, J.H., Kaplan, G.: A scenario construction process. REQUIREMENTS ENGINEERING 5(1), 3861 (2000)
17. Rodríguez, A., Fernández-Medina, E., Piattini, M.: CIM to PIM transformation: A reality. In: Xu, L., Tjoa, A., Chaudhry, S. (eds.) Research and Practical Issues of Enterprise Information Systems II, IFIP International Federation for Information Processing, vol. 255, pp. 1239–1249. Springer Boston (2008)
18. Ruidías, H.J., Caliusco, M.L., Galli, M.R.: Hacia un framework basado en MDA y ontologías para el desarrollo de software. In: Seminario Argentina-Brasil de Tecnología de la Información y las Comunicaciones. Rosario, Argentina (2011)
19. Yue, T., Briand, L.C., Labiche, Y.: A systematic review of transformation approaches between user requirements and analysis models. Requir. Eng. 16(2), 7599 (2011)
20. Zhuang, Q., Feng, J., Bao, H.: Measuring semantic gap: An information quantity perspective. In: 2007 5th IEEE International Conference on Industrial Informatics. vol. 2, pp. 669–674 (2007)