Ontology as a Requirements Engineering Product

Karin Koogan Breitman, Julio Cesar Sampaio do Prado Leite*
Pontifícia Universidade Católica do Rio de Janeiro
karin@inf.puc-rio.br www.inf.puc-rio.br/~julio

Abstract

The "Semantic Web" community poses a new non-functional requirement for web applications. In order to secure interoperability and allow autonomous agent interaction, software for the web will be required to provide machine processable ontologies. We understand that the responsibility, not only for making explicit this requirement, but also to implement the ontology, belongs to requirements engineers. As such, we see the ontology of a web application as a sub-product of the requirements engineering activity. In this tutorial we survey the basic principles behind ontologies as they are being implemented and used by the Semantic Web Community today. Those include ontology languages, tools and construction methods. We focus on a process for ontology construction centered on the concept of application languages. This concept is rooted on a representation scheme called the language extended lexicon (LEL). We demonstrate our approach with examples in which we implement machine processable ontologies in the DAML+OIL language. We finalize with a discussion of today's research issues in ontology engineering, including ontology evolution, , integration and validation.

1. Introduction

The development of the World Wide Web made the Internet accessible to millions by making it easy for anyone to publish and access documents on the Internet. However, the explosive growth of the number of documents published has led to the problem of information overload [Fensel03]. Researchers from industry and academia are now exploring the possibility of creating a "Semantic Web," in which meaning is made explicit, allowing machines to process and integrate Web resources intelligently. Beyond enabling quick and accurate web search, the use of ontologies may also allow the development of intelligent internet agents and facilitate communication between a multitude of heterogeneous web-accessible devices. Unfortunately the majority of the information available is a format understandable to humans alone, making the automation of search and retrieval processes very hard [Berners-Lee02]. Ontologies will provide the necessary meaning to web content therefore enabling software agents to understand and retrieve information in relevant contexts.

2. Ontology Definition

The word ontology comes from the Greek *ontos* (being) + *logos* (word). It has been introduced in philosophy in the 19th century, by German philosophers, to distinguish the study of being as such from the study of various kinds of beings in the natural sciences. As a philosophical discipline, ontology building is concerned with providing category systems that account for a certain vision of the world [Guarino98].

In computer science, ontologies were developed in Artificial Intelligence to facilitate knowledge sharing and reuse [Fensel01]. Today they are becoming widespread in areas such as intelligent information integration, cooperative information systems, agent based software engineering and electronic commerce. One of the most cited ontology definition is provided by Gruber [Gruber93]: "An ontology is a formal, explicit specification of a shared conceptualization." Where conceptualisation stands for an abstract model, explicit means that the elements are clearly defined and, lastly, formal means that the ontology should be machine processable [Fensel01]. There is, however, no universal definition for ontologies [Gruninger02]. One of the reasons is the great spectrum of possible uses for them. Gruninger relates, among others, the possibility of using ontologies for the communication between humans and implemented computational systems, for computational inference and for the reuse and organization of knowledge [Gruninger02]. The formality degree of an ontology is also an issue [Fensel03].

For the purposes of the tutorial, we adopt the ontology structure $\mathcal{O}$ proposed by Maedche [Maedche02].

---

* On leave at the University of Toronto, Department of Computer Science

** This research was supported in part by CNPq under contract ESSMA- 552068/2002-0, and by CAPES.

According to the author, an ontology can be described by a 5-tuple consisting of the core elements of an ontology, i.e., concepts, relations, hierarchy, a function that relates concepts non-taxonomically and a set of axioms. The elements are defined as follows:

$O := \{C, \mathcal{R}, \mathcal{H}^C, rel, \mathcal{A}^O\}$ consisting of :

Two disjoint sets, C (concepts) and $\mathcal{R}$ (relations)

A **concept hierarchy**, $\mathcal{H}^C$: $\mathcal{H}^C$ is a directed relation $\mathcal{H}^C \subseteq C \times C$ which is called concept hierarchy or taxonomy. $\mathcal{H}^C(C1,C2)$ means C_1 is a subconcept of C_2

A **function** $rel: \mathcal{R} \rightarrow C \times C$ that relates the concepts non taxonomically

A set of ontology **axioms** $\mathcal{A}^O$, expressed in appropriate logical language.

Figure 1 exemplifies our use of the ontology structure. We borrow from the Meeting Scheduler ontology. We choose three concepts, meeting details (x), meeting date(y) and agenda(z), one relation, is_registered (w) that holds between concepts meeting date and agenda. Concept meeting date holds a subsumption relationship to the meeting details concept. According to the structure we have something like, $C := \{x, y, z\}$ and $\mathcal{R} := \{w\}$. The following hierarchical relation, $\mathcal{H}^C(y,x)$ and the non taxonomic relation, w(y,z).

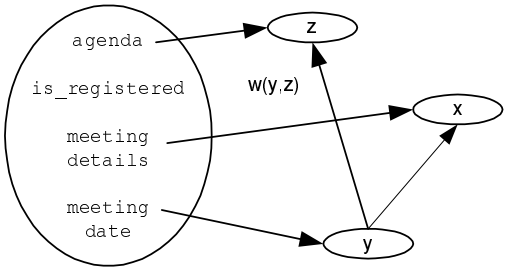

Figure 1 – Example of ontological elements and their relations using the ontology structure O proposed by [Maedche02].

The reason behind our choice is that the representation proposed by Maedche can be mapped to most existing ontology representation languages. In the next section we survey some of the ontology languages.

3. Ontology Languages

Tim Berners Lee proposed a layered language model for the WWW at the XML 2000 conference, depicted by Figure 2. At the bottom layer there is HTML and XML, that enabled the enormous growth of the web. Because of its simplicity, HTML hampered the development of more sophisticated applications [Fensel03]. As a result another language was defined, XML. On the second layer there is the Resource Framework Description (RDF) that provides a formal data model and syntax to encode machine processable metadata. The idea behind RDF is to provide interoperability between applications that exchange machine understandable information on the web[Hjelm01]. It provides a set of primitives for modeling simple ontologies in RDF schema, including subsumption of relationships for classes and properties. RDF however, was criticized as an ontology language because it lacked expressiveness [Heflin01]. In the RDF Schema logical connectives such as negation, disjunction and conjunction are not provided, thus restricting the expressive power of the ontology.

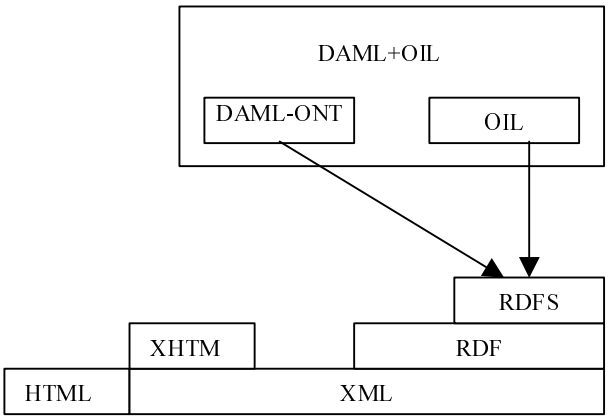

Figure 2 – Layer Language Model for the WWW [Fensel03]

The ontology inference layer (OIL) was sponsored by the European Community via the on-to-knowledge project. OIL sprung from the need of an expressive language for creating ontologies on the web, since RDF provides inadequate expressiveness and lacks formal semantics and reasoning support. OIL's formal semantics and efficient reasoning support is provided by Description Logics. The semantics of OIL rely on a translation into the description logic $\mathcal{SHIQ}$ extended with concrete data types, $\mathcal{SHIQ}(\mathcal{d})$. A complete mapping of Oil to $\mathcal{SHIQ}(\mathcal{d})$ is available in [Horrocks99]. The OIL community made available tools that support editing and ontology reasoning.

The Defense Advanced Research Projects Agency (DARPA) in conjunction with the W3C is developing the DARPA Agent Markup Language (DAML) by extending RDF with more expressive constructs aimed at facilitating agent interaction on the web [Hendler01]. DAML released its first ontology language specification, DAML-ONT in October 2000. In December of the same year DAML+OIL was released to replace DAML-ONT. The formal semantics of DAML+OIL is provided as a mapping to first order predicate logic, written in ANSI Knowledge Interchange Format (KIF) [Genesereth91]. DARPA

maintains an ontology library with near two hundred entries made publicly available at http://www.daml.org/ontologies/. The large adoption and installed base of DAML ontologies is making it a favorite language to provide semantic interoperability on the web [Buranarach01].

A new ontology language is being recommended by the W3C consortium. OWL (web ontology language) is a "revision of the DAML+OIL ontology language incorporating lessons learned from the design and application of DAML+OIL" [McGuiness03]. The documentation for OWL is still in the working draft stage, the last artifacts were released March 2003. As of today, there is no tool to support ontology development using OWL.

Other ontology languages were proposed outside the framework illustrated in Figure 2. The Simple HTML Ontology Extension (SHOE) developed at the University of Maryland, prior to XML and RDF [Luke96]. SHOE is an ontology-based knowledge representation language that is embedded in web pages. The underlying philosophy of SHOE is that intelligent internet agents will be able to better perform their tasks if the most useful information is provided in a structured way. SHOE extends HTML with a set of knowledge oriented tags and associates meaning to the contents of a page by making each web page commit to one or more ontologies. The ontologies permit the discovery of implicit knowledge through the use of taxonomies and inference rules. Compared to RDF, SHOE is analogous an RDF Schema but with less expressive power [Buranarach01], e.g., SHOE does not allow negation in the claim statement nor the subclass relationship for properties. Maintenance of ontologies is also an issue in SHOE, for they are embedded in the web pages as opposed to as separate objects.

The ontology interchange language, Ontolingua [Farquhar97] was designed to support the design and specification of ontologies using a semantics based on KIF [Genesereth97]. The set of KIF expressions that Ontolingua allows is defined in an onology, called the Frame Ontology. The Frame Ontology specifies, in a declarative form, the representational primitives that often supported with special-purpose syntax and code in object centered representation systems.

An Ontolingua ontology is composed of classes, relations, functions, objects distinguished and axioms. The expressive power provided by Ontolingua is unmatched by the previous languages surveyed. Unfortunately no reasoning support is provided until this date.

The development of ontologies, using any of the languages reviewed in this section, would be impossible without tools to filter the markup and present the information in a human understandable way. In the next section we survey some of the tools available for this purpose.

4. Ontology Tools

A few tools to support ontology construction and editing are currently available. Protégé-2000 is a graphical tool for ontology editing and knowledge acquisition [Noy01]. It was developed using open source architecture and allows for the customization of the tools used to create knowledge base. Protégé-2000 does not favor any specific ontology language. The creators of the tool hold that there is little point in designing a tool that serves one language specifically. In contrast they suggest using the customization facilities provided by the open source architecture of Protégé-2000 to adapt and experiment with different ontology languages by the use of special plug ins. Ontologies are built in Protégé-2000 by creating classes, instances of those classes, slots representing attributes of the classes and instances and facets. A plug in for the OIL language exists and a DAML one is currently being developed.

OilEd is an ontology editor that allows its users to build ontologies using DAML+OIL [Horrocks99, Bechhofer01]. The initial intention behind OilEd was to provide a simple editor that demonstrated the use of, and stimulated interest in, the Oil language. OilEd is an ontology editor and not an environment, i.e., it does not provide support to ontology integration (alignment) or versioning OilEd has proved very popular, with over 2000 downloads from the original site in the last 12 months [OilEd site]. It has been used within a number of companies and institutions for both teaching and research purposes.OilEd is now available as an open-source project under the GPL licence at [OilEd site]. Reasoning support for OilEd is available and provided by the FaCT (fast classification of terminologies) inference engine, publicly available at http://www.cs.man.ac.uk/~horrocks/FaCT/.

The reasoning services provided include inconsistency detection and determining subsumption relationships. OIL provides an extension to RDF and RDFS. Based on its RDF syntax, ontologies written in OIL are valid RDF documents.

OntoEdit is an ontology environment that supports the W3 standards by exporting ontologies to the RDFS, XML and DAML+OIL formats [Erdmann02]. Similarly to software development environments, OntoEdit provides a method to support ontology development. It is comprised of three stages: requirements, refinement and evaluation. Integrated tools provide automated support to each stage. OntoEdit is a commercial tool.

Chimaera is an ontology environment that supports the creation and maintenance of distributed ontologies. Although Chimaera can be used to edit ontologies, its focus is in merging multiple ontologies together and diagnosing individual or multiple ontologies. It supports

users in tasks such as loading knowledge bases in differing formats, reorganizing taxonomies, resolving name conflicts and browsing ontologies [McGuiness02].

5. Ontology Development Methods

The importance of ontologies as a model for capturing and reusing knowledge is well understood among the research community. Ontology development, however, seems to be an art rather than a science [Uschold96]. So far, a number of research groups have been trying to tackle this problem by providing methodologies for ontology development. Ushold and King proposed a methodology based on the experience gained from constructing the Enterprise ontology [Ushold96, Ushold98]. The authors refrain from providing guidelines on elicitation and the identification of information sources for building the ontology.

Gruninger and Fox propose the Toronto Virtual Enterprise (TOVE) methodology [Gruninger95]. This methodology was derived from the authors experience in the development of ontologies in the business process and enterprise domains. The authors use what they call motivational scenarios to describe problems or examples that are not adequately covered by existing ontologies Following the scenarios, one must elaborate the competency questions for the ontology, which helps its evaluation (analysis) In our opinion the major shortcoming of this approach is to suppose that the ontology concepts and relationships could be easily derived from the motivation scenarios. In fact, scenarios are best employed to model dynamic aspects rather then static ones, e.g., concepts and relationships as intended by the authors.

Noy and McGuiness provide what they call a simple knowledge engineering methodology [Noy01a]. Built on their own experiences with the ontology-editing environments Protégé-2000 [Noy01b], Ontololingua [Farquhar97] and Chimaera [Chimera00], reviewed in section 4, the authors provide a guide to help new ontology designers develop their ontologies. The authors provide no guidance to concept elicitation but put forward a lengthy discussion on design decisions, e.g., when to introduce a new class as opposed to represent it by means of restrictions and provide useful comments on terminology choices. Again, compared to this line of work we see our process as strong based on a disciplined elicitation strategy.

Methontology is a framework developed in the artificial intelligence laboratory of the University of Madrid that, among other functionality, provides support to ontology construction [Fernández-López97, Gómez-Pérez98]. Associated to the Methontology framework is the Ontology Development Environment (ODE) that provides automated support to the development activities. The authors make use of elicitation techniques very similar to the ones we use, e.g., structured interviews, document reading and questionnaires. The modeling of the concepts, however, seems to very heavy weight. The LEL on the other hand, uses a very simple structure, based on the denotation and connotation of terms. We have observed that the simplicity and organization provided by the lexicon is a key factor in validating it with users [Hadad00].

5.1. Lexicon Based Ontology Construction

Available methods for ontology construction, such as the ones presented in the previous section, , concentrate in the modeling aspects and are either vague or lacking on how concepts and relationships are to be elicited from the macrosystem. The result is the ad hoc use of elicitation techniques, whose choice depends solely on the developer's personal experience and skills. As such, we are proposing a process for the construction of ontologies, which is centered on a established elicitation strategy based on the concept of application languages. This concept is rooted on a representation scheme called the language extended lexicon (LEL).

This technique has long been incorporated to our requirements elicitation practice and it focuses on using the language of the problem to describe the problem [Leite90]. The lexicon will provide a systematization for the elicitation, model and analysis of ontology concepts. The underlying philosophy of the lexicon falls in the *contextualism* category, according to which particularities of a context of use of a system must be understood in detail before requirements can be derived [Potts97]. This approach is new to ontology building, that traditionally associates generalization and abstraction approaches to the organization of information.

5.1.1 Extended Lexicon of the Language

The Language Extended Lexicon (LEL) is a representation of the symbols in the application language [Leite90, Leite93].It is anchored on a very simple idea: understand the language of the problem without worrying about understanding the problem. Each term in the lexicon has two types of description. The first type, *notion*, is the denotation of the term or phrase. The second type, called *behavioral response*, describes the connotation of the term or phrase, that is, provides extra information about the context at hand. In addition, the lexicon terms are classified in four categories: object, subject, verb and state. Figure 6-a shows an example of a lexicon term of type subject.

Central to the LEL are two principles. The first is to maximize the use of other lexicon terms when describing

the notions and behavioral responses of a new term. This is the *principle of closure*. The second is to minimize the use of terms, external to the Universe of Discourse. If unavoidable, make sure they belong to the basic vocabulary of the natural language in use, and, as much as possible, have a clear mathematical representation (eg. set, function). This is called the *principle of minimal vocabulary*.

The elicitation of LEL terms is usually performed by a combination of elicitation techniques. The enumeration of interviews. In unstructured interviews the general heuristic is to pay attention to terms used frequently and which seems unfamiliar or is used in an unfamiliar way. Once the initial terms are enumerated, either structured interviews or detail reading of documents can be employed to assign notions and behavioral responses to the listed terms. Since the lexicon is a self contained network, by the *closure principle*, it is natural that during the elicitation of notions and impacts for one term others terms by also be dealt with.

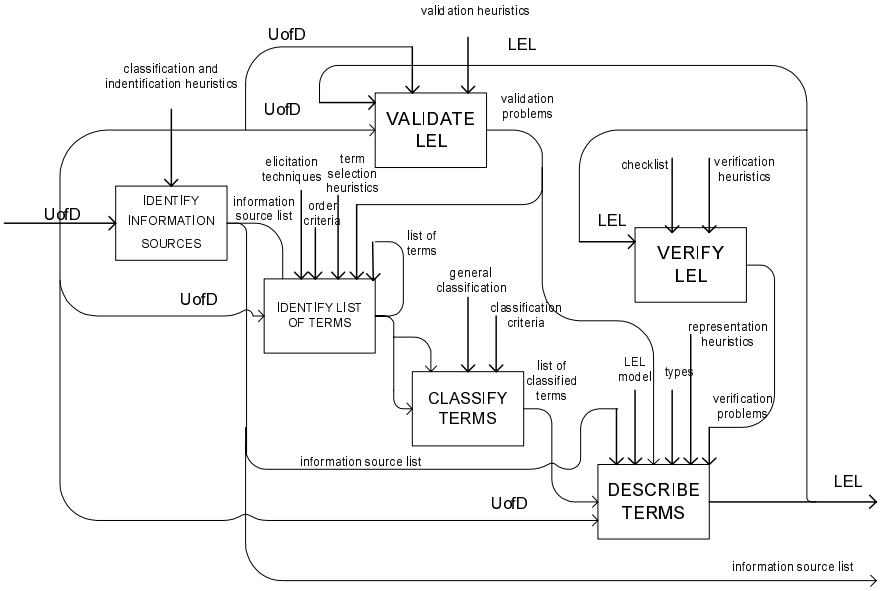

Figure 3 – Lexicon Construction Process [Kaplan00]

languages terms can be done by reading documents of the Universe of Discourse (UofD) and applying frequency heuristics to find proper candidates, by observing the UofD and electing the important terms, or by unstructured

Secondly we must identify a list of terms relevant to the UofD using a set of elicitation techniques. The main heuristic used at this stage is: *list each terms that seems to have a special meaning*. The next step is to classify the terms. They can be one of object, subject, verb or state. The terms are described next. In this step the objective is to elicit the meaning of each term. Each lexicon term is a set of notions (denotation) and behavioral responses (connotation) for that term. When describing the terms in the lexicon one should enforce the principles of closure and minimal vocabulary. The lexicon is verified using an inspection strategy [Kaplan00]. Validation is performed by actors of the UofD using reading techniques.

The process to build the LEL is comprised of six steps, as depicted in Figure 3. First we have to identify the main information sources of the Universe of Discourse (UofD). The most reliable sources are people and documents [Leite93].

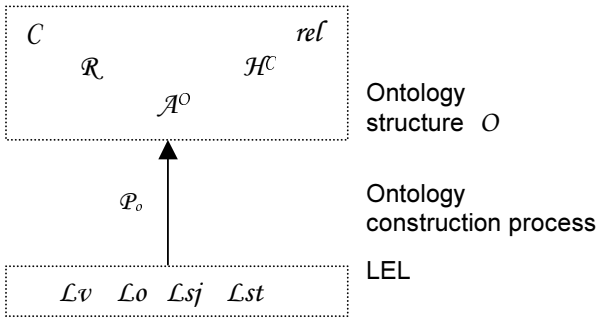

Figure 4 – Ontology Construction Process P_o, (inspired in the layered ontology engineering approach proposed by [Maedche02])

5.1.2 The ontology construction process

Based on the Language Extended Lexicon, presented on the previous section, we propose an ontology building process. The process is independent of the ontology language used in the implementation. The resulting ontology will be expressed using the core ontology elements defined in section 2, a 5-tuple consisting of concepts, relations, a concept hierarchy, functions that relates the concepts non taxonomically and axioms $\{C, \mathcal{R}, \mathcal{H}^C, rel, \mathcal{A}^O\}$ [Maedche02]. Figure 4 shows the role of the lexicon in the ontology structure. On the bottom layer we have the LEL, composed of terms classified into verb ($\mathcal{L}^v$), objects ($\mathcal{L}^o$), subject($\mathcal{L}^s$) and state ($\mathcal{L}^{st}$); on the top layer we have the ontology structure O. Process $\mathcal{P}_o$ maps the lexicon terms into ontology elements. We depict process $\mathcal{P}_o$ in Figure 5.

We detail process P_o as follows:

1. List lexicon terms alphabetically according to their type (verb, object, subject and state)
2. Make 3 lists: concept (C), relations ($\mathcal{R}$) and axioms ($\mathcal{A}^O$). In the concept list, each entry will have a name, a description and list of zero, one or more *rel* (function that relates the present concept to others, non taxonomically). The entries in the relation and axiom lists will have labels only.
3. Using the list of lexicon terms classified as either object or subject, for each term:
- **3.1 Add a new concept to the concept list. The concept name is the lexicon term itself. The concept description is the notion of the term.**
  - **3.1.1 For each behavioral response,**
    - **3.1.1.1 Check the relation list for a relation that expresses it.**
    - **3.1.1.2 If there is none, add new a relation to the relation list. The relation name is based on the verb of this behavioral response.**
      - **3.1.1.2.1 Verify consistency**
    - **3.1.1.3 In the concept list, add a new *rel* to the concept in question. The *rel* is formed by the concept in question + relation (defined in 3.1.1.1) + concept it relates to (The concept is the direct/indirect object of the verb in the behavioral response. It is usually a term in the lexicon and appears underlined).**
    - **3.1.1.4 Check for negation indicators in the minimal vocabulary that relate the term to other terms. Analyze the pair of terms in order to identify a possible disjoint relationship.**
      - **3.1.1.4.1 If true, add the disjoint relationship to the axiom list.**
- **3.2 Verify consistency**

4. Using the list of lexicon terms classified as type verb, for each term:
- **4.1.1 Check the relation list for a relation that expresses it.**
  - **4.1.1.1 If there is none, add new a relation to the relation list. The relation name is the lexicon term itself.**
    - **4.1.1.1.1 Verify consistency**

5. Using the list of lexicon terms classified as type state, for each term:
- **5.1.1 For each behavioral response**
  - **5.1.1.1 Try to identify the importance of the term to the ontology. This strategy is equivalent to the use of competency questions proposed in [Gruninger95]. The questions could be obtained from rephrasing the behavioral responses of the term in question into questions of type, when, what, who, where, why and how.**
  - **5.1.1.2 Check for negation indicators in the minimal vocabulary that relate the term to other terms. Analyze the pair of terms in order to identify a possible disjoint relationship.**
    - **5.1.1.2.1 If true, add the disjoint relationship to the axiom list.**
- **5.1.2 If the term seems to be central to the ontology, classify it as a concept (C).**
- **5.1.3 If the term is not central to the ontology, classify it as a relation ($\mathcal{R}$).**
- **5.1.4 Verify consistency**

6. When all terms are added to the ontology,
- **6.1 Check ontology looking for sets of concepts that share identical *rel***
  - **6.1.1 For each set of concepts that share identical *rel*, build a separate concept list.**
  - **6.1.2 Search the ontology for a concept that refers to all members of this list.**
    - **6.1.2.1 If such concept is not found, search the notion and behavioral response of each member of the concept list trying to identify a common term from the minimal vocabulary.**
  - **6.1.3 Build a concept hierarchy where every member of the concept list is a sub-concept of the one found in 6.12**
  - 6.1.4 **Verify consistency**

Note that every time a new element is added to the ontology, a consistency verification is needed. Ideally, the verification task is done with the aid of automated tools. In the next section we present an example of the ontology construction process proposed, using the OilEd and FaCT tools, that provide automated support to edition and consistency checking respectively.

We understand, however, that it is a fallacy to believe that in some point in the future we will be able to build ontologies based solely in reuse. A certain degree of contextualized (wet) information is always to be expected.

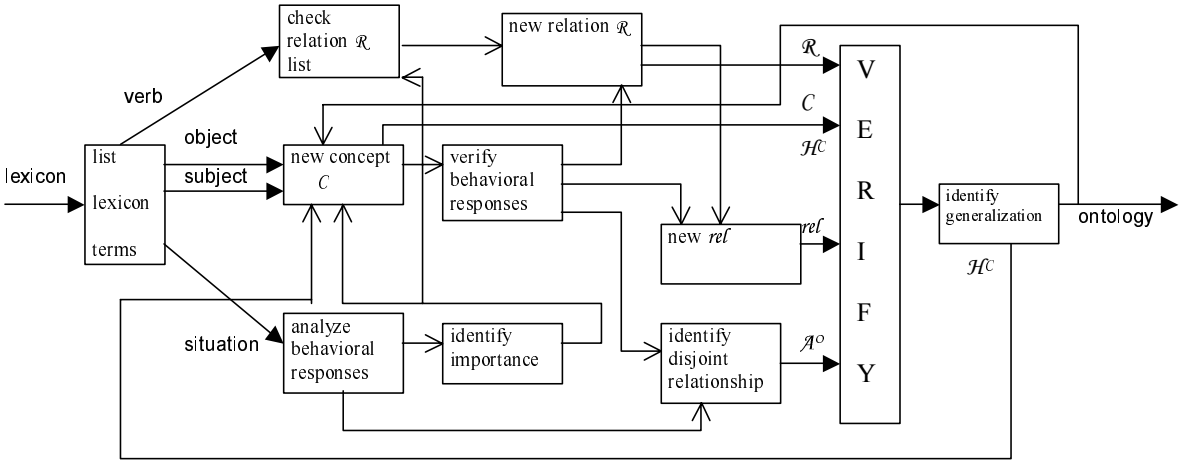

Figure 5 – Proposed Lexicon Based Ontology Construction Process

5.2 Example

In this section we exemplify the proposed ontology construction process presented in section 5. Automated support for ontology edition and reasoning were key factors in our decision for DAML+OIL as the ontology language used to implement our example. Another issue that influenced our decision is today's large community of DAML users and the existence of a public ontology library. We are currently using the OILed tool for edition and the FaCT tool as an inference engine to build our ontologies. OIL's formal semantics and efficient reasoning support is provided by Description Logics. The semantics of OIL rely on translation into the description logic *SHIQ* extended with concrete data types, *SHIQ(d)* . A complete mapping of Oil to *SHIQ(d)* is available in [Horrocks99]. OilEd generates DAML extension ontologies, using an export mechanism.

OIL implements all ontology core elements of the O ontology structure, introduced in section 2. The terminology mapping is depicted in Table I. Concepts are mapped to OIL classes, relations are properties, a class hierarchy is implemented using the subsumption relationship SubClassOf, and the function that relates concepts in a non taxonomic way are mapped to restrictions. Another issue that influenced our decision is today's large community of users and the existence of public DAML+OIL ontology libraries. The prevailing philosophy in the literature is to stimulate maximum ontology reuse, as a means to shorten development time.

O Ontology Structure (section 2)		DAML+OIL (used by the OilEd tool)
C	Concept	Class
$\mathcal{R}$	Relation	Property
$\mathcal{H}^C$	concept hierarchy	Subsumption relationship: SubClassOf
$\mathcal{R}el$	function that relates the concepts non taxonomically	Restriction
$\mathcal{A}^O$	Axiom	Axiom

Table I – terminology mapping between the O ontology structure and the ontology language DAML+OIL

One of the ontologies we have build is the ontology for a meeting scheduler. The lexicon we used in this case was built in 1998, and was inspired by the problem description proposed by Axel van Lamsweerde. This description aimed to be a benchmark in which to test and compare different requirements approaches [Lemsweerde95]. The lexicon describes the procedures of scheduling meetings at the University of Belgrano [Hadad99]. This example was implemented using the OilEd and FaCT ontology using lists of concepts, relations and axioms, as proposed in process P_o. The FaCT tool provides automated reasoning to support our consistency checks. The ontology was written in the OIL ontology language and exported to the DAML format. The terminology used by DAML+OIL is different from the one used by our ontology definition, the O ontology structure proposed by [Maedche02], although the concepts are the same. A complete mapping of terms

is provided by Table I. In this section we are going to use OIL terminology.

Central to the idea of ontology engineering is clearly distinguishing concepts from relationships [Gandon02]. We started building the ontology by first separating the lexicon terms in different listings according to its classification (steps 1 and 2 of process P_o). The lexicon classification of terms helps distinguishing from lexicon terms that are directly mapped to the ontology from those who need further analysis. Terms marked as object and subject are usually mapped directly into classes (concepts) of the ontology (step 3). Similarly the verb terms are usually mapped into properties (relations) (step 4). Properties are the building blocks of class restrictions (function that relates the concepts non taxonomically – rel) that serve to relate ontology classes (step 3.1.1.3). The state terms have to be evaluated in order to decide whether they would be best modeled as a class or property (step 5). Axioms are derived from an analysis of behavioral responses of terms of types subject, object and state (steps 3.1.1.4 and 5.1.1.2).

We start building a skeleton of the ontology using the object and subject terms first. For each term, a new class must be added whose description is the notion of the lexicon term (3.1). The behavioral responses of the term, that describe the relationships to other lexicon terms are to be added as restrictions to the class (3.1.1). The behavioral responses that describe the relationship to other lexicon terms, of types object or subject, should be immediately implemented by creating a restriction that relates both terms, and whose property label is the nature of the relationship (usually described by a transitive verb) as it appears in the behavioral responses of the term in the lexicon (3.1.1.2).

Figure 6-a shows the lexicon term - *initiator* - and Figure 6-b a screen snapshot of the OilEd tool showing the implementation of the class *initiator* and Figure 6-c a screen snapshot of the implementation of the properties of the meeting scheduler ontology. Notice in Figure 6-b that there is a name for the class (*initiator*), documentation (obtained directly from the notion of the lexicon term, as indicated by the arrow), a list of superclasses, i.e., the classes which the class *initiator* is a subclass of, and a list of restrictions (that are derived from the behavioral responses of term *initiator*, The restrictions for class *iniator* are formed using properties of the meeting scheduler ontology. – Figure 6-c shows a part of the list of properties of the meeting scheduler ontology.

Listed as one of the superclasses of the class *initiator (*Figure 6-b) is class *person#2*. Note the symbol # in the name of the class. It indicates that the concept *person* was not defined locally, but borrowed from another ontology. In this case we used the general.1.0.daml ontology proposed by Jeff Heflin as ontology that models many general concepts that are required by other ontologies. This ontology is public and available at [general]. The symbol # stands for the alias for namespace #2, the URL where the file containing ontology general.1.0.daml is located. There is no limit to the number of concepts or different ontologies used in the composition of a new one.

The description of both notions and behavioral responses of lexicon terms make use of terms that are not lexicon terms themselves. These terms are part of the minimal vocabulary of the lexicon, i.e., are assumed to be fully understood by readers, and therefore do not deserve an explicit entry in the lexicon. Something similar happens when building ontologies, i.e., sometimes concepts are 'relevant' but no 'specific' to a domain [Guarino97], in those cases the commonsense meaning of the concept suffices for the ontology purposes. In ontologies, as opposed to the lexicons, if a concept is used in the definition of others it has to be made explicit, but does not necessarily need to be "reinvented". In those cases the ideal procedure is to reuse the concept from another ontology. This is the case with the concept *person* – it belongs to the minimal vocabulary, i.e., bears no particular or specific meaning in the meeting scheduler application (6.1.2.1). Ideally the effort of ontology building should be concentrated in modeling only the concepts with specific contextual meaning and maximize reuse from concepts defined in other ontologies.

The restrictions are obtained from the behavioral responses of the lexicon term *initiator*. The first step is to make sure that there is a property that expresses the verb in the behavioral response (3.1.1.1). This check is done in the properties panel, shown in Figure 6-c. The first behavioral response of the term *initiator*, "defines goal", uses the verb define in the present tense. "Defines" was added as a property (3.1.1.2), so that it could be used to compose one or more class restrictions. The restriction is the first shown in the restriction panel in the bottom right of Figure 6-b (3.1.1.3). The first restriction in the list relates the current class (*initiator*) to another class in the ontology, *goal* (observe the class panel in the left. The class goal appears above the *initiator* class) using the property "defines". This is a non taxonomical relation between the classes *initiator* and *goal*. In fact, the only taxonomical relation held by the class *initiator* is to its superclass, *person#2*.

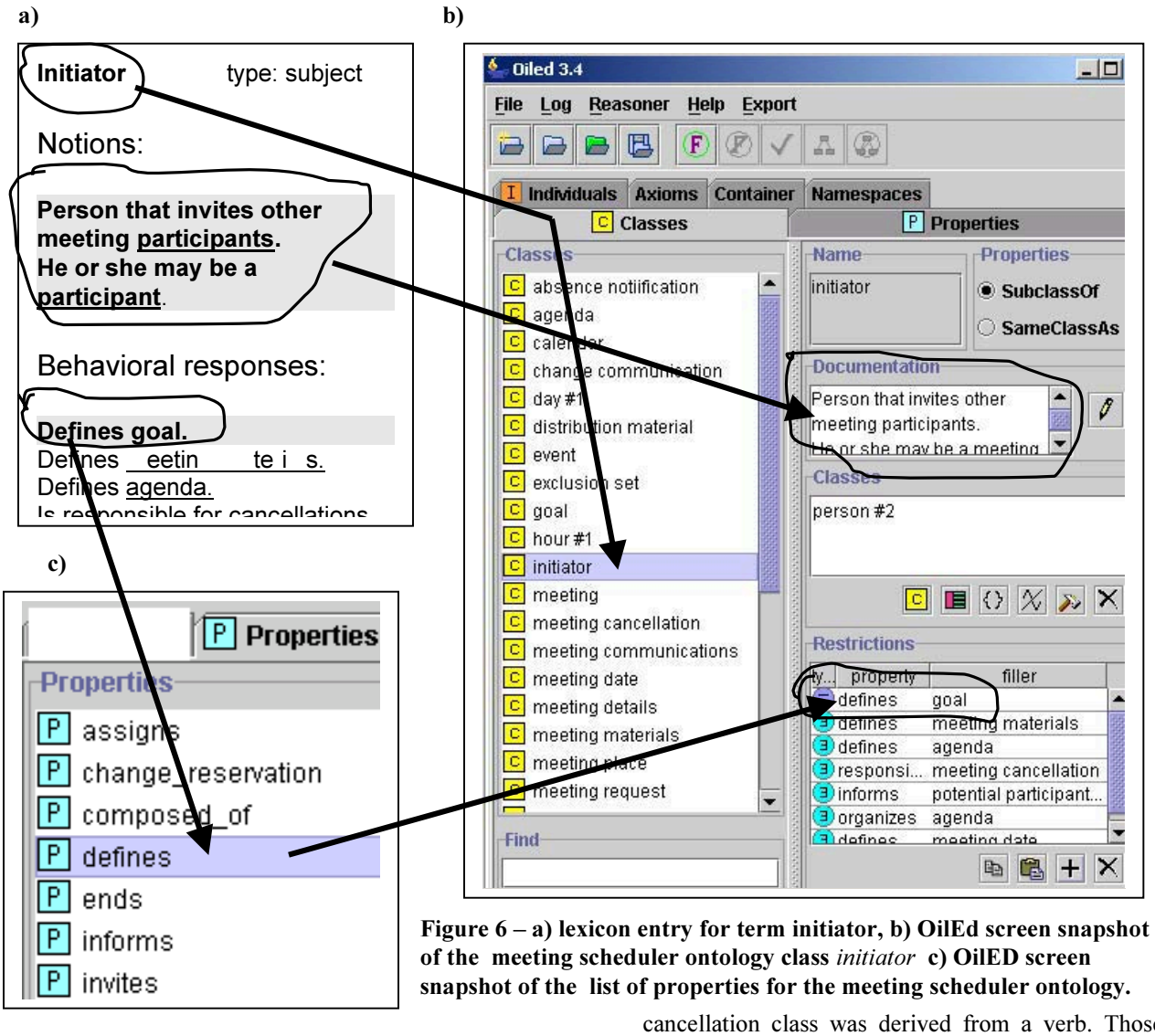

Figure 6 – a) lexicon entry for term initiator, b) OilEd screen snapshot of the meeting scheduler ontology class *initiator* c) OilED screen snapshot of the list of properties for the meeting scheduler ontology.

Note that the terminology used to name the property is the same used in the lexicon term, even the verb tense is respected. Observe the properties column in the restriction panel in Figure 6-b and compare them to the verbs used in the behavioral responses of the term initiator in Figure 6-a. Arrow III shows one of these comparisons.

For the lexicon terms of type verb, we first check the properties list (4.1.1). By the time lexicon terms of type verb are added to the ontology it is possible that its corresponding property was already added as a result of the implementation of the behavioral responses for the terms of types object and subject added in the previous step (3.1.1.2). If that is not the case, we add a new property (4.1.1.1) to the list of existing properties. We show an example of the implementation of the lexicon term of type verb *notify* using the OilEd tool in Figure 7. In the left panel the list of properties is shown. *notify* is highlighted. In the right panel we show the list of classes that use the property. Note that the property *notify* is used by four classes. Three of the classes, *substitute*, *participant* and *secretary*, were derived from lexicon terms of type subject that bear the same name, the meeting cancellation class was derived from a verb. Those terms were added to the ontology before the term *notify*. By the time we got to term *notify* (4.1.1) the property already existed. It is only natural to suppose that, the lexicon being built using the *closure principle*, situations as such are frequent as we approach the end of the ontology implementation process.

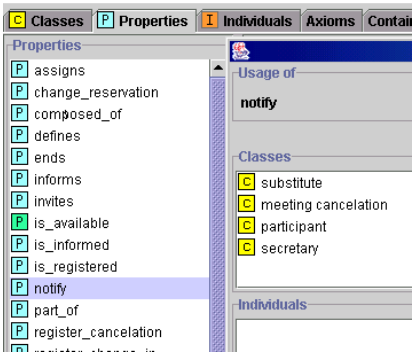

Figure 7– The implementation of lexicon term *notify* as a property

The hardest decision is what to make of the state type terms. There are some cases where a state can be modeled either as a class or property (5.1.2 / 5.1.3). A design decision must be then made. The answer usually lies in the scope defined for the ontology. How important is the state in the application? [Noy01-a]. We recommend the use of ontology competency questions. The latter are the list of questions that a knowledge base based on the ontology should be able to answer [Gruninger95]. This exercise consists of systematically formulating questions whose answers involve the lexicon term in question. Questions should revolve around determining the nature of the term, how it is to be obtained, what other ontology classes are related to it and what if type of questions. Roughly, the competency questions are the behavioral responses of a lexicon term, reformulated as questions (5.1.1.1).

Because ontologies use a taxonomic organization, based on generalization, special attention must be given to lexicon terms that bear part-of/whole relationships among them. Part-of relationships are be expressed in ontologies by means of properties (3.1.1.2) According to Guarino an Welty understanding the proper meaning of the part-of relation is often what ontological analysis is all about [Guarino02].

The ontology construction process is naturally bottom up. We begin with a concept (lexicon term) from the lexicon and systematically add new properties and classes around it. The result is a web of interconnected concepts. In this approach, we emphasize putting together as opposed to diving up, which in turn helps the identification of commonalities among concepts [Uschold96]. The identification of common aspects shared by two or more concepts in an ontology may suggest the creation of a generalization for those concepts (6.1), provided that there is either an ontology concept (6.1.2) or a term of the minimal vocabulary of the lexicon can be found to (6.1.2.1) represent the generalization. We do not introduce new concepts for the sake of generalization alone. We view this strategy very cautiously, and would rather avoid the introduction of artificial concepts that may be foreign to the Universe of Discourse.

6. Conclusion: Application Ontology versus Domain Ontology

We claim that application ontology construction must be of responsibility of the requirements engineering team. This non-functional requirement was recently put forward by the Semantic Web community as a means to support application interoperability on the Web[Berners-Lee02]. From that standpoint, we survey some of the ontology languages, tools and building methodologies that are currently being put to practice by the Web community. In addition, we propose a requirements engineering based process for the construction of ontologies. The process is centered on a established technique, the language extended lexicon (LEL). Based on a lexicon of the application, we provide an implementation independent process that builds an ontology for this application. The lexicon is the main source of knowledge for building the ontology. The lexicon has a quality oriented construction process systematized by sub-processes for elicitation, modeling and analysis, and our ontology construction process preserves this characteristic. As such, we are able to produce an application ontology that is quality oriented.. We demonstrate our approach by modeling a meeting scheduler ontology in the ontology language OIL. The analysis of the ontology is done with the aid of an automated reasoner, implemented by the FaCT tool.

Current ontology literature points out to several topics that require further research[Fensel03, Gandon02, Hendler01, Davies03]. It is our belief that the requirements engineering community can significantly contribute to the advancement of some of shortcomings in the area of ontology development and management. In this spirit we list some of the areas, we identified greater potential for contributions.

Although some of the work in ontology construction mention evaluation aspects, we feel that there is a lack of strategies to validate ontologies. We know ways to verify ontologies, either by checking their internal consistency or by comparing them to other models. However, the literature does not deal with ontology validation as we do for other software artifacts. For instance; how do I test an ontology? Who are the customers of an ontology: agents or human beings? If agents, how could I measure the efficacy of the application ontology? We will continue to improve our process, by means of applying it to other lexicons that we have built in the past, and by studying the ontology validation aspect, focusing on ways to incorporate it into our process. We will also study the evolution aspect.

Concluding, we would like to stress the differences in envisioning ontology in the more traditional way versus the actual usage of ontology in the Semantic web.

Ontology, in the traditional way is supposed to reflect with precision and formality the well established knowledge of a given area. In that sense is it like a theory, it should be stable and throughout used. Of course that its construction demands time. We see a large similarity with the movement towards domain engineering [Prieto-Diaz91] from the point of view of software reuse.

Notwithstanding, a quotation from Hendler - “Instead of a few, large, complex, consistent ontologies that great numbers of users share, I see a great number of small ontological components consisting largely of pointers to each other. Web users will develop these components much the same way that Web components are created” [Hendler 2001] – stresses the point we are making. That is, we will need application ontologies to enable agent cooperation on the web. As such, application ontologies are more much restricted than domain ontologies and have a much more modest objective. We again, stress that requirements engineers should be prepared to produce such application

ontologies, as software designers will have to learn on how to best use these application ontologies. According to a recent report from the Gartner group "By 2005, lightweight ontologies will be part of 75 percent of application integration projects" [Jacobs02]. Research is just opening the trails, avenues will follow as we tackle aspects of evolution and validation.

Bibliography:

Books:

[Davies03] – Davies, J., Fensel, D.; Hamellen, F.V., editors – Towards the Semantic Web: Ontology Driven Knowledge management – Wiley and Sons – 2003.

[Fensel01] – Fensel, D. – Ontologie: a silver bullet for knowledge management and electronic commerce – Springer, 2001

[Fensel03] – Fensel, D.; Wahlster, W.; Berners-Lee, T.; editors – Spinning the Semantic Web – MIT Press, Cambridge Massachusetts, 2003.

[Geroimenko03] – Geroimenko, V.; Chen, C.; editors – Visualizing the Semantic Web: XML Based Internet and Information Visualization – Springer, 2003.

[Hjelm01] – Hjelm, H. – Creating the Semantic Web with RDF – Wiley- 2001

[Maedche02] – Maedche, A. – Ontology Learning for the Sematic Web – Kluwer Academic Publishers – 2002.

[Sowa00] – Sowa, J. F. – Knowledge Representation: Logical, Philosophical and Computational Foundations – Brooks/Cole Books, Pacific Grove, CA – 2000.

Tools:

[OilEd site] - http://oiled.man.ac.uk/

[Erdmann02] – Erdmann, M.; Angele, J.; Staab, S.; Studer, R. - OntoEdit: Collaborative Ontology Development for the Semantic Web - Proceedings of the first International Semantic Web Conference 2002 (ISWC 2002), June 9-12 2002, Sardinia, Italia

[Bechhofer01] - Sean Bechhofer, Ian Horrocks, Carole Goble, Robert Stevens. OilEd: a Reason-able Ontology Editor for the Semantic Web. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396--408. 2001.

[McGuiness02] – McGuiness, D.; Fikes, R..; Rice, J.; Wilder, S. – An Environment for Merging and Testing Large Ontologies – Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), Brekenridge, Colorado, April 12-15, San Francisco: Morgan Kaufmann. 2002. pp.483-493.

[Chimaera00] – Chimaera ontology environment - www.ksl.stanford.edu/software/chimaera

[FaCT] – Fast Classification of Terminologies – TOOL - http://www.cs.man.ac.uk/~horrocks/FaCT/

[Noy01-b] – Noy, N.; Sintek, M.; Decker, S.; Crubezy, R.; Fergerson, R.; Musen, A. – Creating Semantic Web Contents with Protégé 2000 – IEEE Intelligent Systems Vol. 16 No. 2, 2001. pp. 60-71

Methods:

[Fernandez-Lopez97]- M. Fernandez, A. Gomez-Perez, and N. Juristo. METHONTOLOGY: From Ontological Arts Towards Ontological Engineering. In Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, Stanford, USA, pages 33--40, March 1997.

[Gómez-Pérez98] – Gómez-Pérez, A. – Knowledge sharing and reuse – in The Handbook of Applied Expert Systems. CRC Press, 1998

[Gruninger95] – Gruninger, M.; Fox, M. – Methodology for the Design and Evaluation of Ontologies: Proceedings of the Workshop on basic Ontological Issues in Knowledge Sharing – IJCAI-95 Canada, 1995.

[Noy01-a] – Noy, N.; McGuiness, D. – Ontology Development 101 – A guide to creating your first ontology – KSL Technical Report, Standford University, 2001.

[Sure03] – Sure, Y.; Studer, R. – A methodology for Ontology based knowledge management in Davies, J., Fensel, D.; Hamellen, F.V., editors – Towards the Semantic Web: Ontology Driven Knowledge management – Wiley and Sons – 2003. pp. 33-46.

[Ushold96] - Ushold, M.; Gruninger, M. – Ontologies: Principles, Methods and Applications. Knowledge Engineering Review, Vol. 11 No. 2 – 1996. pp. 93-136

Languages:

[Farquhar97]–Farquhar, A. – Ontolingua tutorial – http://ksl-web.stanford.edu/people/axf/tutorial.pdf

[Genesereth91]- M. R. Genesereth: Knowledge interchange format. In J. Allen, R. Fikes, and E. Sandewall, editors, Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference (KR'91). Morgan Kaufmann Publishers, San Francisco, California, 1991

[Heflin01] – Heflin, J.; Hendler, J. – A portrait of the Semantic Web in Action - IEEE Intelligent Systems – March/April - 2001. pp.54-59.

[Hendler00] – Hendler, J.; McGuiness, D. – The DARPA agent Markup Language . IEEE Intelligent Systems. Vol 16 No 6, 2000. pp.67-73.

[McGuiness03] – McGuiness, D.; Harmelen, F. – OWL Web Ontology Overview – W3C Working Draft 31 March 2003

General:

[Berners-Lee02] – Berners-Lee, T.; Lassila, O. Hendler, J. – The Semantic Web – Scientific American – May 2001 –

[Gruber93] – Gruber, T.R. – A translation approach to portable ontology specifications – Knowledge Acquisition – 5: 199-220

[Hendler01] - Hendler, J. – Agents and the Semantic Web – IEEE Intelligent Systems – March/April - 2001. pp.30-37

[Buranarach 01] – Buranarach, M. The Foundation for Semantic Interoperability on the World Wide Web – Doctoral thesis - Department of Information Science and Telecommunications - School of Information Sciences - University of Pittsburgh - November 8, 2001.

[Everett02] – Everett, J.O; Bobrow, D.; Stolle, R.; Crouch, R.; Paiva, V.; Condoravdi, C.; van der Berg, M.; Polanyi, L. – Making Ontologies Work for Resolving Redundancies Across Documents – Communications of the ACM, Vol. 45 No. 2 - February 2002

[Gandon02] – Gandon, F. _ Ontology Engineering: a sinthesis – Project Acacia – INRIA Technical Report 4396 – March 2002 – 181 pages

[Gruninger02] – Gruninger, M.; Lee, J.; - Introduction to the Ontology Application and Design section – guest editors – Communications of the ACM – February, Vol. 45, No.2 February 2002 – pp.39-41.

[Guarino02] – Gurarino, N.; Welty, C.; - Evaluating Ontological Decisions with Ontoclean– Communications of the ACM, Vol. 45 No. 2 - February 2002 – pp.61-65

[Guarino98] – Guarino, N. – Formal Ontology and information systems – In Proceedings of the FOIS'98 – Formal Ontology in Information Systems, Trento – 1998.

[Hadad99] – Hadad, G.; Doorn, J.H.; Kaplan, G.N.; Leite, J.C.S.P. – Enfoque middle-out en la construcción e integración de escenarios - II (Ibero-American) Workshop on Requirements Engineering - Buenos Aires, September, 1999. Pp. 79-94.

[Horrocks99] - I. Horrocks, U. Sattler, and S. Tobies: Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), number 1705 in Lecture Notes in Artificial Intelligence, pages 161-180. Springer-Verlag, 1999.

[Jacobs02] – Jacobs, J.; Linden, A.; - Gartner Group Research Note T-17-5338, 20 August, 2002.

[Kaplan00] – Kaplan, G.; Hadad, G.; Doorn, J.; Leite, J.C.S.P. –Inspección del Lexico Extendido del Lenguaje– In Proceedings of the Workshop de Engenharia de Requisitos – WER'00 – Rio de Janeiro, Brazil – 2000.

[Lamsweerde95] - Goal-directed elaboration of requirements for a meeting scheduler: problems and lessons learnt - in the *Proceedings of the Second IEEE International Symposium on Requirements Engineering (RE'95)* - York, March 27 to 29 – IEEE Computer Society Press, 1995 – pp.194-203

[Leite00] - Leite, J.C.S.P., ., Hadad, G., Doorn, J., Kaplan, G. – Scenario Construction Process - Requirements Engineering Journal vol(5) N.1 pp. 38-61 – Springer Verlag - 2000.

[Leite90] - Leite, J.C.S.P.; Franco, A. P. – O uso de hipertexto na elicitação de linguagens de da aplicação – em Anais do 4˚ Simpósio Brasileiro de Engenharia de Software– editado pela Sociedade Brasileira de Computação – pp.124-133 – 1990.

[Leite93] - Leite, J.C.S.P.; Franco, A.P.M.Franco - A Strategy for Conceptual Model Acquisiton. Proceedings of the IEEE International Symposium on Requirements Engineering, IEEE Computer Society Press, Pags. 243-246, San Diego 1993.

[Potts97] – Potts, C. – Requirements Models in Context - Proceedings of the third IEEE Symposium on Requirements Engineering RE97 – Annapolis, Maryland – 1997 – pp.102-104

[Prieto-Diaz91] – Prieto-Diaz, R., and Arango, G. (eds.) *Domain Analysis and Software Systems Modeling*. Los Alamitos, Ca.: IEEE Computer Society Press, 1991

Mini Tutorial Summary:

The presentation of this mini-tutorial will include the following topics: a) Introduction, b) Ontologies in the Semantic Web (the Tim Berners Lee vision) c) Ontology concepts, d) Ontology languages and tools e) Ontology construction (our approach versus others approaches), f) Application ontology versus Domain ontology, g) Examples, h) Applications g) Further Research.