Software Engineering and Ontological Engineering: Contributions, Perspectives, Challenges and Lessons Learned *

Marcelo José Siqueira Coutinho de Almeida

Universidade Federal de Pernambuco - UFPE
Centro de Informática - CIn
Recife, PE, Brazil
mjsca@cin.ufpe.br

November 27, 2008

Abstract

The growing value associated to semantics in software engineering and the necessity to adopt systematic development approaches in ontology engineering has provoked a convergence between these two areas. In this paper we review the main benefits envisioned from the existing collaborations between software engineering and ontology engineering. Also we enlighten the possible new scenarios to future collaborations what will enable the developing of more sophisticated forms to create, use and manage ontologies.

Keywords: Ontologies, Ontology Engineering, Software Engineering.

1 Introduction

Unlike of some decades ago, nowadays there is a certain consensus about the importance of ontologies in the most different areas of computer science such as Knowledge Engineering, Artificial Intelligence, Software Engineering and Business Management, covering different types of application, like knowledge management, natural language processing, electronic commerce, intelligent

*Article presented in *Software Quality* course in Informatics Center (CIn) of Federal University of Pernambuco in november/2008.

integration of information, information retrieval, design and integration of databases, bio-informatics, education, and, mainly, semantic web [14]. One of consequences of this fact is that larger and increasingly complex ontologies are being developed regarding the demands of these domains.

Therefore it is important that systematic, efficient and, mainly, more productive forms of development can be established. This must be resulted of a discipline that shifts the efforts from an ad-hoc, immature and amateur approaches to a formal and systematic approaches, consisting of basic principles that can be shared throughout all developers community and guide them throughout the creation and maintaining ontologies effectively and efficiently.

According to Gómez-Pérez and colleagues, *Ontology Engineering* (OE) refers to the set of activities regarding to process, life cicle, methods, methodologies, tools and languages that support the ontology development [29]. Conform Devedzic [14], OE can be described as being a set of activities conduced during the conceptualization, design and implantation of ontologies. To Mizoguchi, OE covers a range of topics like philosophy, metaphysic, formalism of knowledge representation, development methodology, common sense knowledge, knowledge reuse and sharing, knowledge management, business process modeling, domain knowledge systematization, retrieval, standardization and evaluation of information from Internet [47].

Behind these efforts to conceptualize OE, it is the need to attend the growing demands of market, taking out the excessive conceptualization discussions and looking for principles like productivity, time and quality. Such initiatives are similar to those taken some decades ago when the software industry felt the necessity to use rigid forms of software development, giving birth to the Software Engineering area [51].

Despite of ontologies are a kind of software like for example a set of conceptual models, it is not suitable to use the methodologies of software engineering to develop them. Ontologies are models of a domain, but they don't generate executable software, what imposes restrictions in its development model.

In this work we present a review about the actual state of existing ontology engineering methodologies, highlighting the limitations found during this research in order to turn possible the proposals of a new, actual, better founded and sophisticated methodology. Also it is our aim to make some comparisons with the software engineering in order to envision the similarities, differences and experiences from that area, possibly to apply them on ontology engineering.

The remainder sections of this work are organized in the following: in

Section 2 we present the motivation to develop this work. In Section 3 we describes the main methodologies of ontology engineering. In Section 4 we make considerations about some negative points in each methodology previously presented. After, in Section 5 we overview the state of art regarding to ontology engineering. In next Section, we approach the similarities, differences, perspectives and challenges of software engineering and ontology engineering. Finally, in Section 7 we conclude the paper and discuss some ongoing and future works.

2 Motivation

Although ontologies are not a new subject in areas like Artificial Intelligence and Knowledge Management, they came back to computer area mainstream discussions in last years after the so mentioned paper of Tim Bernes-Lee [7]. This is probably the start point for what is today known as semantic web. After that moment ontologies passed away from a theoric approach of philosophers and AI scientists to be the silver bullet of a new myriad of web applications.

Even before that, Thomas R. Gruber [30] calls the attention of AI community about the needs of ontologists adopt an engineering perspective of ontology development. Among the reasons presented, knowledge reusability is outlined as one important issue in order to provide a systematic approach of ontology development. With the acceptance by the market and further raising of semantic web, it is natural that the quantity of ontologies being developed grows each time more.

In other hand side, in according to Gómez-Pérez and colleagues [29], the development of area depends basically of comprehensive methodologies in order to shift the ontology development approaches from an ad-hoc to a professional way. Issues as standardized phases, identification of roles, adoption of tools, etc. are posed as crucial to maturing of area and produce knowledge based high quality products. Simperl and Tempiche [57] identified the fact of developers community uses almost no methodology in their projects. This shows the lack of commitment in order to obtain products professionally like already occurs in software engineering for years.

Despite of all attention regarding to ontologies in nowadays and the recent advancements of area, neither methodology was developed to accomplish these advancements and to propose modern forms to approach ontology development. In this sense, we can observe that important themes must be discussed in ontology methodology area. Herein we outline some themes

that we consider as being the ones that deserves urgent attention.

Firstly, we detach the importance of **reusability** to increase all the process. We consider that the way how reusability is being treated is not satisfactory to attend the current objectives of community and market. Since the work of Gruber reusability almost ever is being used in an (full) ontological level, leaving away the (partial) modular level. Some works like [60] proposes and discusses the needs to adopt modules in ontology development and others like [18] outlines the advantages from a software engineering perspective.

A second important theme is **quality**. According to ISO 9000, [72] quality refers to the degree to which a set of inherent characteristic fulfills requirements. In this sense, Deming affirms that productivity is augmented by means of quality [40].

Further, quality affects such the process as its final product. If ontologists aim to develop high level quality ontologies it is necessary to create forms to measure them. Roger Pressman considers the quality of development process is a major issue to observe the quality of product being developed [51]. Since that ontologies have its own peculiarities it is necessary that be developed specific forms to evaluate and enforce quality in development processes.

The third important theme is **market**. Ontologies are being adopted in software products in many forms [32]. Besides this fact, the emerging semantic web will demand ontologies. Recent approaches of distributed computing like semantic web services are an example of how ontology based products can contribute to create complex and larger software architectures in order to provide more useful services. In this sense, we argue the necessity of existence of an ontology market similar to the component market of today which products like *Enterprise Java Beans* (EJB), *Distributed Communication* (DCOM) etc. This will enable a faster, cheaper and better ontology development process.

The last theme is **standardization**. While software engineering community has looking for to standardize their methodologies, approaches, products, etc. through different efforts. Today, the most important organization for software advancement and maturing is the Object Management Group (OMG). We believe that similar behavior must be adopted by ontology community. In this way, concepts, models, approaches, methodologies and tools could be developed under a consensus, reducing efforts and conflicts. In the other hand, software engineering is receiving contributions from the ontologies, improving all the software life cycle.

3 Ontology Engineering Methodologies

In the late 90's, efforts to promote integration and reuse of massive knowledge base written in the formal knowledge and representation of different languages, used in expert systems and intelligent agents, brought back the field of ontologies for the agenda of the research in artificial intelligence. From this fact, innovative tools and techniques were developed to tackle these tasks since then, and the field matured over the years [23].

One of the subareas of ontologies that evolved was the ontology engineering. In the beginning, the development of ontologies was similar to software engineering in its early days because each team adopted its own set of principles, criteria and phases of the project [29]. The process of building ontologies is leaving out these ad hoc efforts to become a strict discipline of engineering. This change is still ongoing, and the methodologies of OE are an active area of today's research [23].

Making an analogy with the software engineering, we can say that the ontology engineering is still moving towards its development, especially regarding the methodological aspects. One of the biggest challenges remains to be finding productive ways to develop ontologies. The reuse of ontologies modules, similar to what is made for years in object oriented programming, is not yet covered by existing methodologies. This facility would allow developers to the same convenience of software developers in to provide resources to compose modules of ontologies, making the process fast, efficient and economical.

In next sections, we will summarize the main methodologies of ontology development in order to better understand the actual ontology engineering scenario.

3.1 Cyc

Probably the first known initiative to develop systematic ways to create ontologies came as part of the Cyc project (reduction of enCYClopaedia) [29]. It started around the middle of 1980's at MCC (Micro Electronics and Computer Technology Corporation) and the goal was to create a broad knowledge base with general information of common sense. Its core consisted of more than one million of sentences entered manually in order to represent what people considers as consensual knowledge about the world.

As shown in figure 1, the development of Cyc ontology was conducted according to three separate phases:

- **Manual extraction of knowledge**: In this phase the extraction is

done manually because of the natural language processing and machine learning systems don't manipulate well the common sense in order to pursue this new kind of knowledge;

- **Extraction of knowledge supported by computers**: this phase can be realized when tools for natural language processing and machine learning can use common sense knowledge already stored at the Cyc base of knowledge to search for new knowledge;

- **Extraction of knowledge managed by computers**: this phase is realized largely by specific tools.

Besides the process, this proposal introduced the concept of micro theory to support the elicitation of information in specific fields such as chemistry and astronomy. These micro theories adapt some hypothesis and make the simplifications in order to facilitate the collection and modeling in some areas.

Furthermore, the aim of Cyc methodology is to support the development of an ontology with the same name, not attending any other projects. Its implementation through three stages emphasizes the simplicity of its design, but on the other hand it leaves gaps and the teams have to create their own solutions.

3.2 Uschold and King Methodology

The work of Uschold and King came from their experience in the development of ontologies in the project of the Office of Enterprise Ontology Applications of Artificial Intelligence at the University of Edinburgh, Germany, in partnership with several companies, among them IBM and Unilever [68].

The method proposed by them is composed by guidelines through scenarios of motivation. This technique is widely used today in software development and it is based on use cases. The steps of this method basically consist of (i) identification of purpose, (ii) building, (iii) evaluation, and (iv) documentation. It can be briefly described as follows [29]:

- **Identification of Purpose and scope**: The goal here is to clarify the motives for the construction of ontology and how it will be used (sharing, reuse or to serve as a basis of knowledge);

- **Building**: This phase is divided into three sub-steps:

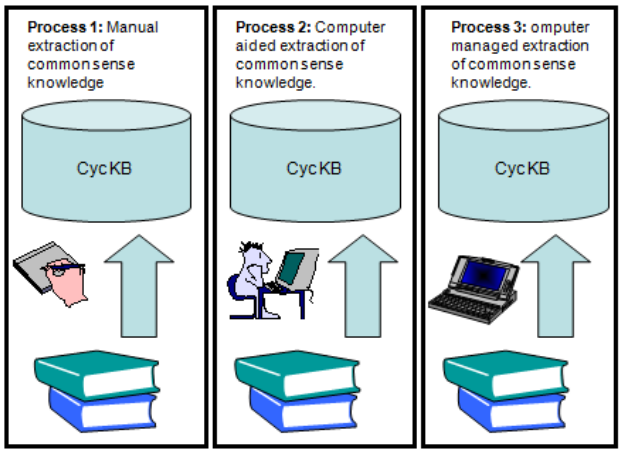

Figure 1: Processes proposed by the Cyc methodology

- **Capture**: textually capture concepts and relationships existing in the domain;
- **Encoding**: codify the concepts and relationships obtained during the capture using any formal language;
- **Integration**: check the possibility to reuse existing ontologies. This step can be done in parallel to the two previous;
- **Evaluation**: verify if the requirements are being met, if the questions of competence are being answered, etc.
- **Documentation**: description of the process through of records. There is no rigor regarding to the form of the document, and users can create their own conventions.

3.3 Gruninger and Fox Methodology

This proposal was developed from the experience of Gruninger and Fox on a project called TOVE (Toronto Virtual Enterprise) developed at the University of Toronto, Canada [31]. This methodology has been used to develop ontologies for the corporate area.

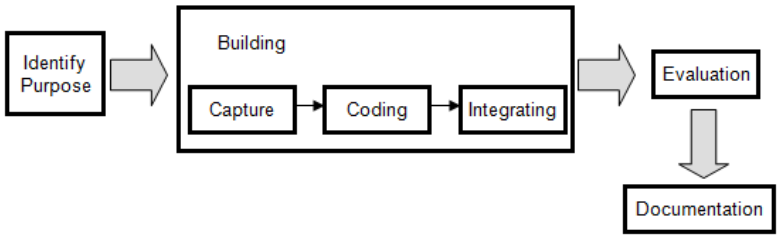

Figure 2: Processes proposed by the Uschold and King.

The inspiration of this methodology was the development of knowledge-based systems using first order logic. It is composed of a sequence of steps which has as their starting point the description of scenarios of motivation. Figure 3 describes the processes of this methodology and then after its given a brief summary of each of them.

- **Description of motivating scenarios**: the motivating scenarios must be related to the applications that will use the ontology and describe a set of requirements that the ontology should satisfy after being formally implemented.
- **Establishment of informal question of jurisdiction**: these issues are written in natural language to be answered by the ontology once it is expressed in a formal language in order to validate it.
- **Specification of terminology using first order logic**: the ontologist can extract the knowledge to be included in the formal definition of concepts, relationships and axioms from the responses obtained in the previous step.
- **Writing of competence questions using formal terminology**: translation of competence questions in a formal way to an informal using first order logic.
- **Formal Specification of axioms**: creation of the rules described in the first order logic to define the semantics of the terms of ontology and relationships;
- **Verification of completeness**: statement of necessary conditions such that the answers to the questions of competence formally described are considered valid.

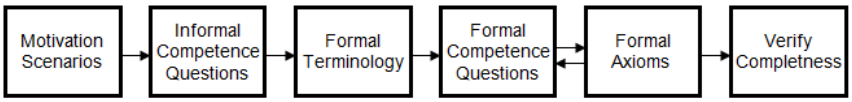

Figure 3: Processes proposed by the Gruninger and Fox.

3.4 Methontology

This methodology is certainly the most comprehensive one for development of ontologies, because it can be used to build ontologies "from scratch" such as from others existing ones. It also supports the reengineering of ontologies. It was developed in the Polytechnic University of Madrid in the laboratory of Artificial Intelligence and its main inspiration came from the main activities identified by the standardized process of software development of the IEEE [36] and the methods of knowledge engineering [27] [71]. Such as it is depicted in figure 4 the development processof Methontology is composed by the following steps:

- **Plan**: in this phase it is described the planning of tasks to be performed;
- **Specification**: it is defined the scope and goals of ontology. Questions like "why this ontology is being built?" and "who are its users?" are answered;
- **Conceptualization**: the terms of ontology are described. Techniques normally employed in the requirements elicitation of software engineering can be used;
- **Formalization**: the conceptual model obtained in previous phase is formalized through a language for formal description of ontologies;
- **Integration**: it is realized the integration of ontologies being built with the existing ones;
- **Implementation**: the ontology is implemented in any language, as OIL (Ontology Inference Language) [5], DAML + OIL (DARPA Agent

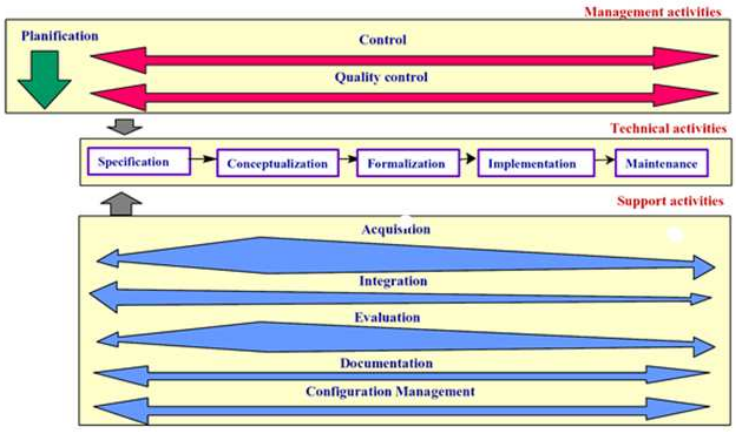

Figure 4: The Ontology Development Process of Methontology.

Markup Language) [33] and OWL (Web Ontology Language) [8], to allow their processing;

- **Evaluation**: the developed ontology is evaluated in order to ensure quality and adherence to standards;
- **Documentation**: it is recorded relevant information to the ontology to facilitate, mainly, its maintenance and development;
- **Maintenance**: it is made adjustments in the ontology so that it can reflect the changes contained in knowledge that it represents.

4 Identified Problems in Current Methodologies

Despite of all existing methodologies, the ontology development activities still suffers of various problems, what prevents the area to get a better status and finally reach the market. In next subsections we will summarize some problems that we consider as being particularly important and deserve special and urgent attention by the ontology development community.

4.1 Absence of Mechanisms to Reuse Modules

Although of already existing research efforts to introduce modularization on ontology development process, neither one of the existing methodologies hold it. We believe that this new promising approach (i.e. the use of modules) just will succeed if it will make part of a methodology. In other words, it won't be possible successfully to use a sophisticated concept like a module without use a systematic set of guidance as occurs in Software Engineering Component Based Development (SECBD) [13].

4.2 Absence of Mechanisms to Develop Distributed Ontologies

Distributed software development teams are very common in nowadays. It is very probable that the same approach can be used in ontology development, creating a distributed authoring environment which people localized in different regions worldwide can share their efforts to create new ontologies. In the same way, the networked communication infrastructures like the Internet can turn possible the existence of modules distributed over this infrastructure similar to what happens in database distributed environments. One foreseen benefit is the possibility to compose and decompose pieces of ontologies or full ontologies from different sources in order to obtain larger ontologies.

4.3 Absence of Mechanisms to Trace Competence Questions

One important issue when we are developing a ontology is: what are the classes, relations, etc. able to answer a specific competence question? Dynamic use of ontologies can include or remove determined questions along the time. Furthermore, some classes, relations, etc. cannot make sense to be there and must be removed. Similarly, in UML (Unified Modeling Language) [52] it is possible to trace a single use-case through all the development phases, since the requisites until the source code.

4.4 Absence of a Graphical Language to Better Communicate with the User

It is very important that the representation of ontologies be made by means of graphical notations, like UML. The model created with this language could generate an ontology. However a graphical language is not sufficient to introduce all the necessary information to describe a specific domain.

The language could include constraints of diverse kinds, similar to OCL (Object Constraint Language) [49]. The main vantage of this language is the simplification of communication between client/domain specialist and ontology engineers. Without knowing the details of ontology development and representation, users will can to analyze and confirm if what is on the picture is right or wrong.

4.5 Absence of Mechanisms to Evaluate Inconsistence Among Modules

We believe that the inclusion of modules won't be a soft process, what will demand additional work. Probably modules coming from different ontologies will conflict one with others and questions like these will rise: How they collide? How much conflict? Where is the conflict? Is the conflict acceptable?

4.6 Absence of Generalization

Some methodologies were developed in order to solve a specific problem, like Cyc and TOVE. In order to satisfy all existing (or not) domain problems it is necessary that the methodologies can be general as possible at same time it must allow to handle all specific details by means of adequate resources like graphics and appointments.

4.7 Absence of a Role-Based Development Model

Ontology is a complex artifact. Further its development process must be conducted by teams composed by experts. Some of them are specialized in analysis, some in tools, and others in tests. A methodology must encompass the possibility to gather various persons and each one concentrate in particular tasks. In modern methodologies of software engineering like XP (eXtreme Programming) [6] defines specific roles in each team.

4.8 Absence of Standardization

To be an universally accepted and adopted, it is fundamental that the methodology be supported and used by a significative quantity of users. Besides it has good technical qualities, it should be standardized like RUP, XP and others approaches of software development.

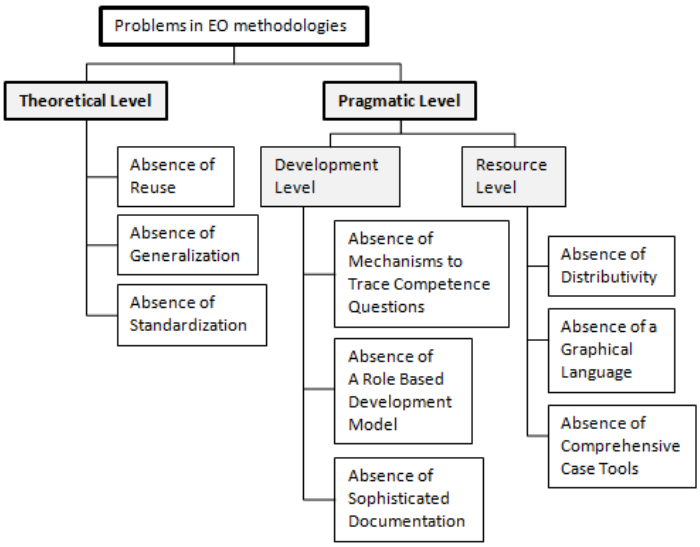

Figure 5: Taxonomy of observed problems in OE methodologies.

4.9 Taxonomy of Problems in Ontology Engineering Methodologies

According to what was discussed in last section, it was organized the taxonomy depicted in figure 5 to better understanding the problem organization and further proposal of solutions.

This taxonomy is initially divided in two parts: **Philosophical** and **Pragmatical**.

The first encompass the *absence of reuse* (*monolithic approaches*), *absence of generalization* and *absence of standardization*.

The pragmatical level contains the following: **Development** and **Resource**.

Development level consists of the *absence of mechanisms to trace competence questions*,*absence of a role-based development model* and *absence of documentation forms*. Resource level consists of *absence of distributivity*,*absence of a graphical language* and *absence of a suitable tool support*.

5 State of Art in Ontology Engineering

5.1 Patterns in Ontology Development

Patterns are a successful strategy in software engineering to promote the reuse of knowledge and expertise about the software development [24]. Currently there are patterns for all life cycle activities, since from the analysis phase until to test phase. Probably the major success regarding to patterns usage in software engineering is that one obtained by means of "Gang of Four" (GoF) catalogue [24]. This catalogue was the very first initiative in computing to reuse experiences in a patterned way. It contains 23 patterns which each one is depicted in a well organized template in order to easy the learning. The main advantages are the time and effort reduction and the increased quality [9].

Although patterns were not adopted largely by ontology community yet, some recent initiatives have been given rise inspired by this experience. The reasons to invest efforts in promote the pattern based development are straightforward: firstly, patterns simplify the work of domain expert and knowledge engineers. Secondly, patterns help to integrate ontologies [74]. Thirdly, manual ontology development is a tedious and complex task. Therefore, it is need to create automatic or semi-automatic ways to develop ontologies [9].

One interesting work is being realized by the Semantic Web Best Practices and Deployment Working Group (SWBP) of W3C, including a task force on Ontology Engineering Patterns [64]. These efforts have produced some design patterns in language level (for example, Web Ontology Language - OWL) regarding to whole-part, time and semantic integration issues.

The work of [9] focuses in proposing a classification scheme for ontology patterns. The scheme divides ontology patterns in five levels:

- **Application Patterns**: aim to describe generic ways to use the implemented ontologies in terms of purpose, context, interfaces etc. this idea of abstracting the best ways to apply and use an ontology, or several ontologies, within some context or application is an important issue.

- **Architecture Patterns**: aim to describe a generic way to design the overall structure of an ontology, in order to fulfill the goal of the ontology in question. Important issues here are whether to divide the

ontology into components or modules, or to divide into layers or use other construction principles.

- **Design Patterns**: aims to describe a generic recurring construct in ontologies. Similar to Software Engineering Design Patterns (SEDP) [**?**], the main idea is to be specific enough, allowing them to be used automatically or semi-automatically, and at the same time generic enough to be useful in several ontologies of a certain domain [74].

- **Semantic Patterns**: aims to describe a certain concept, relation or axiom in a language independent way.

- **Syntactic Patterns**: aim to describe representation symbols in a language dependent way in order to create a certain concept, relation or axiom [64].

Specifically to Design Pattern Level, [74] has proposed conceptual patterns based in experiences in ontology engineering experiences in Laboratory for Applied Ontology[1] (LOA). These patterns emerged out from different domains, different tasks and while working with experts from different areas. These approach bases on a set of typical competence questions like these following:

- Who does what, when and where?
- Which objects take part in a certain event?
- What are the parts of something?
- What is an object made of?
- What is the place of something?
- What is the time frame of something?
- What technique, method or practice is being used?
- Which tasks should be executed in order to achieve a certain goal?
- Does this behavior conform to a certain rule?
- What is the function of that artifact?

[1] http://www.loa-cnr.it

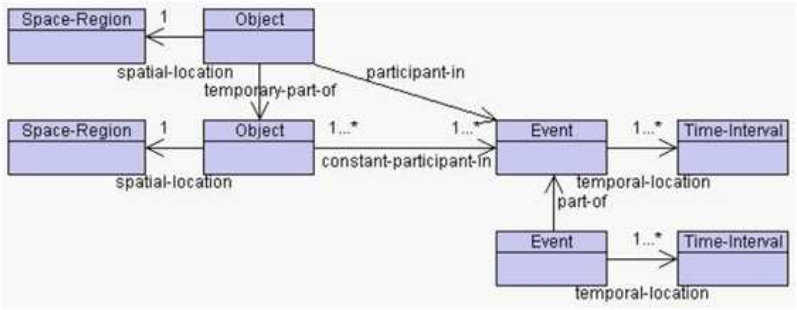

Figure 6: Participation Pattern.

- How is that object built?

Each question is called *Generic Use Case* (GUC). It provides the generic vision of a recurrent question in various different domain problems. Each GUC (or even a set of GUC) must be encoded throughout a formal pattern called Conceptual Ontology Design Pattern (CODeP). A CODEP is a template to represent, and possibly solve, a modeling problem and describes a "best practice" of modeling [74]. Some examples of CODePs are: participation pattern, role-task, information-realization, description-situation, design-object, attribute parametrization etc.

A very simple example is the Participation Pattern. As depicted in figure 6 it assumes a relation participant-in of various objects in a determined event during a time interval. Some objects participates constantly and others participates during a certain part of time. Also, Objects are localized in a determined Space Region.

5.2 Modular Ontologies

Despite the fact that one of the major goals of the ontologies is the reuse of knowledge, the resources currently available in existing methodologies have little or nothing to offer in that sense, almost always confined to the extension or import of whole ontologies. Thus, despite the fact that definitions like that presented in [30] suggest that the creation of ontologies in itself encourage the reuse of knowledge, this fact is not evidenced in practice.

Thus, concepts such as *modularization*, *composition* and *component* are

not taken into account, despite having significantly facilitated the collaborative development and reuse within the scope of SE, and therefore could, in principle, be applied successfully in OE. The fact is that there is still a theoretical basis mature enough behind the concept of modules, which precludes their methodologies, tools and languages of developing ontologies evolve in that direction, reversing a tradition exists in information technology to apply the "divide and conquer" principle.

The next section will discuss some pertinent issues related to creating plenty of ontologies based on modules. This discussion aims to highlight the difficulties in the approach based on modules and therefore draw attention to the importance of a methodology based on modules that guide developers in conducting their work.

5.2.1 Module Definition and Description

Modularization is a technique already widely used successfully in SE. Depending on the maturity of the objects as a guideline for development in the past decade, the concept of module has been widely used as a form of decomposition of the areas being examined, as the basic unit of design, and as abstract data type implementation. More recently the concept of component has attracted the attention of the community as the great promise for the development of software in the same way that traditional engineering (automotive industry, the electric-electronics industry, etc.) create their artifacts.

In order to develop an engineering Modular Ontologies (or based on modules) with resources and know-how mature, we need some way of a consensus in the community of developers about what is a module in the context of ontologies. An intuitive understanding of the concept of module in the context of ontologies is that this is a part of a whole that makes any sense. Moreover, in some way, it can be separated from the whole, without having necessarily to keep the same functionality [62]. Therefore, part of an ontology should continue being an ontology, i.e., should consist of a coherent set of classes, relationships, axioms and bodies, and a way to reuse these elements into a different environment, probably different from that for what was projected.

A very interesting definition is given by Doran: "A ontology module is a reusable component of a larger or more complex ontology, which is self-contained but at the same time, bears a clear relationship to other modules of the ontology" [15]. As a result, there is a module that can be used as designed or can be extended with new concepts and relationships.

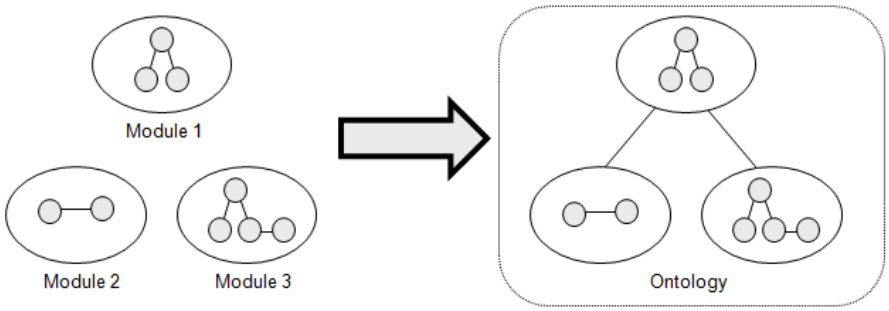

Figure 7: Module Composition.

A module is self-contained if all the concepts in the module are defined in terms of other concepts contained in the module itself, rather than refer any concept outside. In [26] it stated that "the module to an entity is the minimum subset of the ontology axioms that captures its meaning precise enough and therefore the minimum set of axioms that are required to understand, process,evolve and reuse the entity. "To be able to reuse a module, there must be a way to describe its functionality. That definition, however, still needs to be deeper as the fact that module is a "sense" or not is very subjective, leaving room for different interpretations. From the viewpoint of an application, for example, a module must be able to respond to any query sent to the ontology as a whole.

5.2.2 Composition and Decomposition

Module is the basis for two major operations for the emergence of an engineering modular ontologies: composition and decomposition.

In composition, as shown in Figure 7, modules are developed independently and are linked together to form a new ontology and thus provide a new feature. This modular architecture should describe any facility to import the modules.

A module should describe only a portion of a field so that it can be reused in other areas. For example, a module describing an organ of the human body should be used in an ontology of congenital diseases and in an ontology about animals in general.

In decomposition, as shown in Figure 8, modules are built from the divi-

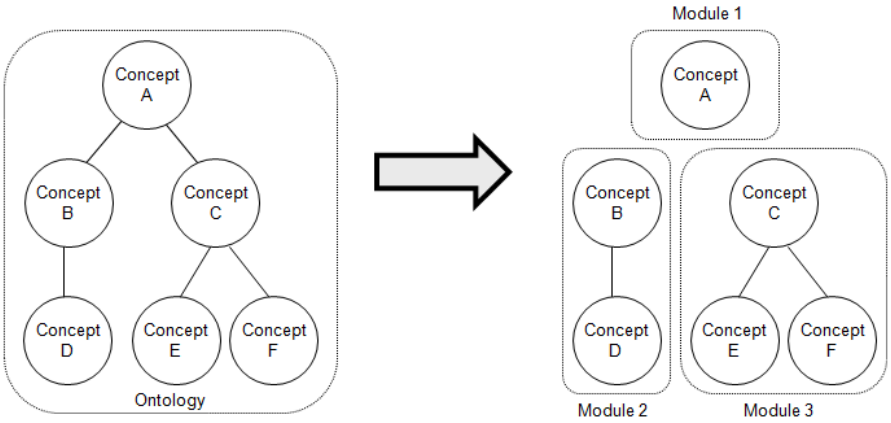

Figure 8: Module Decomposition.

sion of an existing ontology. Thus, according to criteria established by the designer of the new ontology, the ontology is broken into parts semantically well-defined. The problem with this approach is to define what will be the criterion of division, because according to this is that you can identify which classes, relationships, instances and axioms are held in each module. Currently, it is planned that this division can be done manually (by a specialist), or (semi-)automatically (by ontologies management systems).

The understanding of what the modularization means exactly, and what are the advantages and disadvantages, dependent on goals that are allocated.

5.2.3 Objectives of Modularization

According to [62], the main objectives in the adoption of modules for construction of ontologies can be described as follows:

1. **Scalability**: This goal is well connected to the issue of performance during the processing of an ontology. It is known who think their performance has deteriorated as increasing the size and complexity of the ontology being processed. Thus, if the processing occurs in only a part of ontology (module), the resulting performance will be much more acceptable. The challenge in this case lies in the fact that we need to properly define the size of each module that can answer the

query in question. This goal is associated with the process of composition and decomposition. In the case of the composition, it turns now possible to leave the modules separately rather than integrate them and form an ontology in an only file. This would involve the adoption of mechanisms for distributed reasoning, since the modules are possibly distributed in a network of modules of ontologies. In the case of decomposition, the concept of module appears as a natural approach, depending on the need to find more efficient ways of information retrieval. Furthermore, the issue of scalability can be divided into two topics:

- **Scalability for information retrieval**: in this case the modules serve as a prerequisite to define the areas where to make the searches to a given query, restricting the processor to focus only on certain classes, relationships, axioms and instances.
- **Scalability for development and maintenance**: in this case the modules helped to understand the impact of an upgrade ontology. Thus, concentrate updates within a module would be an interesting way to avoid it or to propagate throughout much of ontology.

2. **Complexity Management**: this goal is very similar to the previous one, there is a tendency to create an inverse relationship between the project's ontology and its complexity. Therefore, it is expected that smaller and more specific modules can be properly manage them, and subsequently compare them to form well designed ontologies.

3. **Understandability**: understanding the contents of an ontology depends much of its size and complexity. Thus, the modularization will provide a more rational during the learning, as being it a human agent or an intelligent mobile agent.

4. **Customization**: the development of ontologies may be the result of efforts made by several people, possibly located in different localities, or the availability of modules in repositories scattered over networks of information, such as the semantic web. This scenario describes the need to describe the individual responsibility for creating the various modules being used in the creation of new ontologies.

5. **Reuse**: this goal is related to the need for approaches that promote the rapid development of new ontologies from existing modules and

has direct implication in large adoption of this technology.

5.2.4 Research Challenges in Modular Ontologies

In this subsection we present a few issues that are seen as fundamental to the creation of a modular OE and therefore need special attention in the application of effort, investment and research. Clarifying these issues will be of great importance to the development of methodologies, tools and languages based on the concept of the module.

Criteria for Modularity: It is important that the definition of the modules follow certain criteria. However, the way it should be done is an issue that remains open and that it needs further studies so that they can be clearly defined. The criteria may vary according to how occurs its creation, use or maintainability. If the existence of the composition forms a highest-level ontology which they will form or join, there is a concern about what should be contained in the module because they already exist, except in cases where there is a need to re-allocate or redistribute the content of the modules within the collection. Moreover, in the case of decomposition, the ontology already exists and the modules will be made in accordance with the established criteria. Set a criterion remains as a challenge for researchers. Two ways can be considered:

- **Manual decomposition**: simplest solution, but the quality of the result depends entirely on the knowledge and expertise of designers of ontology. It is recommended that the development of a mechanism for monitoring the reliability and trust services offered by the modules, which could allow a reorganization (or resizing) of the modules when necessary;

- **Decomposition automatic or semi-automatic**: This approach requires knowledge of the requirements of the application. This knowledge can be obtained by analyzing the queries that are sent to the ontology and the storage of ways (paths) that they will go in ontology during the implementation of these. The decomposition of an ontology can also be based on performance criteria. In that case the criteria for the application, such as semantic aspects, are not necessarily considered.

Properties of Modules and Modularization: A property very important in the context of ontologies modular and that represents an important field for search concerns to *correctness*. The problem can be depicted

in two ways: *correctness* of the modules *individually* and correctness of ontology formed by all the integrated modules. Regarding correctness of modules, one should answer questions like: (i) what ensures that a module is an ontology? e (ii) What ensures a module that makes sense semantically?

Regarding to *correctness* it should be considered the issue of the composition and decomposition:

- *Composition*: The correctness has to do with the fact that the composition of the modules resulted in something semantically correct. Two issues must be considered: (i) if the composition goal is to produce a fully integrated ontology, to be verified observing correctness if no information was lost after the composition of the modules, and (ii) if the goal of the piece is the specification of connections (links) found between the modules, will be the correctness defined as the fact that all the relevant routes have been implemented and no connection is implemented duplicated or can be inferred from others.

- *Decomposition*: Just as in composition, correctness verified to be observed if no information was lost in the process. It should be noted that the result of a consultation with a module must be equal to the same being done to the original ontology, or if the result of the composition of the modules obtained through the decomposition leads to the same original ontology.

References between composition and Modules: Even if a module is considered a sub-ontology independent, it is unlikely that a he, alone, is capable of fully responding to queries. In this case it is necessary that there is a management service ontologies that identifies the modules required to meet a particular query, to see that each one individually, integrate and synchronize the individual responses in order to compose a global response . According to the type of module, you need to establish a mechanism linking the different modules:

- Closed Modules: modules are not related to each other directly. The? links that unite are stored as metadata outside in a central repository or distributed.

- Open Modules: modules are connected directly to? one or more other via links, which represent the paths for additional information.

Conflicts and Overlapping Modules: During the process of composition of an ontology is natural to appear various types of heterogeneities

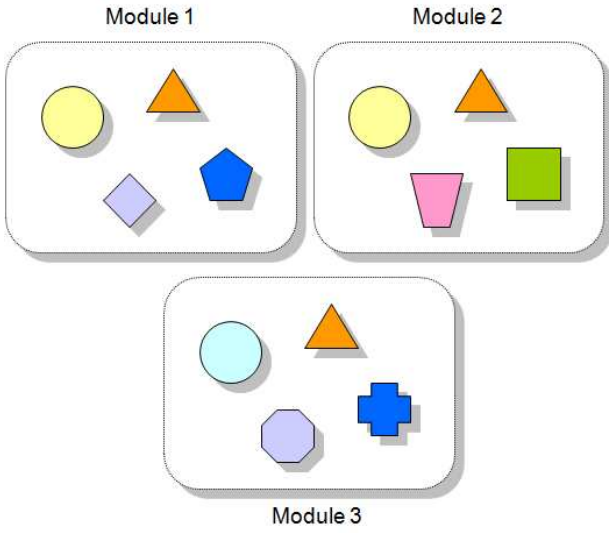

Figure 9: Different modules containing the same concepts.

caused by the different perceptions that the developers may have about the world. These heterogeneities ranges from the formalism used in the construction of the module (RDF, OWL, etc.) to its semantic content (for example, a module could describe a mesh of roads and another, a public transport system). Despite the fact that the modules can come from various sources (authors, locations, etc.), it is natural to believe that many modules provide conceptual intersections or overlapping, inclusive with the possibility of dual concepts (Figure 9).

Several types of conflicts can arise from the union of several modules. For example, we could have the concepts *developer*, *author* and *creator*, each one in a different module, representing someone who has a particular product. Other types of conflicts come from the fact that different modules can be concepts with different structures:

- *Structural Conflicts* occur when we have the same ideas, but with different internal structure. For example, a module could have the concept restaurant with an attribute evaluation, which values the field would be "bad", "reasonable", "good" and "excellent" to indicate its quality. Another module would also have a concept called Restaurant, and this would have an attribute indication, the field would be the values "yes", "no" and "with restrictions".

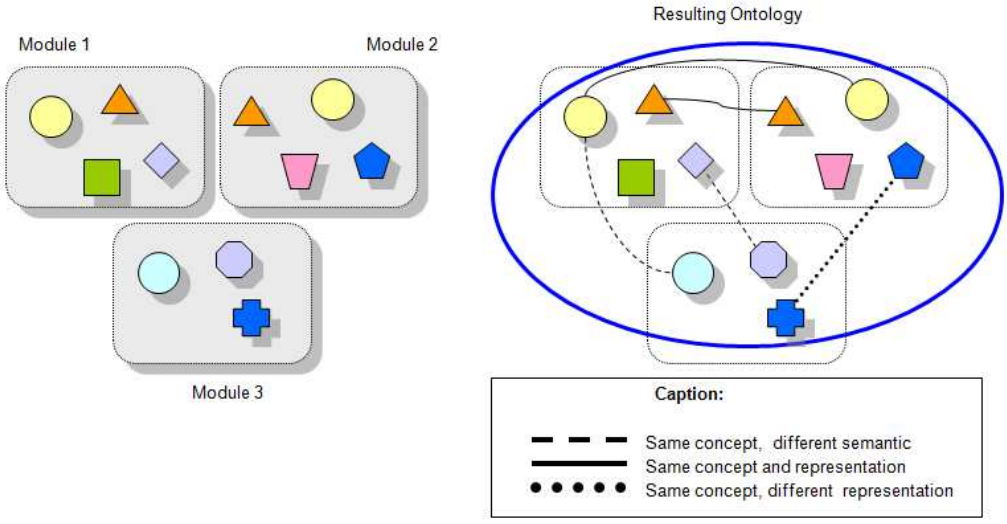

Figure 10: Conflicts in an ontology from the composition of several modules.

- *Semantic conflicts* occur when we have different classification schemes, based on understandings and / or different goals. For example, a module would have the concept Class with a connection to the concept pupil, where it would have the attribute ratings. Another module could have the concept with the Class attribute ratings.

Many studies have been developed in order to enable the accommodation of concepts from different sources into a single ontology [16] [12] [17] [18]. This process known as alignment is to eliminate the various conflicts arising in the composition of modules. The idea is that differences, whether structural or semantic, are parts of the natural process of building ontologies. Therefore, a methodology for construction of modular ontologies should consider steps and mechanisms for implementation and documentation which they consider as key alignment.

5.3 Agile Methods

In software engineering, agile methods are a response to traditional methodologies in order to turn the software development a flexible, simpler, cheaper, less error prone, better understood and with quickest deliveries [6] [53]. In a similar way, some works are being proposed in order to bring the agile

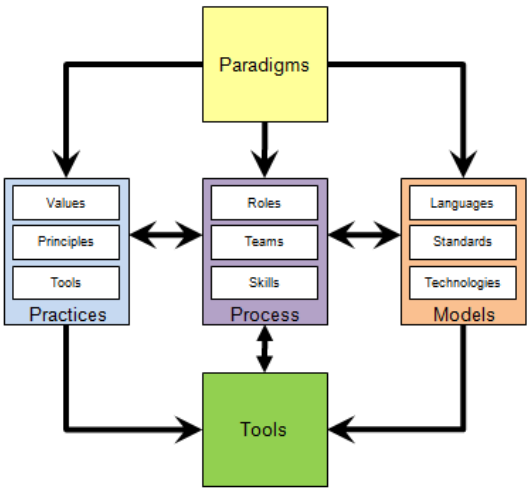

Figure 11: Main ideas of Agile Methodologies.

principles to the ontology development. In next subsections we will describe the following approaches RapidOWL and OntoAgile:

5.3.1 RapidOWL

RapidOWL is a lightweight methodology for collaborative Knowledge Engineering. It is based mainly on two precedents approaches: XP.K (eXtreme Programming of Knowledge-based systems) [39], and the Wiki idea, which established collaborative ideas to text editing [41].

RapidOWL methodology is grounded over a well defined structure already adopted by other agile methodologies. This structure is composed by the following: paradigms, process, people, models and tools.

Each one describes a particular aspect of development. How it can be seen in figure 11, paradigms influence the process, they lay the foundation for the models and they have to be internalized by people. Tools also are an important part of the approach since they implement the collaboration processes among the people involved.

Following one of principles of the agile philosophy which developers must delivers functionalities as fast as possible, according to the authors, one of the major advantages of this methodology is the fast delivery of RDF chunks

(Resource Description Framework) [54] statements. These chunks are produced by developers from the User Stories. In each iteration, initially it is created a user story which describes a determined aspect of domain being studied. Further, developers analyze each story and formalize using the RDF language. RapidOWL has the same set of values of eXtreme Programming (XP), namely Communication, Feedback, Simplicity and Courage [6]. In same way, it does not have any development cycle with well defined phases and milestones among them. Instead values are proposed in order to guide the development process throughout Principles and Practices as can be seen in figure 12. The motivation behind this is the simplicity and flexibility during the development. Also, besides that, neither every XP practice must combine with RapidOWL (and vice-versa) because of specific characteristics of knowledge engineering.

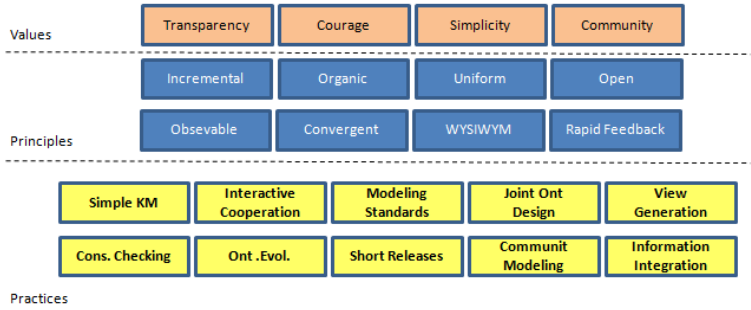

Figure 12: The building blocks of RapidOWL.

Despite of its promises, we identified some problems with this methodology. Firstly, just as others agile methodologies, RapidOWL scope is very limited and does not consider important activities of ontology engineering like reengineering and modular development. The first one is a necessary approach when developers have an ontology and need to enhance or adapt it to attend a new demand. The former approach is a tendency in ontology engineering since that each time more ontology engineering inspires on software engineering. One strength point in every agile methodology is communication. However, the informality or the lack of standards in RapidOWL during the activities can harm the communication among the participants and then harm the development process. Regarding to Knowledge Representation, RapidOWL focuses on implementation level [48] which limits the portability of the ontology produced. Also this fact prevents the usage of

ontological modules from existing ontologies and reuse of full top ontologies.

RDF is a datamodel for objects ("resources") and relations between them, provides a simple semantics for this datamodel, and these datamodels can be represented in an XML syntax [66]. However, this is not enough to semantically describes objects in ontologies. In order to attend this, OWL adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. Therefore we consider that is not a good design decision adopt RDF as being the knowledge representation language.

5.3.2 OntoAgile

This methodology is influenced by two major agile methodologies: XP [6] and Scrum [53]. The main idea of OntoAgile is to use only necessary documentation for development and concentrate the efforts on the client throughout short interactions in order to deliver products as fast as possible. Basically the OntoAgile process is composed of three interactive and incremental phases: (*i*) Planning Meeting, (*ii*) Construction and (*iii*) Delivery Meeting. At each iteraction one small part of ontology is produced.

In **Planning Meeting**, participants (*customer*, *leader* and *others members of development team*) develop user stories in order to identify what knowledge must be specified. The user stories will be form the Product Backlog, a repository of functional requirements, or in others words "what will be built" [72]. These stories will be organized in priority order and they will be the starting point of process.

Construction is divided in six subphases: (i) Concepts Acquisition (identify the main concepts), (ii) Existing Ontologies (look for the concepts in existing ontologies in order to discover their contexts), (iii) Relationship (connects the concepts discovered in 1 and 2), (iv) Codification (implements the ontology using a tool like Protégé or directly with a language like OWL), (v) Integration (puts together results from each iteration (called Sprint) with the ontology), and (vi) Consistency Checking (verifies concepts, relations and restrictions by means of a reasoner to check or SPARQL).

In the **Delivery Meeting phase**, stories are used to deliver that part of ontology being produced in current iteration. The same people that was present in first phase must be present in order to participate of this validation. The main problem observed in OntoAgile regards to the fact that it works just with small-size ontologies.

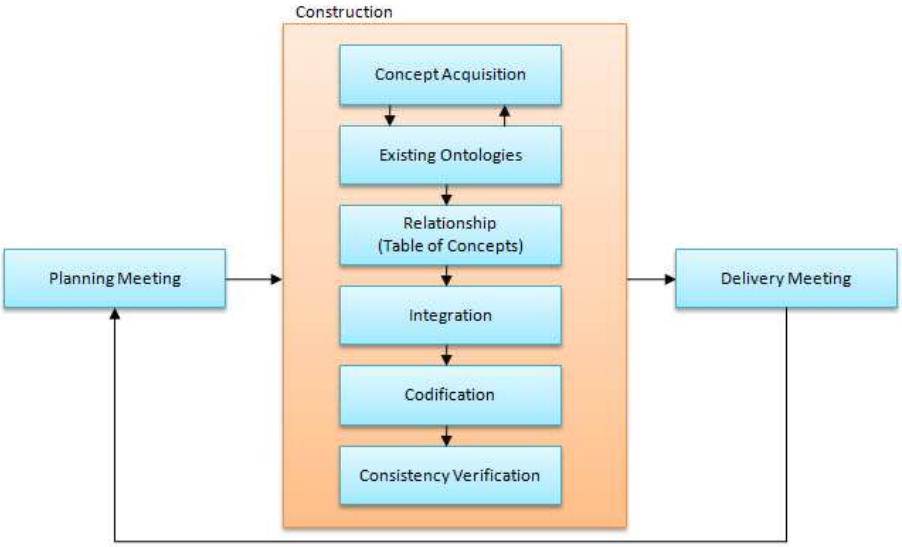

Figure 13: OntoAgile process.

6 Ontology Engineering and Software Engineering

Until this moment we have described some subjects of ontology engineering that are in state of art yet. We believe that the continuous development of these subjects has a great importance to the advancement of area. In next subsections we will compare each aspect regarded to that (and some others) in order to provoke discussions and enlighten the next steps of ontology engineering.

Despite the broadness of these subjects, because of space limitations, here we concentrate our efforts on those points that we consider more important to promote a agenda of investments and leverage the ontological engineering practices in next years.

6.1 Similarities and Differences

6.1.1 Preliminary Analysis

Since that OE and SE works with the conception of the same object (software) it is obvious to say that both of areas has many similarities.

The first difference concerns to the result being produced by each one. In one hand side, OE and SE deals with the development of systematic, well organized and tooled approaches to develop software. However, in other hand side, the nature of the software being produced is quite different and, further, the approaches must be slightly different. Despite of ontologies are software like any other in their essence, ontologies are not applications or, in others words, running software. Also, in same sense, software are not simply data like strings, structures or literals representing attributes or even data schemas of a data base. Ontologies are situated in middle of both of them (application and data).

One important difference between OE and SE regards to its origins. While SE was developed having as inspiration the traditional engineering, OE was developed as being an evolution of Knowledge Engineering.

6.1.2 Theoretical Analysis

How it was already discussed, SE has served as an inspiration for the development of OE. In this sense, we believe that it is important to compare the current development state of OE with the current state of SE in order to learn lessons from that. Here, in this work we propose an analytical comparison considering just those aspects approached just before. Others like Standardization and Documentation will be discussed more in future works. Theses comparisons summarizes many topics discussed in early sections.

Modularization has been discussed and studied for many years in SE and today it can be considered as a mature approach in order to accelerate the process. Although some initial efforts [4] [18] [11] [15] [26], modules are still in state of art but without perspectives to be adopted largely in short term. Current methodologies and tools don't accomplish modules and even still not exist a consensus about what exactly it is an ontology module [12].

Documentation is one of the deliverables of a software engineering development methodology. It guides the whole development, throughout the software life cycle, and aids to record the requirements, decision takings, problems found, limitations, roles, etc. Further, it plays a major importance to preserve the historic of an organization regarding to its projects. Traditional SE methodologies like RUP and Larman [52] make intensive use of documentation[2]. However, the large majority of OE methodologies does not present these same worries. For example, the methodology of Gruninger and Fox, TOVE and Cyc don't provide forms to document the development

[2]Agile methodologies like XP, Scrum and Crystal Clear considers that documentation harms the process flexibility and agility.

process else than the own ontology. Methontology describes one specific activity aimed to document the process but it does not accomplish the whole life cycle and neither define systematic ways to do it (for example, what are the documents? what are their formats? when to fulfill each one of these documents?).

Standardization regards to acceptance and consolidation of technologies, tools, approaches, languages, etc. by a considerable amount of developers or organizations. In SE exists many standardization efforts like the Object Management Group[3] (OMG), the Java Community Process[4] (JCP), and the SWEBOK[5] (Software Engineering Body of Knowledge). All these initiatives aims to promote the development, divulgement and maturing of their respective subjects, as well as to integrate the efforts under the same philosophy.

Under the economical perspective, **Quality Assurance** is one of the most important activities in SE because it promotes the necessary means to optimize the development processes and further to leverage the quality of products being developed. If in one hand side, SE has very matured standards and organizations aimed to promote the quality assurance like CMMI, ISO/IEC 15504, MPS-BR, etc., in other hand side OE has nothing similar until this moment.

The large adoption of **tools** is a clear signal that one determined area is getting into maturity. Since its beginning, SE community has called attention to this fact and in late 80's the "Computer Aided" tools generation experienced a exponential growth with the Computer Aided Software Engineering (CASE), Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) tools. More specifically to SE, exists a vast quantity of tools to aid the developers in whole life cycle like requirement editors (planning), domain modeling (analysis), coding (implementation), test and refactoring, etc. Despite the actual existence of a considerable quantity of tools like Protégé[6] (ontology modeler), Racer[7], Pellet[8] (OWL reasoner), SPARQL[9] (query language), and others, all of them with very good quality and acceptance by the ontology community, there is not environments dedicated to the whole ontology life cycle.

[3]http://www.omg.org.
[4]http://www.jcp.org.
[5]http://www.swebok.org.
[6]http://protege.stanford.edu
[7]http://www.sts.tu-harburg.de/ r.f.moeller/racer/.
[8]http://clarkparsia.com/pellet/
[9]http://www.w3.org/TR/rdf-sparql-query/.

Patterns are being adopted largely in SE, mainly the Design and Architectural Patterns. In OE, patterns are still in their very early steps. Some proposals were made [9] [74], but they still demands more efforts to consolidate them. In order to this happen, we believe that one task to be done is the support by means of tools.

The following table summarizes these topics:

Property	**SE**	**OE**
Modularization	Mature	State of art
Documentation	Mature	Ad-hoc
Standardization	Well developed	Inexistent
Tools	Mature	In evolution
Patterns	Mature	State of art
Quality Assurance	mature	inexistent
Methodology	large acceptance very good approaches	low acceptance and limited approaches
Heavy-weight methodologies	RUP, Larman,	Cyc, TOVE, Methontology
Light-weight methodologies	XP, Scrum, Crystal	RapidOWL, OntoAgile, XP.K

Table: summary of comparisons among SE and OE.

6.2 Mutual Contributions

OE and SE communities share a number of common topics. While SE has striving efforts towards a higher level of abstraction, giving more attention each time more to modeling in software development activities,the OE (and previously the Knowledge Engineering) has dedicated efforts to develop systematic and well disciplined processes like that ones existing in SE. How it was possible to see in previous sessions these two areas can benefit themselves in many ways, exchanging experiences and techniques.

Happel and Seerdorf [32] enumerates the problems existing in all phases of software life cycle and describes how the application of ontologies can solve each one. According to them, ontologies can improve, among others, (i) the communication abilities in requirement specification because of knowledge representation formalisms, (ii) the possibility to create richer models in analysis and design phases because of higher level abstractions, and (iii) the annotation of API elements with unambiguous concepts.

Ruiz and Hilera [55] surveys the using of ontologies in SE and proposes a very useful taxonomy to assist researches to identify, understand and choose

ontologies in software development projects, to identify what proposals of new methodologies or previous adaptations exist for the construction of ontology-driven software, and what types of software artifacts can be formed by or include ontologies.

Others works like presents ontologies developed aiming to software maintenance [1], software measurement [3], development environments [65] and quality assurance [2] [3] [42] [58].

In previous sections of this work, we have examined distinct types of inspirations from SE to OE. Approaches like that ones based on concepts like patterns, components, and agile methodologies can contribute positively to OE evolution in a near future. We believe that others approaches not cited here because it was not possible to find works related to like RAD (Rapid Application Development), distributed development, quality models etc. also can leverage the OE.

6.3 Perspectives

Analyzing these two areas since their beginnings it is possible to observe that they are converging in certain aspects. Today we have still a separating each area. However, we believe that these mutual contributions tend to grow each time more and such as semantic and ontologies will be make part of software development projects as the development of ontologies will be conducted like the software processes of today, in others words well documented, systematically and observing quality issues.

This scenario allow us to design perspectives around a **Semantic Software Engineering**, which the software development will be based on ontologies, and an Ontology Engineering based on methodologies more matures and better suited to market demands.

6.4 Challenges

Although the all increasingly growing efforts about OE there is much to do yet. We believe that the maturity of OE will come just when its community adopt methodologies, techniques and tools to develop its products similar to what happens in SE today. The nowadays demands for ontologies in semantic web as well as in traditional areas like database management systems, distributed computing, artificial intelligence etc. are the main motivations for the progress of OE.

As it was discussed, the use of modules in ontology development can leverage all the ontology process development. Many works call attention

to this, but no existing methodology deals with that yet.

7 Final Remarks and Future Works

The usage of methodologies in development of ontologies is a crucial issue for the area evolution and maturing. The growing adoption of ontologies in semantic web and in traditional computing areas applications demands systematic ways to develop ontologies and the necessity to improve its quality. In this paper we have presented a discussion about the convergence between software engineering and ontology engineering, mainly under the ontology development perspective. Thus, OE will benefit greatly from SE experiences since that there is many overlapping areas. Our strategy to develop the research that originated this work was to analyze how it was the SE progress along the time and outline the approaches being adopted that contributed to that and the main weaknesses that harmed the efforts being applied. This taxonomy will help us to design solutions to that problems.

In next steps, in order to achieve the aimed flexibility and productivity, we will concentrate our efforts to modular ontologies.

References

[1] Anquetil, N., Oliveira, K., *Software Maintenance Ontology*, In: Ontologies for Software Engineering and Software Technology, Springer, pp. 154-173, 2006.

[2] Abran et al., *Engineering the Ontology for the SWEBOK: Issues and Techniques*, in: Ontologies for Software Engineering and Software Technology, Springer, 2006.

[3] Bertoa M. F., Vallecillo, A., and Garcia, F., *An Ontology for Software Measurement*, in: Ontologies for Software Engineering and Software Technology, Springer, 2006.

[4] Bao, J. and Honavar, V. G, Divide and Conquer Semantic Web with Modular Ontologies - A Brief Review of Modular Ontology Language Formalisms. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[5] Bechhofer S., Harmelen F., Hendler J., Horrocks I., McGuinness D.L., Patel-Scheineider P.F., Stein l. A., OWL Web Ontology Language Ref-

erence. Avaliable in http://www.w3.org/TR/owl-ref/. Last access em 02 de Setembro de 2007.

[6] Beck, K. *Extreme Programming Explained: Embrace Change.*Addison Wesley, 2000.

[7] Berners-Lee, T., Hendler, J.,Lassila, O., *The Semantic Web*, in: Scientific American, pp. 24-30, May, 2001.

[8] Buchoffer, S. et al, Avaliable in http://www.ontoknowledge.org/oil/downl/oil-whitepaper.pdf, Last access in 12 october of 2007.

[9] Blomqvist, E., *State of the Art: Patterns in Ontology Engineering*, Research Report 04:8. School of Engineering, Jnkping University, December, 2004.

[10] Breitman, K., *Web Semântica: A Internet do Futuro*, Rio de Janeiro, Editora LTC, 2005.

[11] D'Aquin, M., Sabou M., and Motta E., *Modularization: a Key for the Dynamic Selection of Relevant Knowledge Components*. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[12] Camila Bezerra, Fred Freitas, Jérôme Euzenat, Antoine Zimmermann, *ModOnto: A tool for modularizing ontologies*, Second Workshop on Ontologies and Their Applications, Brazilian Symposium of Artificial Intelligence, Salvador, BA, Brazil, 2008.

[13] Crnkovick, I., *Component-Based Software Engineering*, 10th International Symposium, CBSE 2007, Heinz Schmidt, Ivica Crnkovic, George Heineman, Judith Stafford, Springer, LNCS Series, Vol. 4608, ISBN: 978-3-540-73550-2, 2007

[14] Devedzic, V., *Understanding Ontological Engineering*. Communications of the ACM, Vol. 45, pp 136-144, Abril, 2002.

[15] Doran, P., *Ontology Reuse via Ontology Modularization*. In Proceedings of KnowledgeWeb PhD Symposium 2006 (KWEPSY2006) Budva, Montenegro, 2006.

[16] Euzenat, J. et al, *State of the Art on Ontology Alignement*. Avaliable in http://www.aifb.uni- karlsruhe.de/WBS/meh/publications/euzenat04state.pdf. Last access em 12 de Outubro de 2007.

[17] Euzenat, J., and Shvaiko, P., *Ontology Matching*, Springer-Verlag, 2007.

[18] Euzenat, J., Zimmermann, A. and Freitas, F., *Alignment-based modules for encapsulating ontologies*, to appear in Bernardo Cuenca-Grau, Vasant Honavar, Anne Schlicht, Frank Wolter (Eds.): 2nd International Workshop on Modular Ontologies, WoMO 2007, Whistler, British Columbia Canada, October 28, 2007.

[19] Fernandez-Lopez M., Pazos A., Pazos J., *Building a Chemical Ontology Using Methontology and the Ontology Design Environment*, IEEE Inteligent Systems and their applications 4(1):37-46.

[20] Fernandez-Lopez M., Gomez-Perez A., Juristo N., *METHONTOLOGY: From Ontological Art Towards Ontological Engineering*, Spring Symposium on Ontological Engineering of AAAI, Stanford University, California, pp 25-34.

[21] Freitas, F.; Stuckenschmidt, Heiner; Volz, Raphael (Eds.). *Workshop on Ontologies and their Applications* - WONTO'2004 - Proceedings, 2004, LivroRapido, Brazil.

[22] Freitas, F., Stuckenschmidt, H., Noy, N., *Ontology Issues and Applications: Guest Editors Introduction*. 2005. Journal of the Brazilian Computer Society, Special Issue on Ontologias Issues and Applications, November 2005, Sociedade Brasileira de Computacao, Brazil.

[23] Freitas, F., and Euzenat, J., *EXMO: Project Report*, 2007.

[24] Gamma, E., Vlissides, J., Helm, R., Johson, R., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.

[25] Ghidini, C. and Serafini, L., Mapping properties of heterogeneous ontologies. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[26] Grau, B. C., A Logical Framework for Modularity of Ontologies, Avaliable in http://www.ijcai.org/papers07/Papers/IJCAI07-046.pdf. Last access in 06 de Junho de 2007.

[27] Goméz-Pérez, A., Juristo, N., Montes, C., Pazos, J., *Ingenieria Del Conocomiento: Diseno y Construccion de Sistemas Expertos*. Ceura, Madrid, Spain.

[28] Goméz-Pérez A, *Knowledge Sharing and Reuse*. In Liebowitz J (ed) Handbook of Expert Systems. CRC, Chapter 10, Boca Raton, Florida.

[29] Gomez-Pérez, A., Fernandez-Lopez, M., Corcho, O., *Ontological Enginering: with Examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*, Springer-Verlag, 2004.

[30] Gruber, T., *Toward Principles for the Design of Ontologies used for Knowledge Sharing*. International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers, Deventer, The Netherlands. 1993.

[31] Gruninger, M.,Fox M., *Methodology for the Design and Evaluation of Ontologies*. In Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp 6.1-6.10.

[32] Happel, H., Seedorf, S., *Applications of Ontologies in Software Engineering*.

[33] Harmelen, F., Reference description of the DAML+OIL (March 2001) ontology markup language, Avaliable in http://www.daml.org/2001/03/reference. Last access in 31 de maio de 2007.

[34] Stuckenschmidt, H., *Modularization of Ontologies*. WonderWeb Deliverable D21, Avaliable in http://wonderweb.semanticweb.org/deliverables/ D21.shtml, éltimo access em 06 de Junho de 2007.

[35] *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Computer Society. New York. IEEE Std 610.121990.

[36] *IEEE Standard for Developing Software Life Cycle Processes*. IEEE Computer Society. New York. IEEE Std 1074-1995.

[37] Jannink, J., Srinivasan, P., Verheijen, D., Wiederhold, G., *Encapsulation and Composition of Ontologies*, Proc. AAAI Workshop on Information Integration, AAAI Summer Conference, July 1998. Madison WI, USA.

[38] Jannink, J., Mitra, P., Neuhold, E., Srinivasan, P., Studer, R., Wiederhold, G., *An Algebra for Semantic Interoperation of Semistructured Data*, in 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99), Nov. 1999. Chicago, USA.

[39] H. Knublauch. *An Agile Development Methodology for Knowledge-Based Systems*. PhD thesis, University of Ulm, 2002.

[40] Kosciansky, A., Santos, M., *Qualidade de Software*, 2nd. edition, Editora Novatec.

[41] B. Leuf and W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley Professional,2001.

[42] Liao, L., Qu, Y., Hareton, K., *A Software Process Ontology and Its Application*, ISWC 2005, Workshop on Semantic Web Enabled Software Engineering, 2005.

[43] Loebe, F.,*Requirements for Logical Modules*. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[44] Luttich, K., Masolo, C. and Borgo, S., *Development of Modular Ontologies*, in Casl. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[45] Manola, F., Miller, E., *RDF Primer*, 2004. Avaliable in http://www.w3.org/TR/rdf-primer/. Last access in 24 of august of 2007.

[46] Melnik, S., Garcia-Molina, H., and Rahm, E.. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. 2002. In 18th International Conference on Data Engineering (ICDE-2002), IEEE Computing Society, San Jose, California.

[47] Mizoguchi, R., *A Step towards Ontological Engineering*, Procedings of the 12th National Conference on AI of JSAI, June, 1998, 24-31.

[48] Newell, A., *The Knowledge Level*, Artificial Intelligence Magazine, 18(1), pp 87-127, 1982.

[49] , OMG, *Object Constraint Language*, available in http://www.omg.org/spec/OCL/2.0, Last access in november 27, 2008.

[50] Pan, J , Serafini, L and Zhao, Y., *Semantic Import: An Approach for Partial Ontology Reuse*. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[51] Pressman, R., *Engenharia de Software*. Preentice-Hall, 2004.

[52] Larman, C., *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, Prentice-Hall, 3rd Edition, 2004.

[53] Mountain Goat, *Uma Introdução ao SCRUM*, Available in http://www.mountaingoatsoftware.com/system/hidden-asset/file/52/PortugueseScrum.pdf, last access in november 27, 2008.

[54] , Klyne, G., Carrol, J., *Resource Description Format: Concepts and Abstract Syntax*, availabe in http://www.w3.org/TR/rdf-concepts/, last access in november 27, 2008.

[55] Ruiz F., Hilera J., *Using Ontologies in Software Engineering and Technology*. In: Ontologies for Software Engineering and Software Technology. Springer. pp. 49-102. (2006)

[56] Schlicht, A., Stuckenschmidt, H., *Towards Structural Criteria for Ontology Modularization*. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[57] Simperl, E. P. B., Tempich C., *Ontology Engineering: a Reality Check*. In: 5th International Conference on Ontologies, Databases, and Applications of Semantics, ODBASE (2006).

[58] Soydan, G. H., Kokar, M. M., *An OWL Ontology for Representing the CMMI-SW Model*, International Workshop on Semantic Web Enabled Software Engineering (SWESE 2006), USA.

[59] Stuckenschmidt, H., *Implementing Modular Ontologies with Distributed Description Logics*. In: Proc. of the ISWC 2006 Workshop on Modular Ontologies. (2006)

[60] Stuckenschimidt, H., Klein, M., *Modularization of Ontologies*, In: Wonderweb: Ontology Infrastructure for the Semantic Web, IST Project 2001-33052, 2003. Avaliable in http://wonderweb.semanticweb.org/deliverables/documents /D21.pdf. Last Access in 5 de julho de 2007.

[61] Schilicht, A., Improving the Usability of Large Ontologies by Modularization, In Proceedings of Knowledge Web PhD Symposium 2007, Austria.

[62] S. Spaccapietra, A. Rector, A. Napoli, G. Stamou, G. Stoilos, H.Wolger, J. Pan, M. D'Aquin, and V. Tzouvaras, Report on modularization of ontologies, Technical report, Knowledge Web Deliverable D.2.1.3.1, (2005).

[63] Stuckenschimidt, H. Structure-based Partitioning of Large Concept Hierarchies. In Proceedings of the 3rd International Semantic Web Conference, Hiroshima, Japan (2004).

[64] Noy, N., *Representing Classes As Property Values on the Semantic Web. W3C Note*, http://www.w3.org/2001/sw/BestPractices/OEP/ClassesAsValues-20050405/ (2005).

[65] Oliveira, K., Villela, K., Rocha, A., Travassos, G., *Use of Ontologies in Software Development Environments*, In: Ontologies for Software Engineering and Software Technology, Springer, pp. 154-173, 2006.

[66] Bray, T., Paoli, J., Sperberg-McQueen, C., Maler, E., Yergeau, F., Cowan, J., *Extensible Markup Language (XML) 1.1*, (Second Edition), W3C Recommendation, 16 August 2006, edited in place 29 September 2006, available in http://www.w3.org/TR/2006/REC-xml11-20060816, last access in november 27, 2008.

[67] Volz, R., Oberle, D., Maedche, A. Towards a modularized Semantic Web In Proceedings of the ECAI-02 Workshop on Ontology and Semantic Interoperability Lyon, July 22, 2002, volume 64 of CEUR Workshop Proceedings. 2002.

[68] Uschold M, King M, Towards a Methodology for Building Ontologies. In: Skuce D (eds) IJCAI 95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, pp 6.1-6.10.

[69] Wache D et al., *State of the Art on the Scalability of Ontology-Based Technology*, Avaliable in http://knowledgewe.semanticweb.org/semanticportal/sewview/frame.jsp. Last access in 11 of October of 2007.

[70] Wang et al., Evaluating Formalisms for Modular Ontologies in Distributed Information Systems, In Massimo Marchiori and Jeff Z. Pan, Proceedings of The First International Conference on Web Reasoning and Rule Systems (RR2007), pp. 178-182. Springer, Innsbruck, Austria, June 2007.

[71] Waterman DA, *A Guide to Expert Systems*, Addison-Wesley, Boston, Massachussets, 1986.

[72] Wikipedia, *ISO 9000*, available in http://en.wikipedia.org/wiki/ISO-9000, last access in 27 of november, 2008.

[73] Wiederhold, Gio, *An Algebra for Ontology Composition*. Proceedings of 1994 Monterey Workshop on Formal Methods, Sept 1994, U.S. Naval Postgraduate School, pp 56-61. Monterey CA, USA.

[74] Gangemi, A., *Ontology Design Patterns for Semantic Web Content*. In Motta E. and Gil Y., Proceedings of the Fourth International Semantic Web Conference, 2005

[75] Auer, S. and Herre, H., *Rapidowl - an agile knowledge engineering methodology*. In Irina Virbitskaite and Andrei Voronkov, editors, Ershov Memorial Conference, volume 4378 of Lecture Notes in Computer Science,pp 424430. Springer, 2006

[76] Herre, *Rapidowl - An Agile Knowledge Engineering Methodology*. In Irina Virbitskaite and Andrei Voronkov, editors, Ershov Memorial Conference, volume 4378 of Lecture Notes in Computer Science, pp 424-430, 2006.