Deriving Ontologies Using Multi-agent Systems

VICTORIA IORDAN, ANTOANELA NAAJI *, ALEXANDRU CICORTAS
Department of Computer Science
West University of Timisoara, * "Vasile Goldis" Western University of Arad
Blvd. V. Parvan 4, Timisoara 300223, Blvd. Revolutiei 94-96, Arad 310025
ROMANIA
iordan@info.uvt.ro, anaaji@uvvg.ro , cico@info.uvt.ro

Abstract: - The complex systems are designed using multi-agent concepts. Agent interaction is complex and requires appropriate models for a communication and cooperation. Also the interaction between the users and the system agents must be done in an efficient way. One of the basic conditions is that to use a convenient "language", a common way of understanding. The ontology is the appropriate concept that allows doing it. The operations on the ontologies cover many of such requirements. Due to the complexity of systems interaction that has an impact on the different ontologies used in them. Our model tries to define a specific operation deriving an ontology form another one. The competence descriptions in education are given as an application. The research for this paper has been partial supported by the project PN II 91-047/2007.

Key-Words: - Ontology, competence description, multi-agent systems

1 Introduction

Ontologies are intended for knowledge representation, sharing, management, modeling, engineering and education among others. In [37] were given software engineering concepts, ideas and knowledge, software development methodologies, tools and techniques into ontologies and used them as a basis for classifying the concepts in communication and enabling knowledge sharing.

In any domain can be used two ontologies: generic ontology and application-specific ontology. Generic ontology is a set of domain terms including the vocabulary, the semantic interconnections and some simple rules of inference and logic for software development. Application-specific ontology is an explicit specification of domain problem for a system from the domain. This ontology can be used for communication between the system components, persons that are implied in system functionality and the agent which are used in the modeled systems. The ontological representation of domain problem not only represents the commonly agreed knowledge but also provides detailed relationships (descriptions) between the concepts and specific features of the domain problem. The application-specific ontology of domain problem can also be customized.

The artificial intelligence (AI) shows that knowledge is critical for intelligent systems. In many cases, better knowledge can be more important for solving a task than better algorithms. To have truly intelligent systems, knowledge needs to be captured, processed, reused, and communicated. Ontologies support all these tasks.

One of the important motivations for using ontologies is also the design and the implementation of software agents [34].

From a practical point of view, a given software ontology establishes the content of messages exchanged among agents and provides facilities to validate them. Moreover, ontologies are a good starting point for defining interaction protocols which are the most common way to define a structured communication among two entities.

The paper is organized as follows. In the second Section the basics of the ontologies are discussed in order to motivate our intentions. The need for an ontology framework is also grounded. The third Section treats the specificities of ontology modeling and design. The fourth Section shortly reviews the ontology representation languages. As an example in the fifth Section a software engineering ontology platform implementation is given. It resembles with our model. The sixth Section details our model in that the deriving operation is presented, in the context of a multi-agent system. The last Section prefigures the future works.

2 Ontologies basics

From a short overview and role of ontologies were presented in [26], here are reasons why

understanding, using and manipulating ontologies can bring practical benefit:

- depending on their degree of formalism (an important dimension), ontologies help make explicit the scope, definition and language and meaning (semantics) of a given domain or world view;
- may provide the power to generalize about their domains;
- if hierarchically structured in part (and not all are), can provide the power of inheritance;
- provide guidance for how to correctly place information in relation to other information in that domain;
- may provide the basis to reason or infer over its domain (again as a function of its formalism);
- can provide a more effective basis for information extraction or content clustering;
- again depending on their formalism, may be a source of structure and controlled vocabularies helpful for disambiguating context; they can inform and provide structure to the lexicons in particular domains;
    - can provide guiding structure for browsing or discovery within a domain, and
    - can help relate and place other ontologies or world views in relation to one another; in other words, ontologies can organize ontologies from the most specific to the most abstract.

The expressiveness of the ontology is given by its structure and formalism that allow classifying the ontologies.

The expressiveness is the mean by which ontology can describe domain semantics. Structure can be defined as the degree of organization or hierarchical extent of the ontology. The granularity is the level of detail in the ontology.

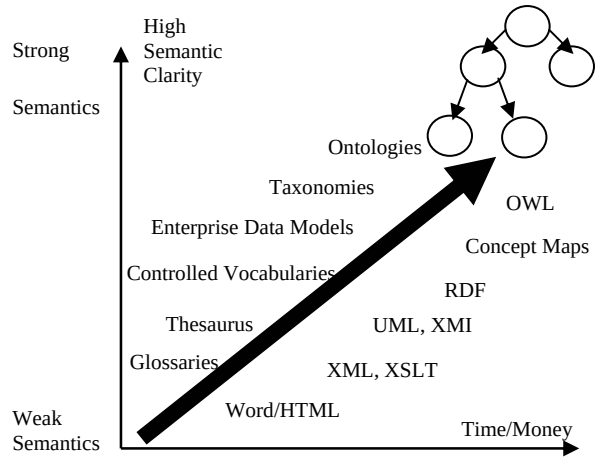

Fig. 1: Evolution of Semantic Clarity

In the Figure 1 is illustrated the increased semantic precision.

Based on the following ontology definitions some comments that characterize the conceptualization will be done in the following:

1. Ontology is a term in philosophy and its meaning is theory of existence.
2. Ontology is an explicit specification of conceptualization.
3. Ontology is a body of knowledge describing some domain, typically common sense knowledge domain.

The first definition is the meaning in philosophy; however it has many implications for the AI purposes. The second definition is generally accepted as a definition of what ontology is for the AI community. The last third definition views ontology as an inner body of knowledge, not as the way to describe the knowledge.

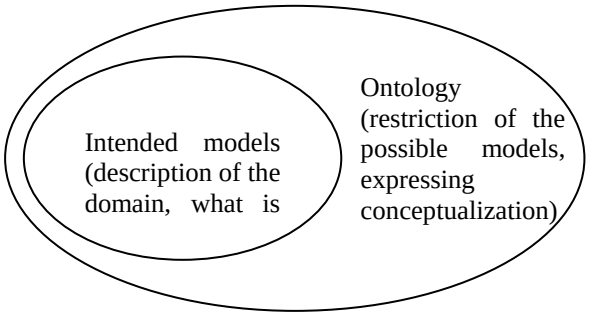

Fig. 2: Possible models expressible in the ontology language

2.1. Conceptualization

The second definition of ontology mentioned above, as explicit specification of conceptualization was done in [10], [11]. The exact meaning depends on the understanding of the terms specification and conceptualization. Explicit specification of conceptualization means that ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set of concept definitions, but more general ontology specification diagram.

The ontology can be defined also as an explicit specification of conceptualization. Ontologies capture the structure of the domain, i.e. conceptualization. This includes the model of the domain with possible restrictions. The conceptualization describes knowledge about the domain, not about the particular state of affairs in

the domain. In other words, the conceptualization is not changing, or is changing very rarely. Ontology is then specification of this conceptualization - the conceptualization is specified by using particular modeling language and particular terms. Formal specification is required in order to be able to process ontologies and operate on ontologies automatically.

Ontology describes a domain, while a knowledge base that is based on ontology describes particular state of domain.

A conceptualization can be defined as an intentional semantic structure that encodes implicit knowledge constraining the structure of a piece of a domain.

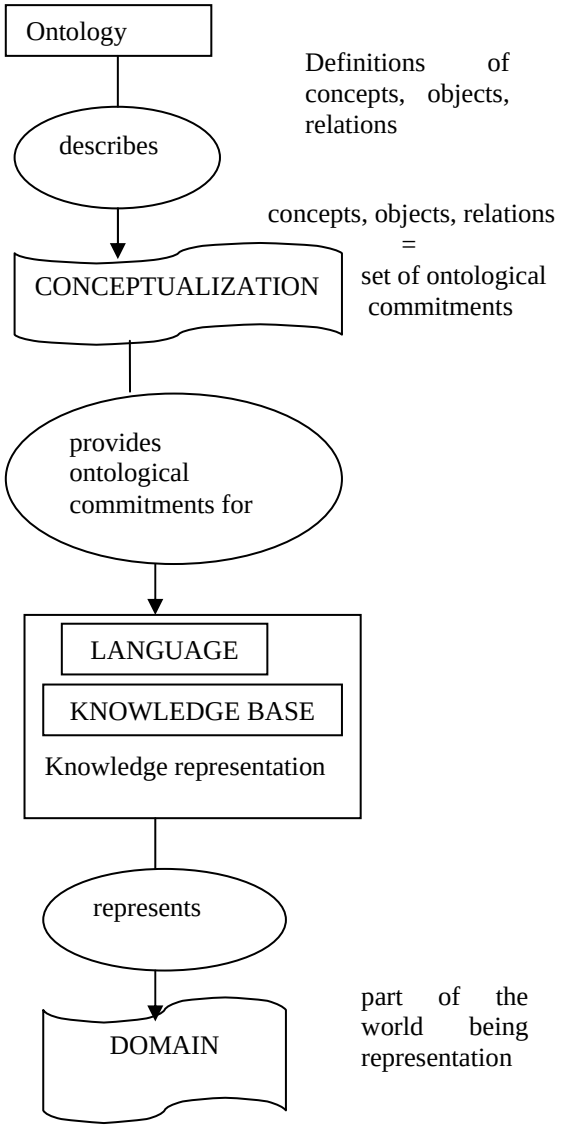

Fig.3: The relation between conceptualizations, ontologies, knowledge representations and domains.

Ontology is a (partial) specification of this structure, i.e., it is usually a logical theory that expresses the conceptualization explicitly in some language. Conceptualization is language independent, while ontology is language dependent. The use can be illustrated in the Fig.3 - it shows how an ontology restricts (i.e., defines) possible use of constructs used in the description of the domain.

Notice that ontology does not have to express all the possible constraints - the level of details in conceptualization depends on the requirements of the intended application and expressing conceptualization in ontology in addition depends on the used ontology language.

The Fig. 3 illustrates the relations between conceptualizations, ontologeis and knowledge representation.

2.2 Ontology Framework

Generally speaking:

- There are at least two important word senses for ontology: ontology as a field of study ontology (philosophy) and ontology as a technology for computer and information scientists. We are talking about the second sense of the word, ontology (computer science);
- Ontology could refer to either a piece of information that can be talked about objectively, communicated in digital media, and shared without loss of information among a community; or a set of ideas, concepts, abstractions, or other entities that are not the same as the representations or descriptions of them. We propose that we limit our discussions to the first sense: ontology as an objective form (the other sense is called a conceptualization that was treated above);
- In the context of computer and information sciences, ontology is a specification of a conceptualization. That is, it specifies the concepts, ideas, relations, abstractions, and so forth in an objective form. The intent is to clarify the meaning, enabling shared understanding.
- Ontology provides a specification of a conceptualization by defining a representational vocabulary (a set of terms that can be used to represent the domain) together with constraints on their meaningful use. The representational vocabulary may include concepts or categories, relations, properties, or other primitives for representing knowledge. The content of the specification includes:

  - identification of the fundamental categories in the domain;
  - identification of the ways in which members of the categories are related to each other;
  - constraints on the ways in which the relationships can be used.
- Although ontology defines vocabulary for representing a domain, it is not a specification of form. That is, it does not prescribe the form in which knowledge is represented, stored, communicated, or reasoned about. This is how ontology differs from a data model, and is why ontology is not defined by its form but by its role: to enable sharing, reuse, and application of knowledge.
- It does not matter to this definition whether ontology is formally equivalent to a logical theory, whether there is a formal difference between ontology and a knowledge base, or whether ontology is only definitional or also contains axiomatic constraints. In fact, it is not necessary that the ontology is represented in any kind of logical formalism. Many highly successful specifications (e.g., the HTTP standard) are given only in natural language, yet can be enforced with machine-understandable tests and examples. What matters is that its purpose is to specify a conceptualization, in what ever representational form is appropriate;

For the purposes of the framework is defined what we mean by the term ontology (computer science). Incorporating the distinctions introduced above, we have adapted the definition of ontology given, to this:

Ontology (definition), for computer and information sciences, is a specification of a conceptualization, which is the set of ideas, concepts, relationships, or other abstractions that comprise a domain of modeling or discourse. Ontology defines a representational vocabulary for the conceptualization, and specifies constraints on the meaningful use of this vocabulary, so that facts about the domain can be shared, communicated, and reasoned about.

2.3 Kinds of Ontologies

Ontologies can vary on several important dimensions. We propose a set of dimensions that can be used to distinguish among different approaches. There are two kinds of dimensions:

- semantic, which shows how ontology specifies the meaning of its representational vocabulary and
- pragmatic, which shows the purpose and context in which the ontology is designed and used.

Semantic Dimension includes:

- level of structure: This is akin to the notion of structured and unstructured data in computer science. An ontology that specifies formally defined concepts such as mathematical abstractions is high in structure, while an ontology that specifies very general concepts such as document and hyperlink is low in structure. Many ontologies are semi structured, containing a mix of formal and informal definitions of concepts and relationships.
- Expressiveness of the language or framework used: Although ontology is not a definition of form (e.g., the syntax of a language), ontology defines its vocabulary in some representational form. Ontologies differ in the expressive power of the language used in the specification. Some conceptualizations require a highly expressive language to define the concepts, where others can be specified with a less expressive language. This is related to the level of structure dimension. A highly structured and formal ontology might require a language capable of stating logical or mathematical constraints, whereas an informal ontology may be expressed only as a list of terms and definitions in a natural language such as English. Furthermore, the language used for stating logical or mathematical constraints can vary in expressivity.
- Representational granularity: While expressiveness is a characteristic of the language in which ontology is given, granularity is a property of the content of ontology itself. Coarse granularity ontology is specified using only very general representational primitives, such as concepts and subsumption in taxonomy, where fine granularity ontology specifies much more detail about the properties of concepts and how they can relate to each other.

Pragmatic Dimension contains:

- Intended use: The intended use may be to share knowledge bases, to enable communication among software agents, to help integrate disparate data sets, to represent a natural language vocabulary, to help provide knowledge-enhanced search, to provide a

starting-point for building knowledge systems, to provide a conceptual framework for indexing content, etc. The intended use often means that there is some application that is envisioned for which the ontology is being developed.

- Role of automated reasoning: Automated reasoning can range from simple to complex. Simple automated reasoning can mean machine semantic interpretability of the content, which only requires that the language that the content is modeled in, is a logic or that a special interpreter/inference engine has been constructed that knows how to interpret the content. The former approach logic) is a principled or standards-based approach; the latter (construction of a special interpreter/inference engine) is an ad hoc and often proprietary approach. In the simple automated reasoning case, the machine may be able to make inferences such as that the subclass relation means that properties defined at the parent class should be inherited down to the children classes; this is the property of transitivity. More complex automated reasoning is usually expressed as deductive rules, i.e., inference rules or expressions that combine information from across the ontology that characterize dependencies much like if-then-else statements in programming languages or business rules that try to characterize things that have to hold in an enterprise but which can't typically be expressed in relational databases or object models. Complex automated reasoning requires that the content be modeled in a logic based language simply because notions like 'validly concludes' or 'X is consistent with Y' are not expressible generally in ad hoc implementations.
- Descriptive vs. prescriptive: The content describe, i.e., characterize the entities and relations among the entities, as a user or an expert might characterize those objects. Or the content prescribe, i.e., mandate the way that those entities and their relationships are characterized. Descriptive often takes a looser notion of characterization, perhaps allowing arbitrary objects into the model, which might not exist in the real world but which are significant conceptual items for the given user community. Prescriptive often takes a stricter notion of characterization, stating that only objects which actually exist or that represent natural kinds or types of things in the real world should be represented in the content of the engineering model.
- Design methodology: how the methodology is employed in the construction of the ontology. Possible ranges of methodology include: bottom-up, top-down. A bottom-up (sometimes called empirical) methodology places strong emphasis on: either solely analyzing the data sources so that the resulting ontology covers their semantics, or enabling arbitrary persons to characterize their content as they personally see fit, using terminology or metadata and whatever structuring relations (or not) that they desire to use, with perhaps an auxiliary notion or assumption that in by doing so, patterns of characterizations may emerge or be preferred by a large group or community of persons.

3 Specific Design

Before development, a designer has to have a model of the conceptual structure of the domain i.e. the ontology as well as an understanding of the structure of information describing instances of these concepts and their relationships [3]. A critical aspect of modeling and designing ontology is lack of graphical notation [8]. The UML can be used to model ontology. UML object diagrams can be interpreted as declarative representations of knowledge. Instance information can be conveyed as a UML object diagram that shows the values of object attributes and the link i.e. instances of associations that exist between objects. There are benefits for using the same paradigm for modeling ontologies and knowledge. Even standard UML cannot express advanced ontology features such as restrictions, cannot easily conclude whether the same property was attached to more than one class, and cannot create a hierarchy of properties [2].

It is a kind of agile modeling method for ontology design and the main aim of this use of UML notation is simply to create a graphical representation of ontologies to make them easier to understand. This use of UML notation to model the underlying ontology should be distinguished from its use in software development to model the application domain model. During the ontology modeling and designing the concepts in object-oriented: the classes and the use case-diagrams must be defined.

4 Ontology Representation Languages

Ontologies are used to capture knowledge in some domains of interest. Ontology describes the

concepts in the domain and also the relationships that hold among those concepts. Different ontology languages provide different facilities [12].

There are many ontology representation languages for creating ontology including Knowledge Interchange Format (KIF) [9], Simple HTML Ontology Extension (SHOE) [18], ISO standard for describing knowledge structures (Topic Maps) [19], Ontology Exchange Language (XOL) [17], Ontology Markup Language (OML) [18], Ontology Inference Layer (OIL [13], DAML+OIL [14]) and Web Ontology Language (OWL)[22].

The XML is used in many languages as support [35], [36].

Between the most current developments in standard ontology languages is OWL [22].

Concerning the considerations and the requirements on ontology languages based on the fact that RDF is a main ontology language, the following refers to the RDF and RDFS [28], [29].

For shared meaning different data sources should be able to commit to the same ontology.

Data sources that commit to the same ontology explicitly agree to use the same identifiers with the same meanings. An organization must be able to create an ontology which extends an existing ontology and adds any desired identifiers and definitions.

Ontology may change during its lifetime.

Different ontologies may model the same concepts in different ways. The language should provide primitives for relating different representations, thus allowing data to be converted to different ontologies. For that, any use case in which data from different providers with different terminologies must be integrated [31]. Different ontologies or data sources may be contradictory. RDF and RDFS do not allow inconsistencies to be expressed.

In order to prevent agents from combining incompatible data or from taking consistent data and evolving it into an inconsistent state, it is important that inconsistencies can be detected automatically.

An ontology language must have the ability to express the most important kinds of knowledge.

Expressivity determines what can be said in the language, and thus determines its inferential power and what reasoning capabilities should be expected in systems that fully implement it.

The design goals (and use cases) motivate a number of requirements for a web ontology language. For web design for example, some requirements are:

- ontologies as distinct resources. Ontologies must be resources that have their own unique identifiers, such as a URI reference;
- unambiguous concept referencing with URIs. Two concepts in different ontologies must have distinct absolute identifiers (although they may have identical relative identifiers). It must be possible to uniquely identify a concept in an ontology using a URI reference;
- explicit ontology extension. Ontologies must be able to explicitly extend other ontologies in order to reuse concepts while adding new classes and properties. Ontology extension must be a transitive relation; if ontology A extends ontology B, and ontology B extends ontology C, then ontology A implicitly extends ontology C as well;
- commitment to ontologies. Resources must be able to explicitly commit to specific ontologies, indicating precisely which set of definitions and assumptions are made;
- ontology metadata. It must be possible to provide meta-data for each ontology.

Some considerations concerning the OWL are giving in the following (OWL ontology consists of Individuals, Properties and Classes).

Individuals represent objects in the domain of interest. Individuals are also known as instances. It can be referred to as being instances of classes or concepts. Properties are relationships between two things i.e. a concept/individual links to a concept/individual known as object property or a concept/individual link to an XML schema data type value or an rdf literal known as data type property.

The inverse of has Relationship is isRelatedTo. Properties can be limited to having a single value; to being functional or multiple values i.e., to being non-functional. Also, they can be either transitive or symmetric. Properties are also known as roles in description logics, and attributes in UML and other object-oriented notions.

Classes are a concrete representation of concepts interpreted as sets that contain individual(s). Individuals may belong to more than one class. Classes may be constructed in a superclass-subclass hierarchy, which is also known as taxonomy. Subclasses are subsumed by their superclasses. For example, in object-oriented design, association dependency and generalization are all a relationship between object classes.

5 Software Engineering Ontology Platform Implementation

The following are based on [37] and in some way, our model is similar with Ontology Classes.
The class hierarchy from [37] is shown in Fig. 4. The class owl:Thing is the class that represents the set containing all individuals. Thereby, all classes are subclasses of owl:Thing. OWL classes are assumed to overlap. Therefore, one cannot assume that an individual is only a member of a particular class; it can be a member of more than one class. In order to separate a group of classes, we must make them disjoint from each other.

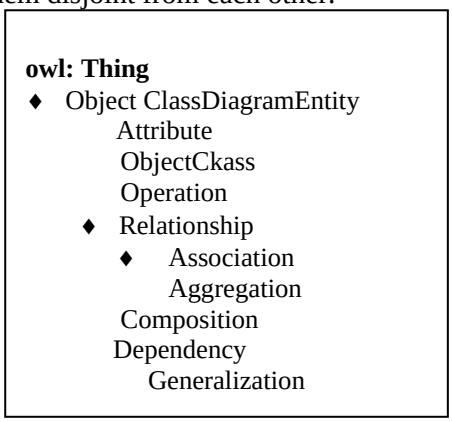

Fig. 4: Class hierarchy shown concept of object class diagram in the object-oriented design

This assures that an individual who has been asserted to be a member of one of the classes in the group cannot be a member of any other class in that group. For example, Association, Dependency, and Generalization have been disjointed from one another. This means that there is no chance for an individual to be an association and dependency and generalization relationship. Likewise, Attribute, ObjectClass, Operation, and Relationship have been disjointed also, because individual such as an Attribute cannot be individual of either ObjectClass, Operation, or Relationship in the group of ObjectClassDiagramEntity.

Ontology has three types of properties: Object properties, Datatype properties, and Annotation properties. Object properties link one class or individual to another; Datatype properties link a class or an individual to an XML schema datatype value or an rdf literal and Annotation properties are used to add information to classes, individuals and object and datatype properties.
The meaning of properties is enriched through the use of property characteristic. The various characteristics show that properties are functional, inverse functional, transitive and symmetric.

So, if a property is functional, there will be at most one individual that is linked to the individual through the property. In the case in which the property links individual x to individual y then its inverse property will link individual y to individual x. If property x is transitive and the property x relates individual a to individual b and also individual b to individual c, then it can be inferred that individual a is related to individual c via property x. The last characteristic shows that, if property x is symmetric and the property links individual a to individual b then individual b is also linked to individual a via property x.
In [37] was illustrated how software engineering ontologies facilitate communication and allow knowledge sharing. Figure 5 shows software engineering knowledge base allowing knowledge sharing.

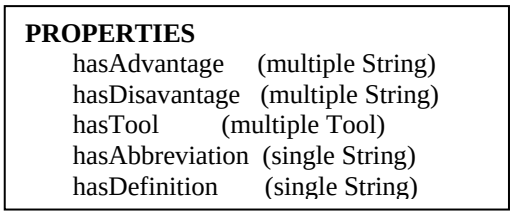

Fig. 5: Functional and non-functional properties

Any concept related to software engineering can be fetched showing the concept's details e.g. its definition, abbreviation, principles, advantage, disadvantage, output, template, tool, involved concept, etc. The user will see all details of the relevant concepts which are arranged in hierarchy. This can be done by utilizing generic ontology and software agent to go through the ontology. Furthermore, from generic ontology software agent will be able to extract information e.g. from templates stored in the ontology as instances and create a handle book for the project.

By utilizing application-specific ontologies and individuals/instances of a particular project data, it can convert the plain text into a UML-like diagram that helps communication among the team members within the same project and provides consistent understanding. Software agents can be utilized to extract information from ontology described in OWL. To do so, the software agent consults, for example, the object class ontology. The ontology shows how class is formed in the class diagram; and each class contains a name, attributes, and operations; and relationships between the classes. Therefore, the software agent dynamically acts to retrieve involved class names, involved class attributes, involved class operations, and involved relationships to draw a class diagram.

6 Our model

6.1 Preliminaries

Reasoning in ontologies and knowledge bases is one of the reasons why a specification needs to be formal one. By reasoning we mean deriving facts that are not expressed in ontology or in knowledge base explicitly. All of the formalisms were created with the outlook of automatic processing, but due their properties such as decidability or computational complexity or even due to the level of formality it is not always possible.

Description logics are created with the focus on tractable reasoning. A few examples of tasks required from reasoner are as follows:

- satisfiability of a concept - determine whether a description of the concept is not contradictory;
- subsumption of concepts - determine whether concept C subsumes concept D, i.e., whether description of C is more general than the description of D;
- consistency;
- check an individual - check whether the individual is an instance of a concept;
- retrieval of individuals - find all individuals that are instances of a concept;
- realization of an individual.

6.2 Operations on Ontologies

It is possible that one application uses multiple ontologies [11], [32], [24], [25] especially when using modular design of ontologies or when we need to integrate with systems that use other ontologies. In this case, some operations on ontologies may be needed in order to work with all of them. The terminology in this areas is still not stable and different authors may use these terms in a bit shifted meaning, and so the terms may overlap, however, all of these operations are important for maintenance and integration of ontologies:

- merge of ontologies means creation of a new ontology by linking up the existing ones;
- mapping from one ontology to another one is expressing of the way how to translate statements from ontology to the other one;
- alignment is a process of mapping between ontologies in both directions whereas it is possible to modify original ontologies so that suitable translation exists (i.e., without losing information during mapping);
- refinement is mapping from ontology A to another ontology B so that every concept of ontology A has equivalent in ontology B, however primitive concepts from ontology A may correspond to non-primitive (defined) concepts of ontology

Refinement defines partial ordering of ontologies;

- unification is aligning all of the concepts and relations in ontologies so that inference in one ontology can be mapped to inference in other ontology and vice versa;
- integration is a process of looking for the same parts of two different ontologies A and B while developing new ontology C - Integration is a process of looking for the same parts of two different ontologies A and B while developing new ontology C;
- inheritance means that ontology A inherits everything from ontology B. It inherits all concepts, relations and restrictions or axioms and there is no inconsistency introduced by additional knowledge contained in ontology A.

Relations between ontologies [6], [30] are: extension, identical, equivalent, strongly-translatable, weakly-translatable and approx-translatable.

The purpose of authoring ontologies is also reusing of knowledge. Once ontology is created for a domain, it should be (at least to some degree) reusable for other applications in the same domain. To simplify both ontology development and reuse, modular design is beneficial. The modular design uses inheritance of ontologies: upper ontologies describe general knowledge and application ontologies describe knowledge for a particular application, as illustrated in the figure below.

Modularization of ontologies depending on the scope and partial ordering defined by inheritance are illustrated [24] in the Figure 6.

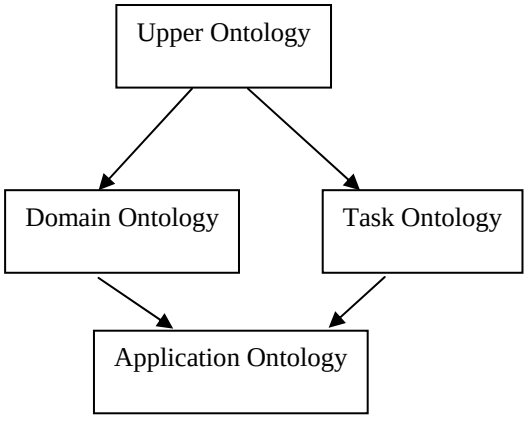

Fig. 6: Modularization of ontologies

6.3 Communication between Agents

The ontology provides the vocabulary from which to construct queries [11], and the semantics so that two agents can agree on what makes sense in a given vocabulary. In this case, the agents can agree about which quantity expressions and term expressions denote quantities and units, and when they are given as arguments to the quantity. These agreements establish a basis for agent discourse. Separating the core ontology about quantities and units from the specific conventions for systems of units minimizes the ontological commitment of participating agents. While they all need to commit to the core theory, they can commit to differing standards of measure. Since commitment to an ontology does not require completeness of inference, agents can understand the conditions under which a value exists (e.g., a magnitude in some unknown unit) without knowing how to compute the value. As it was previously expressed: the ontology is important for the purpose of enabling knowledge sharing and reuse [25] and, an ontology is in this context a specification used for making ontological commitments.

Practically, an ontological commitment is an agreement to use a vocabulary (i.e., ask queries and make assertions) in a way that is consistent (but not complete) with respect to the theory specified by an ontology. Agents then commit to ontologies and ontologies are designed so that the knowledge can be shared among these agents.

The representation of a body of knowledge (knowledge base) is based on the specification of conceptualization. When the knowledge of a domain is represented in a declarative formalism, the set of objects that can be represented (the universe of discourse). This set of objects and the describable relationships among them are reflected in the representational vocabulary with which a knowledge-based program represents knowledge. Thus, in the context of AI, we can describe the ontology of a program by defining a set of representational terms. In such an ontology, definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions, or other objects) with descriptions of what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally it can be said that an ontology is a statement of a logical theory.

Knowledge sharing and exchange is particularly important in multi-agent systems (MAS). An agent is usually described as a persistent entity with some degree of independence or autonomy that carries out some set of operations depending on what he perceives. An agent usually contains some level of intelligence, so it has to have some knowledge about its goals and desires. The whole multi-agent system is created to be capable of reaching goals that are difficult to achieve by an individual agent or a monolithic system. In multi- agent systems, an agent usually cooperates with other agents, so it should have some social and communicative abilities.

In order to communicate, agents must be able to:

- deliver and receive messages - at this physical level,
- parse the messages - at the syntactic level, and
- understand the messages - at the semantic level.

For multi-agent systems the first physical level as well as the second syntactic level is well standardized by the Foundation for Intelligent Physical Agents (FIPA), for example by agent management specification and agent communication language specification. As for the third level, semantics, standard exists that describe the content languages and that describe usage of ontologies.

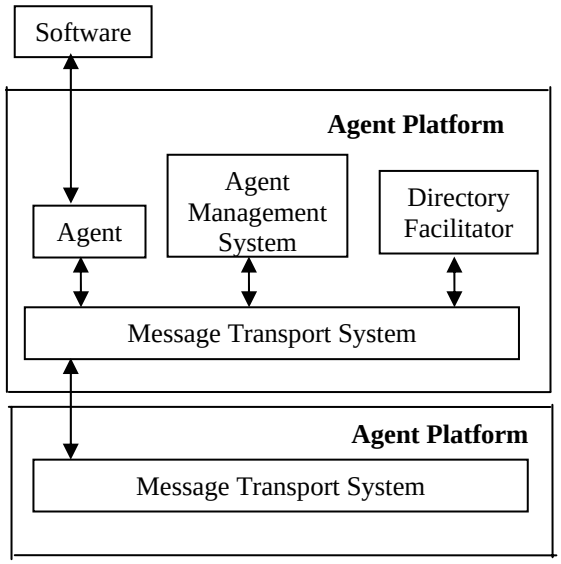

Fig. 7: Communication between agents

The Figure 7 illustrates by arrows the possible communication between agents.

6.4 Ontologies for Agents

Each agent has to know something about a domain it is working in and also has to communicate with other agents. An agent is able to communicate only about facts that can be expressed in some ontology.

This ontology must be agreed and understood among the agent community (or at least among its part) in order to enable each agent to understand messages from other agents.
Unfortunately, the ontology used for communication between agents is not always expressed explicitly - the constructs used to exchange information may be hardcoded in agents, and no explicit form describing the ontology may be available. The assumptions on the meaning of the vocabulary are implicitly embedded in agents, i.e., in software programs representing agents. In this case it is harder to integrate such agents with other agents that were not programmed to communicate together.
The need to obtain form some ontology another one that can be inferred by appropriate tools is one of the goals in our model. As was seen the operations on the ontologies [24] does not give a such and direct capability.
Not all of operations can be made for all ontologies. In general, these are very difficult tasks that are in general not solvable automatically in some cases because of undecidability when using very expressive logical languages or because of insufficient specification of an ontology that is not enough to find similarities with another ontology.
In [33] are given the rules used to group related concepts together for reranking the list of concepts produced by the weighting scheme in such a way that related concepts appear close to each other. This allows concepts that are related to a concept that gets a high score to benefit from this relation and move up the ranking. In the Figure 8 some suggestions can be seen.

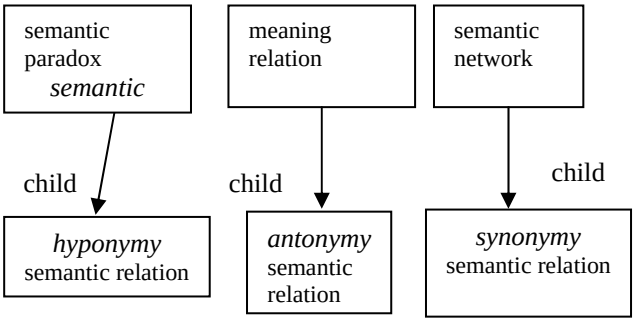

Fig.8: Related concepts

In the OWL version of the ontology [30] special properties hasPart and partOf have been designed as recommended in the W3C Working Draft.

It is needed a framework that provides a complete support to ontology design, implementation, and management. These three functionalities are collectively referred as the ontology service.

The ontology service [30] is based on the following framework components:

- a set of classes representing the object model defining all the elements required to represent an ontology (classes, concepts, instances, attributes, constraints, validation, etc);
- a set of tools that can be used to automatically generate the specific classes for a given ontology by starting from its visual or textual representation;
- an Ontology Agent (OA) which maintains the knowledge about all the ontologies registered in the hosting agent platform and about the agent which are able to communicate by using the concepts defined into a given ontology;
- FIPA SL0 [7] ACL message support.

6.5 Ontology Object Model

The design and the implementation of the object model defining the ontology reflects the specifications outlined in the corresponding FIPA standards [7] and has been inspired by the type system designed in JADE [1] to support ontologies.

The object model defined within the framework defines a meta-ontology which contains all the concepts and the elements which are required to compose user defined ontologies. The meta-ontology defines the following entities: predicate, term, concept, query, action, variable, primitive, and aggregate.
Figure 9 describes how these elements are connected each other.

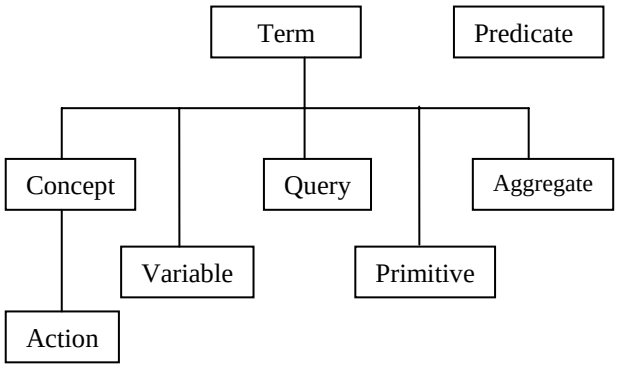

Fig. 9: Ontology elements hierarchy

The elements depicted in Figure 9 define the domain in which every communication based on a given ontology takes place. User defined ontologies will provide specific instances of these elements

and MAS engineers will have to specialize the abstract classes representing the entities defined by the meta-ontology: the new classes will represent the concepts, the queries, the actions, the terms, etc. which are pertaining to the specific problem domain.

6.6 The proposed multi-agent system

In the domain, we had the papers [4], [16] that treat the competence representation and description using ontologies. As we stated above, based on the remarks from [24] and in accordance with our intentions, the following is proposed.

Problem statement: having ontology construct a tool that generates another ontology based on appropriate inference and reasoning.

As a simple example in education: having many course descriptions define the skills and the capabilities and based on these, derive the competences that can be obtained attending these courses. In one of the previous sections were presented the operations on the ontologies. As it can be seen we propose another operation deriving ontology from other one. For that we have at least two possible solutions:

- conceive an expert system with appropriate goals;
- conceive intelligent agents that are able to do it in an appropriate context.

The proposed model for our multi-agent system has as main goal to derive ontology from another one in the following way. It will extract from course descriptions the possible skills and capabilities. From the skills and capabilities, the competences that are acquired which are expressed in terms of a new ontology.

The system has three agents: Extractor, Reasoner and Competence Management Agents.

As is shown in the Figure 10, the ontology that describes the courses and the generic rules are used by the Extractor Agent. It extracts the skills and capabilities that are obtained after attending these courses

After it, the Reasoner Agent defines the possible competences from the skills. These competences are refined based on the comparisons with the similar competences that exist in the Competence repository and the resulted ontology (of the new competences) is obtained.

Our model has some similitude and some functionality like the model presented in [19].

As basis for information and knowledge representation, the XML will be used. The main motivation is due to the fat that on the Internet the information must be extracted processed and presented in some specific form.

Based on it the agents will be able use the information for communicating each other and with the users.

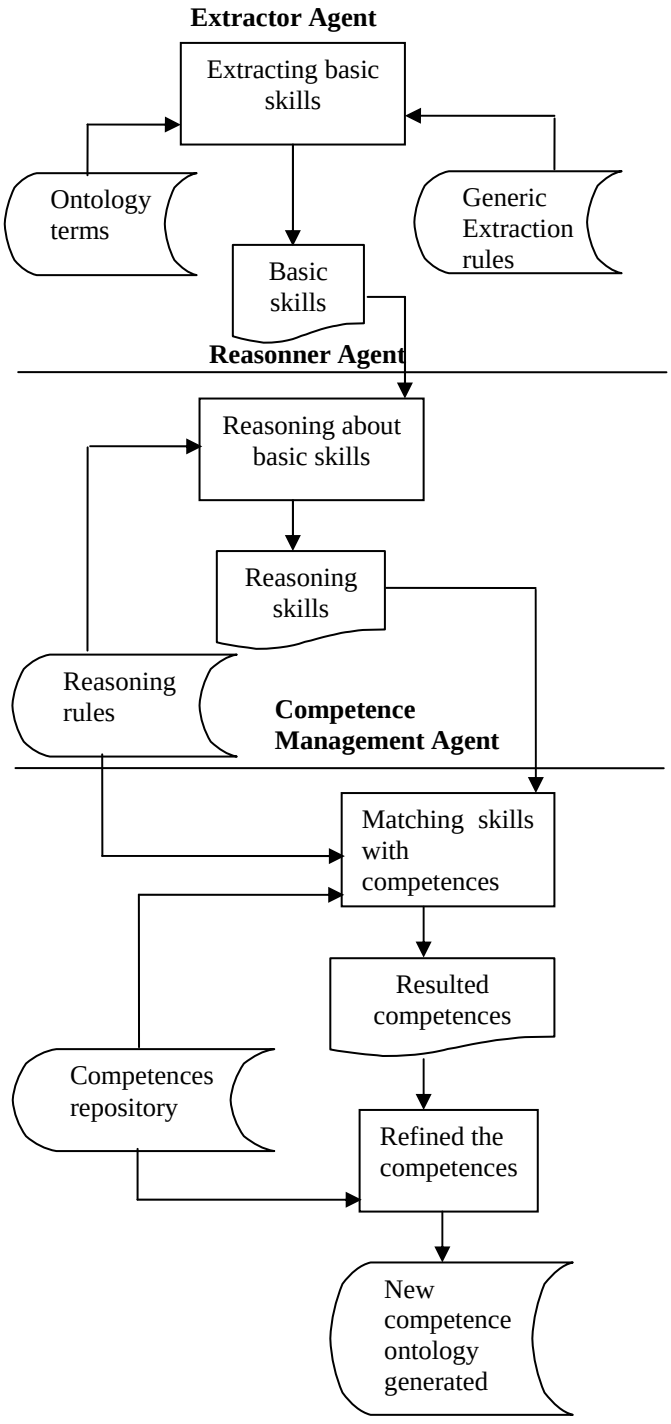

Fig. 10: Multi-Agent system for extracting the competences

7 Future Works

The new proposed operation i.e. the deriving a new ontology from another one will be refined in the new version of our work. The formal definition of derivation operation will be done.

In the next stages we will conceive in detail the capabilities of our agents.

One of the first tasks wills that to define the rules used by the Extractor Agent for:

- finding the sites that have the required information:
- extracting the information, processing and putting it in an XML file for future use.

Other task concern the rules used by the Reasoner Agent that will define the skills and form these appropriate competences.

The other future tasks will be defined after the above work will be conceived and also the concepts and tools used or new tool must be designed, in order to fulfill the model objectives and goals.

References:

[1] G. Caire, *Jade Tutorial – Application-defined Content Languages and Ontologies*, TILab S.p.A, http:/jade.cselt.it/, 2002.

[2] S. Cranefield, M. Purvis, *A UML profile and mapping for the generation of ontology-specific content languages*, Knowledge Engineering Review, 17(1), pp. 21-39, 2002.

[3] S. Cranefield, J. Pan, M. Purvis, *A UML ontology and derived content language for a travel booking scenario*, OAS'03 Ontologies in Agent Systems, 2nd International Joint Conference on Autonomous Agents and Multi-Agent Systems, Melbourne, Australia, 2003.

[4] A. Cicortas, V. Iordan, *A Multi-Agent Framework for Execution of Complex Applications*, Acta Polytechnica Hungarica, Journal of Applied Sciences, Vol. 3 Issue 3, pp. 97-119, ISSN 1785-8860, 2006.

[5] Alexandru Cicortas, Victoria Iordan, Antoanela Naaji, *Ontologies for Competence Description in Multi-agent Systems*, Proceedings of the 10th Wseas International Conference Mathematical Methods and Computational Techniques in Electrical Engineering, Sofia, 2008, ISSN: 1790-5-117, ISBN:978-9606766-602, p.100-107, ISI Thomson proceeding

[6] A.S.Drigas, L.G.Koukianakis, *An Open Distance Learning e-system to support SMEs e-enterprising*, 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering, Data Bases (AIKED 2006), Madrid, Spain, February 15-17, 2006.

[7] A.S.Drigas, A. Tagoulis, P.Kyragianni, P. Nikolopoulos, D. Kalomoirakis, S. Peradonis, D. Kouremenos, CH. Emmanouilidis, J. Vrettaros, *An e-learning platform for multiform and interactive education of scholars in Greek Paleography*, 5th International WSEAS Conference on Distance Learning and Web Engineering, Corfu, Greece, August 23-25, 2005.

[8] FIPA Ontology Service Specification, http://www.fipa.org/specs/fipa00086/.

[9] FIPA SL Content Language Specification, http://www.fipa.org/specs/fipa00008/SC00008I.html.

[10] D. Gasevic, V. Devedzic, D. Djuric, *MDA Standards for Ontology Development*, International Conference on Web Engineering (ICWE2004), Munich, Germany, 2004.

[11] M.R. Genesereth, R.E. Fikes, et al., *Knowledge Interchange Format Version 3 Reference Manual*, Logic-92-1, Stanford University Logic Group, 1992.

[12] T.R. Gruber, *A translation approach to portable ontologies*, Knowledge Acquisition, 5(2):199-220, 1993.

[13] T.R. Gruber, *Toward principles for the design of ontologies used for knowledge sharing*, Workshop on Formal Ontology, International Journal of Human-Computer Studies, Vol. 43, Issues 4-5, November 1995, pp. 907-928.

[14] W. Guo, J. Chen, *A dynamic model for Web-based learning system design*. WSEAS Transactions on Information Science and Applications, Vol. 1, 2004

[15] M. Horridge, *A Practical Guide to Building OWL Ontologies with the Protege-OWL Plug-in*, 1.0, Editor. 2004, University of Manchester.

[16] I. Horrocks et al., *OIL in a Nutshell*, Proc.ECAI '00 Workshop on Application of Ontologies and PSMs, Berlin, Germany, 2000.

[17] I. Horrocks, F. van Harmelen, *Reference Description of the DAML+OIL Ontology Markup Language*, draft report, 2001.

[18] Victoria Iordan, Alexandru Cicortas, *Multi-Agent Systems Used for Managing Competences*, The Proceedings of the 8th International Conference on Informatics in Economy, ASE Publishing House, 2007, Informatics in Knowledge Society, ISBN 978-973-594-921-1, Bucharest, pp. 633-638, 2007

[19] Victoria Iordan, Alexandru Cicortas, *Ontologies used for Competence Management*,

Acta Polytechnica, Journal of Applied Sciences at Budapest Tech, Hungary, ISSN 1785-8860, Volume 5, Issue No.2, 2008, pp.133-144.

[20] R. Karp, V. Chaudhri, J. Thomere, *XOL: An XML-Based Ontology Exchange Language*, Aug. 1999.

[21] R. Kent, *Conceptual Knowledge Markup Language*, 1998.

[22] M. Laclavik, Z. Zoltan Balogh, N.T. Giang, E. Gatial, L. Hluchy, L., *Methods for Presenting Ontological Knowledge to the Users*, L.Popelinsky, M.Kratky (Eds.): Znalosti 2005, Proceedings, VSB-Technicka universita Ostrava, Fakulta elektrotechniky a informatiky, 2005, pp.61-64. ISBN 80-248-0755-6, Vysoke Tatry, Slovakia, February 2005.

[23] S. Luke, J. Heflin, *SHOE 1.01 Proposed specifications*, SHOE Project, Feb. 2000.

[24] G. Librelotto, J.C. Ramalho, P.R. Henriques, *XML Topic Map Builder: Specification and Generation*. In: XATA: XML, Aplicaes e Tecnologias Associadas, 2003.

[25] D.L. McGuinness, F.V. Harmelen, *OWL Web Ontology Language Overview*. 2004.

[26] M. Obitko, V. Snasel, *Ontology Repository in Multi-Agent System*, Procs of Artificial Intelligence and Applications (AIA 2004), Editor: Hamza, M.H., Innsbruck, Austria, 2004.

[27] http://www.obitko.com/tutorials/ontologies-semantic-web/operations-on-ontologies.html

[28] M. Obitko, V. Mafik, *Integration of Multi-Agent Systems: Architectural Considerations*, *Emerging Technologies and Factory Automation*, 2006. ETFA '06. IEEE Conference on 20-22 Sept. 2006 pp. 1145 - 1148

[29] Ontology Summit 2007 - Ontology, Taxonomy, Folksonomy: Understanding the Distinctions, http://ontolog.cim3.net/cgi-bin/wiki.pl?OntologySummit2007

[30] *Ontolog UBL Ontology Project Status Report*, ONTOLOG - collaborative work environment, Open, International, Virtual Community of Practice on Ontology, Ontological Engineering and Semantic Technology, http://ontolog.cim3.net/

[31] *Resource Description Framework (RDF): Concepts and Abstract Syntax*, Graham Klyne and Jeremy J. Carroll, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-concepts -20040210/. Latest version available at http://www.w3.org/TR/rdf-concepts/.

[32] *RDF/XML Syntax Specification (Revised)*, Dave Beckett, Editor, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/. Latest version available at http://www.w3.org/TR/rdf-syntax-grammar/.

[33] A. Rector, C. Welty, *Simple part-whole relations in OWL Ontologies*, 2005, URL:http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/index.html.

[34] *RDF Vocabulary Description Language 1.0:RDF Schema*, Dan Brickley and R. V. Guha, Editors, W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/ . Latest version available at http://www.w3.org/TR/rdf-schema/.

[35] G. Stumme, R. Studer, Y. Sure, *Towards an Order-Theoretical Foundation for Maintaining and Merging Ontologies*, F. Bodendorf and M. Grauer, editor(s), Verbundtagung Wirtschaftsinformatik 2000, pp. 136-149, Aachen,2000.

[36] W. van Hage, M. de Rijke, M. Marx, *Information Retrieval Support for Ontology Construction and Use*, S.A. McIlraith et al, Proceedings 3rd International Semantic Web Conference (ISWC 2004), LNCS 3298, pages 518-533, Springer, 2004.

[37] C. Vecchiola, A. Grosso, A. Boccalatte, *Integrating Ontology Support within Agent Service*, 7thWOA 2006, From Objects to Agents - Grid, P2P, and Self-Systems, Catania (Sicily), Italy, September 2006.

[38] *XML Extensible Markup Language 1.0* (Second Edition), Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Editors, W3C Recommendation, 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006. Latest version available at http://www.w3.org/TR/REC-xml .

[39] *XML Schema Part 2: Datatypes*, Paul V. Biron and Ashok Malhotra, Editors, W3C Recommendation, 2 May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[40] P. Wongthongtham, E. Chang, T. Dillon, I. Sommerville, *Software engineering ontologies and their implementation*, in Kobol, P. (ed), IASTED International Conference on Software Engineering (SE), pp. 208-213, Innsbruck, Austria, Feb 15 2005.