Generation of an OWL Ontology from a Knowledge Domain Extended Lexicon

Karla Olmos-Sánchez
Universidad Autónoma de Ciudad Juárez
Av. Del Charro 450 Nte.
Cd. Juárez Chih. Mex.
(52) 656 6884841
kolmos@uacj.mx

Jorge Rodas-Osollo
Universidad Autónoma de Ciudad Juárez
Av. Del Charro 450 Nte.
Cd. Juárez Chih. Mex.
(52) 656 6884841
jorge.rodas@uacj.mx

Yanet Garay
Universidad Autónoma de Ciudad Juárez
Av. Del Charro 450 Nte.
Cd. Juárez Chih. Mex.
(52) 656 6884841
yanet.garay@gmail.com

Sonia Herrera
Universidad Autónoma de Ciudad Juárez
Av. Del Charro 450 Nte.
Cd. Juárez Chih. Mex.
(52) 656 6884841
sonia.magdiel.269@gmail.com

ABSTRACT
Informally Structured Domains (ISD) are characterized by informal and unstructured information that depends on the context for interpretation; thus, the most of the concepts and their relationships are defined by consensus and domain specialists use large amounts of tacit knowledge in order to solve everyday situations. These characteristics cause that modeling this kind of domains becomes a challenging and time-consuming task in which the representations do not reflects correctly the reality. This paper proposes a new approach to the process of understanding and modeling an ISD, by using a process for generating an OWL ontology from a lexicon named KDEL with the aim of supporting a better understanding of the application domain, hence facilitates the development of products or solutions in ISD.

Keywords
Informally Structured Domains; Ontologies Building Process; Knowledge Domain Extended Lexical

1. INTRODUCTION
The importance of knowledge domain in order to elicit requirements of a solution or product that fulfill the needs and expectative of clients and users is widely accepted among the research community [2][12]; especially when the solution-solvers or product developers are not immerse in the application domain. Recently, the use of ontologies as a means to define and make explicit this knowledge has become seen as a good option [14][6]. Domain ontologies can be used as a way of facilitating the understanding among stakeholders, detecting of missing and erroneous information and describing the domain in the way of domain specialists thinking, hence avoid ambiguous, insufficient and incomplete requirements. In this work, the term *domain specialists* refers to all people involved in the application domain, which could have partial and different knowledge of it depending on their role and experience.

In particular, the OWL Web Ontology Language has been designed for use by computer systems instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content by providing additional vocabulary to XML, RDF and RDF Scheme, along with a formal semantics. OWL allows us to describe the semantics of knowledge in a machine-accessible way, therefore it has promoted the development of multiple and varied software applications [1].

Nevertheless, ontology development is a complex and time-consuming activity that seems to be an art rather than a formal method. Besides, not all domains are equal, there are domains where not all concepts and their relationships can be formally defined, the solutions of most of their problems are situated and diverse, not susceptible to be described by an algorithm, and where domain specialists use large amounts of tacit knowledge in order to solve problems. In this kind of domains, named *Informally Structured Domains*, providing certain structure that synthesizes the knowledge of domain specialists and make it explicit is fundamental in order to develop a correct and appropriate solution or product [9]. This structure can be proportionate by an OWL ontology.

The objective of this paper is to present a new approach to generate an OWL ontology from the Knowledge Domain Extended Lexicon (KDEL), in order to facilitate the understanding, development and validation of an ISD. There is a similar approach proposed by [3], however our work is designed keeping in mind the ISD characteristics. Thus, our motivations is to provide a tool that minimize the time to structure and visualize the domain knowledge, with the further aim of facilitating to discover relationships that could have hidden to domain specialist awareness.

The rest of this paper is structured as follows: Section 2 provides a detailed explication of KDEL, section 3 describes the OWL ontology language, the OWL building process from KDEL is introduced in section 4, section 5 reports the results of the application of the method in ISD real cases, finally section 6 concludes our work with future directions.

2. KDEL
The term *Universe of Discourse (UofD)* generally refers to the collection of objects being discussed in a specific domain. It is evident that there is a correlation between the domain knowledge and the terms used daily by domain specialists. Thus, in order to assist to solution-solver or product developer in understanding the terms of the domain specialists, several authors [8][13] propose the use of a glossary of common terms with the additional aim of facilitating communication and understanding of all involved in a project. Despite that natural language is ambiguous and depends on the context for interpretation, it is the only notation that is commonly readable and understandable by the domain specialists and its use encouraging them to participate dynamically in the first steps of any project [7].

One of these proposals is the Language Extended Lexical (LEL) [11], which is a set of terms related to the application domain with the aim of understanding the language of the problem without worrying about understanding the problem. In order to give an initial structure to the knowledge domain, each term in the UofD is

classified as *object*, *subject*, *verb* or *state* and is described by a *notion* (denotation) and a *behavioral response* (connotation).

2.1 KDEL Process Building

In order to deal with the challenges of ISD, the Knowledge of Domain on an Extended Lexicon (KDEL) evolves LEL by modifying two aspects of it. The first one is that, besides the classification of object, subject, verb and state, it incorporates *definitions* and *NF-Requirements*. The rationality behind this is that in the early stage of any development software project, the domain specialists do not have a clear idea of what they want. In ISD, even they do not have a well-defined structure of the application domain knowledge and a great quantity of it is tacit or implicit. Thus, the domain specialists interleave in their discourse needs, desires, domain properties, and current and future processes. To give a preliminary order to this information, KDEL characterizes the application domain in terms, which can be concepts, definitions and Non-Functional (NF) requirements; as it is explained below:

- **Concepts**. They are equivalents to the terms in LEL and are described by a notion (denotation) and a behavioral response (connotation). KDEL classify the concepts as *objects*, *subjects* or *verbs*. Unlike LEL, *state* is not considered as a concept because we consider that it is inherently attached to subjects or objects.
- **Definitions**. They are statements that assign a precise or consensual meaning to terms used in the applications domain, but that cannot be considered as concepts; thus they cannot have a behavioral response. Definitions are necessary in order to understand the context of the application domain.
- **NF Requirements**. They refer to concerns not related to the functionality of the software, such as usability, flexibility, performance, interoperability and security [5]. One of the objectives of the KDEL is to capture the NF-Requirements introduced in the early discourses of domain specialists; however, in subsequent stages of the process, solution-solvers of product developers can include more of them.

The second aspect that KDEL modifies is the internal structure of

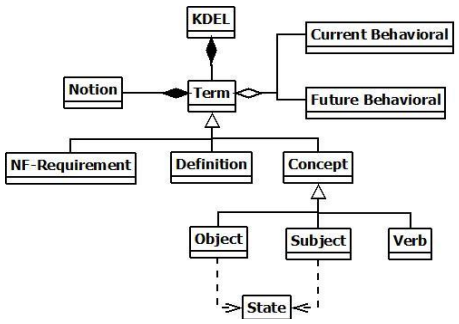

Figure 1. Conceptual scheme of KDEL in UML

terms. The application domain will be affected after a solution or product was deployed in it; hence, the set of terms, as well as its denotation and connotation, will not be the same. To handle this issue, a *future behavioral response* is added to the structure of them, which is not mandatory; it is only added if it is evident in the early stages of the project. It allows the requirements engineers to gain more domain knowledge and explore new possibilities of solution by understanding the problem and the structure of the solution. Figure 1 depicts a scheme of KDEL in an UML class diagram.

Handling synonymous is a special issue of any representation of language. Two or more terms are synonymous if they share the meaning of a concept. In KDEL, when two or more terms are synonymous they share the same structure and the two terms are separated by a diagonal slash.

Based on the rules to describe terms proposed in [7], a set of suggestions for description of terms have been proposed for KDEL. Table 1 gives the rules of description for *objects*, Table 2 for *subjects*, Table 3 for *verbs*, Table 4 for *definitions* and Table 5 for *NF-requirements*.

Table 1. Rules of description of objects

Object	
Notion	Define the object and its relationships with other objects or subjects
Current Behavioral	Describe the actions that are done with the object in the current time
Future Behavioral	Describe the actions that are done with the object once the solution or product were deployment
States	Describe all possible states of the object and the event that triggers it

Table 2. Rules of description of subject

Subject	
Notion	Define the subject and its relationships with other objects or subjects
Current Behavioral	Describe the actions that are done by the subject in the current time
Future Behavioral	Describe the actions that are done by the subject once the solution or product were deployment
States	Describe all possible states of the object and the event that triggers it

Table 3. Rules of description of subject

Verb	
Notion	Describe who performs the action represented by the verb and when the it occurs
Current Behavioral	Describe in detail the action in the current time
Future Behavioral	Describe in detail the action in the future time

Table 4. Rules of description of definitions

Definition	
Notion	Describes the meaning of the term in the domain

Table 5. Rules of description of NF-requirements

NF-requirements	
Notion	Describes the NF-Requirement

Goals	Describe the goals to be achieved by the NF-requirement

Besides the rules describes allow, in order to build KDEL the following task are also necessary:

- Apply techniques of discourse analysis to identify syntactic constructions that could hide tacit knowledge.
- Record, for each term in KDEL, questions or comments that will be consulted to domain specialists.

Table 6 depicts the representation of the term *evaluator* in a domain of cognitive diagnosis of multiple sclerosis patients.

Table 6. Structure of the term *evaluator* in KDEL

Term: Evaluator	
Classification	Subject
Notion	The evaluator is a certified neuro psychologist in the cognitive diagnostic and rehabilitation of multiple sclerosis patients.
Current Behavioral	- The evaluator applies the neuro psychological battery of test for cognitive diagnostic to multiple sclerosis patients. - The evaluator proposes the rehabilitations cognitive training based on the result of the evaluation. - The evaluator indicates to patients when the next test will be applied.
Future Behavioral	The concept of evaluator disappears in the future system; their functionalities will be carried out by the software system.
States	*No apply*
Questions	- Must the evaluator be a neuro psychologist? - How the evaluator determines that the patient has multiple sclerosis? - How he or she makes the evaluation? - How the evaluator defines the date of the next test?

2.2 Limitations of KDEL

KDEL facilitates to solution-solver or product developers to be familiar with the language of domain specialist; hence it minimizes the difficulties involved by describing the domain using a semi-formal or formal method. However, there are some drawbacks such as the redundancy in the description of terms, which causes a validation time-rise. In addition, it must be considered the hard and time-consuming activity of using KDEL to build a graphical conceptual model, which will be used by solution-solvers or product developers in two different ways: to facilitate the validation of the structure of domain and to let bring to light relationships that were hide to domain specialists.

3. OWL ONTOLOGIES

An ontology is an explicit formal specification of how to represent the entities and relationships that exist in a domain. They describe the properties of a domain and reasoning about it [4]. Ontologies have also been used to capture and synthesize knowledge from diverse domain specialists, especially when their knowledge depends on their own interests and points of view [6]. Thus, several authors have raised the use of ontologies to formalize the application domain.

3.1 OWL Structure

OWL is a widely used proposal of formal languages for ontologies [1] , which is defined using the syntax of RDF/XML. The elements of an OWL ontology concern *classes*, *properties*, *instances of classes*, and *relationships* between these instances. This section presents the essential components of this language in order to introduce those elements.

1) Classes are concrete representations of concepts; OWL classes are interpreted as a set of individual objects with similar features. The RDF/XML syntax to represent OWL classes is:

```
<owl:Class rdf:ID="Class_X"/>
<owl:Class rdf:ID="Class_Y"/>
```

The taxonomic constructor for classes is `rdfs:subClassOf`. It relates a more specific class to a more general class. If *X* is a subclass of *Y*, then every instance of *X* is also an instance of *Y*.

```
<owl:Class rdf:ID="Class_X">
     <rdfs:subClassOf
     rdf:resource="#Class_Y"/>
      ...
</owl:Class>
```

2) *Individuals* represent objects in the domain of discourse; they can be referred to as being instances of classes. The RDF/XML syntax to represent OWL individuals where *Individual_X* is a member of *Class X* is:

```
<owl:Thing rdf:ID="Class_X" />
 <owl:Thing rdf:about="#Class_X">
   <rdf:type rdf:resource="#Individual_X"/>
  </owl:Thing>
```

3) Properties are also known as roles in description logic or relations in UML and other object oriented notations. In brief, they represent relationships. There are a number of ways to restrict the relation defined by a property: the domain and range can be specified and the property can be defined to be a specialization

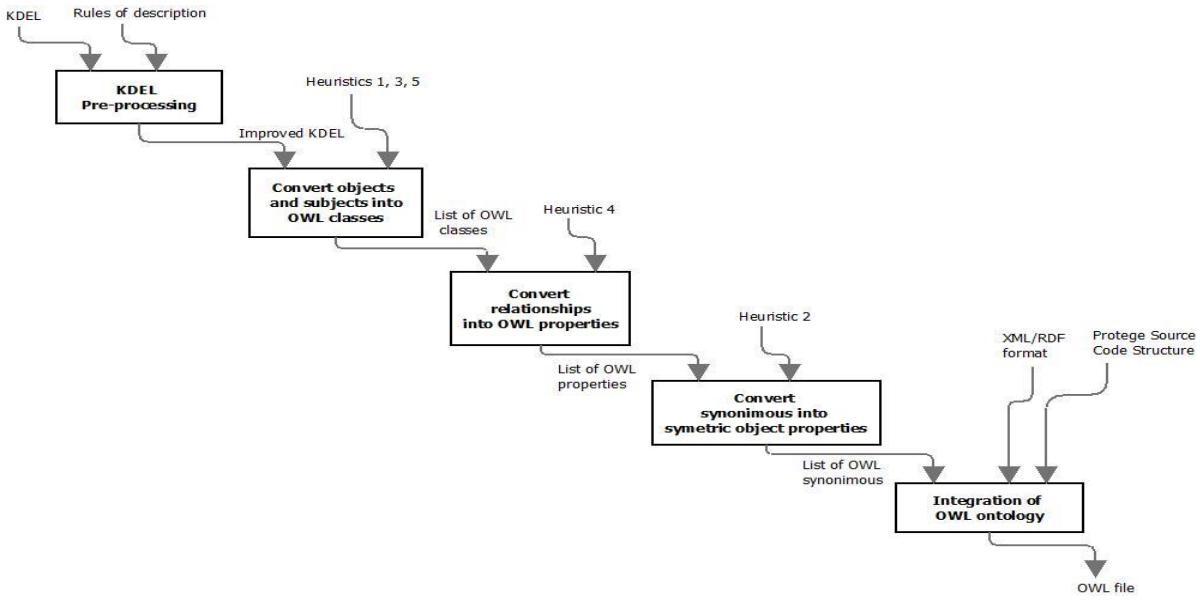

Figure 2. OWL ontology building process from KDEL

(*subproperty*) of an existing property [36]. There are two main types of properties: *object properties* and *datatype properties*. *Object properties* are relationships between two individuals. In the next example *X has a link with Y*.

```
<owl:ObjectProperty rdf:ID="    hasALinkWith">
  <rdfs:domain rdf:resource="#X"/>
  <rdfs:range rdf:resource="#Y"/>
</owl:ObjectProperty>
```

Datatype properties describe relationships between an individual and data values. In the next example a *Person_X has age* and it is a *positive integer*.

```
<owl:DatatypeProperty rdf:ID="hasAge">
  <rdfs:domain rdf:resource="#Person_X" />
   <rdfs:range
rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

4. FROM KDEL TO AN OWL ONTOLOGY

As was mentioned above, KDEL is a glossary of interrelated terms. They are classified as object, subject or verb and have a description. In particular, objects and subjects represent an entity in the domain and have relationships with other terms, which are described in the current behavioral. Thus, there is a correlation of them with OWL *classes*. Likewise, the relationships between concepts are equivalent with OWL *properties*.

A set of heuristics, rules or methods that helps to solve problems faster than it would if all the computing were done, have been proposed in order to facilitate the conversion of KDEL to OWL, which are listed follows:

1) Each *subject* or *object* of KDEL becomes an OWL *class*.
2) For each KDEL construction with the structure *X is synonym of Y/Z* the following OWL *properties* are created:

   X *Is_Synonym* of *Y* and

   X *Is_Synonym* of *Z*.
3) Descriptions of the terms (notion, current and future behavior) become OWL *comments.*
4) Relationships in KDEL with the syntactic structure *Term + verb phrase + term* are turned into OWL *properties*.
5) *Definitions* in KDEL are converted into OWL *classes*.
6) *NF-Requirements* are currently not part of the ontology; their handling is considered to be future work.

4.1 OWL Ontology Building Process

In order to build an OWL ontology from a KDEL lexicon, the following process is proposed. The process works together with the heuristics proposed above.

1) Perform a pre-processing of KDEL based in the *rules of description* (Section 2.1).
2) Convert *KDEL terms* into *OWL classes* (Heuristics 1, 3, 5).
3) Convert *KDEL relationships* into OWL properties (Heuristic 4).
4) Convert *KDEL synonymous* into *OWL classes* by creating the *OWL property* between them: *Is_Synonym_of* (Heuristic 2).
5) Create an OWL file following the format of RDF/XML by integrating classes, properties and individuals, as identified in the previous steps.
6) Name the file created in the last step with the name selected for the OWL ontology including the file extension for OWL.

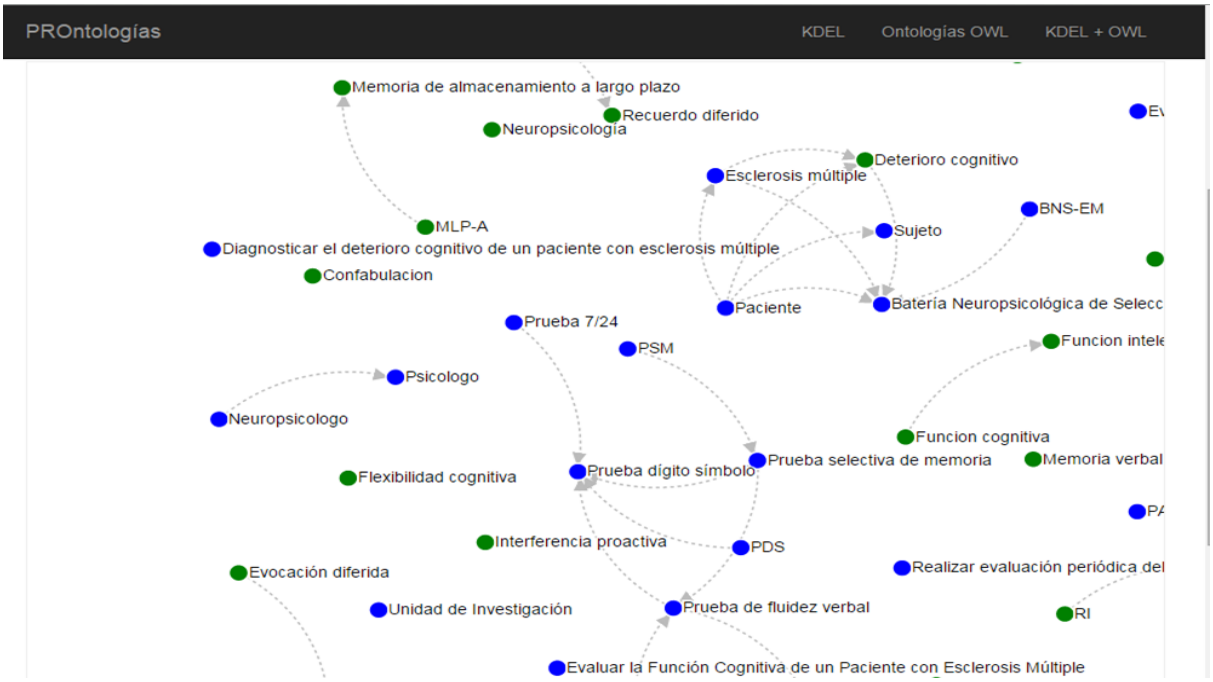

Figure 3. Screen for the Graphical Representation of an OWL ontology

Figure 2 depicts the OWL building process in a SADT diagram.

4.2 Software Tool

The solution-solvers or product developers must learn the domain terms in a short period of time in order to reduce the symmetry of ignorance, improve the cognitive dialogue and find, with the domain specialists, the set of solution or product requirements. However, in Informally Structured Domains, the universe of discourse is frequently too large and specialized. In addition, the process of validation of the terms is generally a boring and stressful task. Thus, a software system has been developed to support the handle of KDEL with the aims to facilitate the building, maintenance and validation of the KDEL. The system is designed to be used by the design team and it is able to manage several projects. The terms of KDEL are recorded in a relational database, following the structure showed in Figure 1.

This database is the input of another software system that executes the OWL ontology building process described in the previous section. The software also has the functionality of represent in a graphical format the RDF/XML file. Thus, it allows the visualization of KDEL with the aim of facilitating the validation process by domain specialists. In addition, if domain specialists realize that the description of a term must be improved, the software allows this change and automatically reconstructs the KDEL and the OWL file.

Figure 3 depicts a screen with the graphical representation of KDEL of the domain of cognitive diagnosis for multiple sclerosis patients. This project was developed for a Mexican real organization; thus the KDEL was developed in Spanish. However, it is not our intention to give a detail description of the lexical, but demonstrate the utility of the software tool in order to facilitate the validation of the domain terms and their relationships. The visualization also allows domain specialists to discover relationships that were hidden to them. Therefore, the software system is also appropriate for discovery knowledge issues.

5. APPLICATION IN ISD REAL CASES

Our proposal has been applied to generate OWL ontologies as a part of the process for diverse solution in ISD real cases, which are listed below:

- Software Development of a Cognitive Rehabilitation System for Sclerosis Multiple Patients [10].
- Case-based Reasoning System to Support Heating Ventilation and Air Conditioning (HVAC) Design Decisions.
- Method to develop Bayesian Networks for evaluation in Intelligent Tutoring System for complex domains.
- Analysis of the requirements elicitation process of a HVAC company.

The generation of the ontology in each project allowed the extraction of relevant information from KDEL, which led to a view of the information in a synthesized way. This process also facilitated the correction of the following errors committed in KDEL: 1) repeated information, 2) ideas vaguely described, 3) ideas mixed or unfinished and 4) typing errors. In summary, the OWL ontology building process improves KDEL, which facilitates the validation of it. In addition, the automatic generation of the OWL ontology significantly shortens the time and effort required to generate a graphical representation of the domain and contributes to the understanding of the ISD.

6. CONCLUSIONS AND FUTURE WORK

The application of KDEL and the generation of an OWL ontology from it in order to provide certain structure and facilitate the understanding of the domain to the solution-solvers or product developers in ISD real cases showed that the process minimizes the time of understanding the domain. It also minimizes the time it would take the domain specialists to validate the domain structure due to the graphical domain visualization. Finally, the graphical domain visualization also facilitates the discovery or relationships that were hidden to the domain specialist; allowing the detection and correction of errors in KDEL.

As future work it is necessary to apply the process in others ISD real cases in order to verify their effectiveness and improved it, if necessary.

7. ACKNOWLEDGMENTS

Our thanks to the Unity for Health Research UIS for its acronym in Spanish (Unidad de Investigación en Salud) for allowing us to work with them in the development of the rehabilitation system for multiple sclerosis patients.

8. REFERENCES

1. Dean Allemang and James Hendler. 2008. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann Publishers Inc.

2. Dines Bjørner. 2010. Rôle of domain engineering in software development : Why current requirements engineering is flawed. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2–34. http://doi.org/10.1007/978-3-642-11486-1_2

3. Karen Koogan Breitman and Julio Cesar Sampaio do Prado Leite. 2003. Ontology as a requirements engineering product. In *Proceedings of the 11th IEEE International Requirements Engineering Conference, 2003*, 309–319.

4. Verónica Castañeda, Luciana Ballejos, Ma. Laura Caliusco, and Ma. Rosa Galli. 2010. The Use of Ontologies in Requirements Engineering. *Global Journal of Research In Engineering* 10, 6.

5. Luiz Marcio Cysneiros and Julio Cesar Sampaio do Prado Leite. 2004. Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering* 30, 5: 328–350.

6. Diego Dermeval, Jassyka Vilela, Ig Ibert Bittencourt, et al. 2015. Applications of ontologies in requirements engineering: a systematic review of the literature. *Requirements Engineering*, 1–33. http://doi.org/10.1007/s00766-015-0222-6

7. Shivani Goel. 2012. Transformation from LEL to UML. *International Journal of Computer Applications* 48, 12: 975–888.

8. Julio Cesar Sampario do Leite, Ana P M Franco, and others. 1993. A strategy for conceptual model acquisition. In *Proceedings of IEEE International Symposium on Requirements Engineering 1993*, 243–246.

9. Karla Olmos and Jorge Rodas. 2013. Requirements engineering process model for informal structural domains. *International Journal of Computer and Communication Engineering* 2, 1: 75–77.

10. Karla Olmos and Jorge Rodas. 2014. KMoS-RE Knowledge Management on a Strategy to Requirements Engineering. *Special Issue on Requirements Engineering in Software Product Line Engineering, Requirements Engineering Journal* 19, 4: 421–440.

11. Julio Cesar Sampaio do Prado Leite, Jorge Horacio Doorn, Graciela D S Hadad, and Gladys N Kaplan. 2005. Scenario inspections. *Requirements Engineering* 10, 1: 1–21.

12. Pedro O Rossel, María Cecilia Bastarrica, Nancy Hitschfeld-Kahler, Violeta Díaz, and Mario Medina. 2014. Domain modeling as a basis for building a meshing tool software product line. *ADVANCES IN ENGINEERING SOFTWARE* 70: 77–89. http://doi.org/10.1016/j.advengsoft.2014.01.011

13. Matt Selway, Wolfgang Mayer, and Markus Stumptner. 2014. Semantic interpretation of requirements through cognitive grammar and configuration. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8862: 496–510. http://doi.org/10.1007/978-3-319-13560-1

14. Katja Siegemund, Edward J Thomas, Yuting Zhao, Jeff Pan, and Uwe Assmann. 2011. Towards ontology-driven requirements engineering. In *Proceedings of the Workshop Semantic Web Enabled Software Engineering at 10th International Semantic Web Conference (ISWC), Bonn*.