Using Ontologies to Formalize Services Specifications in Multi-Agent Systems

Karin Koogan Breitman, Aluizio Haendchen Filho, Edward Hermann Haeusler, Arndt von Staa

{karin, aluizio, hermann, arndt}@inf.puc-rio.br

Abstract. One key issue in *multi-agent systems* (MAS) is their ability to interact and exchange information autonomously across applications. To secure agent interoperability, designers must rely on a communication protocol that allows software agents to exchange meaningful information. In this paper we propose using ontologies as such communication protocol. Ontologies capture the semantics of the operations and services provided by agents, allowing interoperability and information exchange in a MAS. Ontologies are a formal, machine processable, representation that allows to capture the semantics of a domain and, to derive meaningful information by way of logical inference. In our proposal we use a formal knowledge representation language (OWL) that translates into Description Logics (a subset of first order logic), thus eliminating ambiguities and providing a solid base for machine based inference. The main contribution of this approach is to make the requirements explicit, centralize the specification in a single document (the ontology itself), at the same that it provides a formal, unambigous representation that can be processed by automated inference machines.

1 Introduction

The anchor of our research is the multi agent architectural framework proposed in [Haendchen03]. So far we have analyzed the architectures of several multi agent platforms, notably MESSAGE [Evans00], ZEUS [Azarmi00], JADE [Vitaglione02] and proposed a framework whose innovative structural model overcomes most flexibility shortcomings of other platforms at the same time that promotes large scale architectural reuse. The Agent Framework is described in detail in [Haendchen03, Haendchen04].

In the elaboration process of the Agent framework, we have identified the need for a reference model that centralized the requirements for the services provided/requested by agents operating within our domain in a meaningful way. The initial service specification was written in XML. The document was structured to reflect the MAS architecture hierarchy, i.e., each section corresponded to one of its architectural layers. Although highly structured, this document did not provide any further semantics to aid either the understanding, verification or validation of the specification. Agents could only interact if they shared the exact same specification.

No negociation was possible, for the semantics of the services can not be fully expressed in XML.

We decided to migrate to a more expressive representation. Ontologies were the natural choice, as they are becoming the standard for information interoperability on web [Goméz-Peréz04]. With the adoption of a ontological representation it was possible to formalize terms used in the previous XML service specification, i.e, services, objects, agents and components present in the architecture and the desired ways in which they should interact. In addition to the required syntax, the ontology specification was enriched with semantic content, thus allowing automatic verification, validation with users, and the possibility of negotiating with agents using different service specifications. Different ontologies can be negotiated through the processes of alignment, mapping or merging [McGuiness02, Bouquet03, Breitman03b]. This problem is defined as semantic coordination and can be described as the situation in which all parties have an interest in finding an agreement on how to map their models but given that there is more than one possibility, the right one (or a sufficiently good one) must be chosen [Bouquet03].

An ontology serves as the service specification of an agent operating in the domain, and will be used in making ontological commitments among other software agents [Fensel01]. An ontological commitment is an agreement to use a vocabulary in a way that is consistent with respect to the theory specified by the ontology, i.e., an agreement on what local models are about to achieve user goals [Bouquet03]. We build agents that commit to our ontology. Conversely we design ontologies in order to share knowledge with and among these agents [Gruber93]. The ontology concentrates the desired behaviors and service descriptions in a single document. It serves both as a specification and the reference model to which the agents operating in the domain should comply to.

The rest of the paper is divided as follows: in the next section we briefly introduce the ontology definition and representation language we adopted in the context of our research. In section 3 we describe the context of our MAS. In section 4 we show an example of our approach. In section 5 we briefly describe the lessons learned from this experience and, finally in section 6 we provide our conclusion remarks and future work.

2 Ontology

In order to secure interoperability among autonomous agents, a protocol in which to exchange the necessary information to support this process is required. We argue that ontology, commonly defined in the literature as a *specification of a conceptualization*, is the representation that will provide this requirement [Gruber98]. On one hand ontologies are expressive enough to capture the essential attributes present in MAS, in terms of their classes and relationships. On the other hand, ontologies provide the necessary formality in which to perform automated inference and model checking. According to Tim Berners Lee, ontologies will allow machines to process and integrate Web resources intelligently, enable quick and accurate web search, and facilitate communication between a multitude of heterogeneous web-accessible agents [Berners-Lee01].

We adopt the ontology structure O proposed by Maedche [Maedche02]. According to the author, an ontology can be described by a 5-tuple consisting of the core elements of an ontology, i.e., concepts, relations, hierarchy, a function that relates concepts non-taxonomically and a set of axioms. The elements are defined as follows:

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

- Two disjoint sets, C (concepts) and R (relations)
- A **concept hierarchy**, H^C: H^C is a directed relation $H^C \subseteq C \times C$ which is called concept hierarchy or taxonomy. $H^C(C_1, C_2)$ means C_1 is a subconcept of C_2
- A **function** $rel : R \rightarrow C \times C$ that relates the concepts non taxonomically
- A set of ontology **axioms** A^O, expressed in appropriate logical language.

Most existing ontology representation languages can be mapped to this structure, e.g. RDF, Oil and DAML, but there seems to be a consensus to adopt OWL as the *de facto* language to represent ontologies. OWL is being developed by the W3 consortium as an evolution of the DAML standard [Hjem01, Hendler00, McGuiness03]. The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with a formal semantics. The OWL specification comprises three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full. OWL Lite supports classification hierarchies and simple constraints, e.g., cardinality. It is intended as quick migration path from taxonomies and thesauri, i.e., that are free from axioms or sophisticated concept relationships. OWL DL supports *"expressiveness while retaining computational completeness (all conclusions are guaranteed to be computed) and decidability (all computations will finish in finite time)"* [McGuinees03]. DAML+OIL is equivalent, in terms of expressiveness, to OWL DL. Finally, OWL Full supports maximum expressiveness. According to the W3 consortium, it is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

The existence of a large repository of ontologies also influenced our decision to migrate to OWL as the ontology representation language used in our projects. In table I we show the mapping between the nomenclature used by the O ontology model and the one adopted by OWL.

Table 1. Terminology mapping between the O ontology structure and the ontology language OWL

O Ontology Structure		OWL
C	Concept	Class
R	Relation	Property
Hc	concept hierarchy	Subsumption relationship: SubClassOf
rel	function that relates the concepts non taxonomically	Restriction
AO	Axiom	Axiom

OWL provides the modeling primitives used in frame based systems, i.e., concepts (or classes), the definition of its superclasses and attributes. Relations are also defined, but as independent entities, properties, instead of class attributes. The primitives provide expressive power and are well understood, allowing for automated inference. The formal semantics are provided by Description Logics (DL). DLs also known as terminological logics, form a class of logic based knowledge representation languages, based on the primitives above [Horrocks02]. DLs attempt to find a fragment of first order logic with high expressive power which still has a decidable and efficient inference procedure [Newell82, Heinsohn94]. FaCT is a working example of a system that provides reasoning support (i.e., consistency and subsumption checking) to OWL-encoded ontologies [Horrocks01].

An OWL ontology is a sequence of axioms and facts, plus references to other ontologies, which are considered to be included in the ontology. OWL ontologies are web documents, and can be referenced by means of a URI. Ontologies also have a non-logical component that can be used to record authorship, and other non-logical information to be associated with an ontology [OWL, McGuiness03].

In the next section we present the MAS Framework we have been experimenting with and relate the construction process of its service ontology.

3 MAS Framework

Agent-oriented software engineering extends the conventional components' development approach, leading to the construction of more flexible and component-based MASs [Griss03], emphasizing reuse, low-coupling, high-cohesion and support for dynamic compositions. Rapid and problem-specific system construction can be attained through the use of model-driven development and reuse techniques in order to achieve a more flexible, adaptable, robust and self-managing application. These properties can be constituted by the combination of several technologies, such as component-based software engineering [Griss03,24,38], frameworks [Bosh99, Fayad99, Pree99, Roberts98], design patterns [Gamma95, Larman98], rule-based systems [Gelfond93, Paton95, Yu00] and now ontologies [Fensel03, Berners-Lee01, Hendler01]. The MAS Framework architecture comprises five layers: Domain, Multi Agent System (MAS), Agent, Module and Class. Figure 1 depicts the Framework architecture. Note that the Module and Class layers are located inside each agent, the modules are represented in Figure 1 by the circles labeled S30, S40, S50 and S51 (the classes are not represented in the Figure. They are internal parts of the modules). Note that there are two ontologies in the architecture, illustrated by circles S4, and S9. The first one, S4 is the upper ontology and contains the specification of shared domain services, i.e., infrastructure, interface and communication services that will always be instantiated by our Framework. This ontology was built by experts and is part of implementation of the Framework. The second ontology, located at the MAS level, illustrated by circle S9 in Figure 1, represents the agent specific ontology. It contains hot spots where particular application services are to be specified during the Framework instantiation process. As a consequence of the multi layered architecture of the Framework, application services are specified under the domain level, i.e., as leaves of the upper ontology. For all practical purposes, the agent specific ontology is a composition of the upper ontology (top levels) with the addition of the specification of application specific services at the bottom levels (MAS and agent).

Each MAS centralizes its service specification in a single document (represented by the circle labelled S_9 in Figure1). In our architecture, agents preferentially receive services requirements through a single interface, instead of interacting directly with one another, using multiple interfaces. This communication is done using highly structured messages composed using the terminology formalized by the service ontology. This way, both the syntax required by the interface specification and the semantics associated to the terms used in the service request are now available. Providing clear semantics of of the terms in use, helps maintain clarity and transparency of the specification. It serves as an aid to the ontology validation process and also as a guide to non expert users in the processof including new service specifications at the agent layer.

The syntax of the services provided by each agent, and how they can be accessed, is provided by the interface specification. Thus, an essential part of the process is defining a syntactic description of each interface and how the services can be accessed. The aim of the service specification ontology is to identify the services associated with each agent, specifying the main properties of these services. For each service that may be performed by an agent, it is necessary to document its properties. In particular we must identify inputs, outputs, pre-conditions, post-conditions, parameters, states, transitions and rules.

Initially we used an XML document to serve as the service specification. It contained descriptions of the services provided and interface parametrization. Although structurally sound, the XML document was found semantically weak and unfitting to describe some aspects of the service specification, e.g., rules and states. Migrating to an ontological respresentation was a natural move.

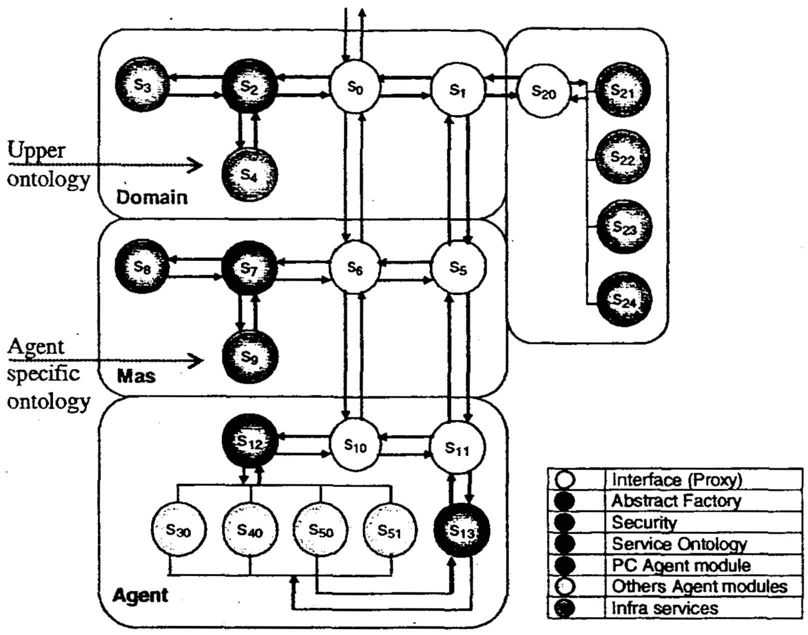


Fig. 1. – MAS Framework Architecture

We had to question ourselves whether it was possible to express all the necessary information in the MAS service specification using the available ontology languages. As presented in section 2, the current W3C recommendation language for ontology modelling is OWL, the evolution of previous efforts in finding a standard ontology language. OWL comprises three different languages, the choice of which should be based in the level of expressiveness desired for the ontology in question. The first language, Lite OWL, was definetively not expressive enough to capture the necessary information present in the service specification. Our choice was between OWL-DL and Full OWL. The later, although allowing for maximum expressiveness, does not guarantee the possibility of automatic reasoning in computable time [OWL]. In our case, the use of inference to help verify overall specification consistency is very important, so we chose OWL-DL as the preferred language. The last ensures decidability and the existence of an efficient inference mechanism for the language [McGuiness03]. This choice, however, came with an additional modelling overhead. OWL-DL does not directly provide some modelling primitives, e.g., class attibutes and an-ary relationships. Those can be obtained by means of some workarounds . This is common practice in the mark up language community. Assuncíon Gómez-Pérez, Mariano Fernández-López and Oscar Corcho published a table of the most common workarounds (partially reproduced in Table II) [Gómez-Pérez04].

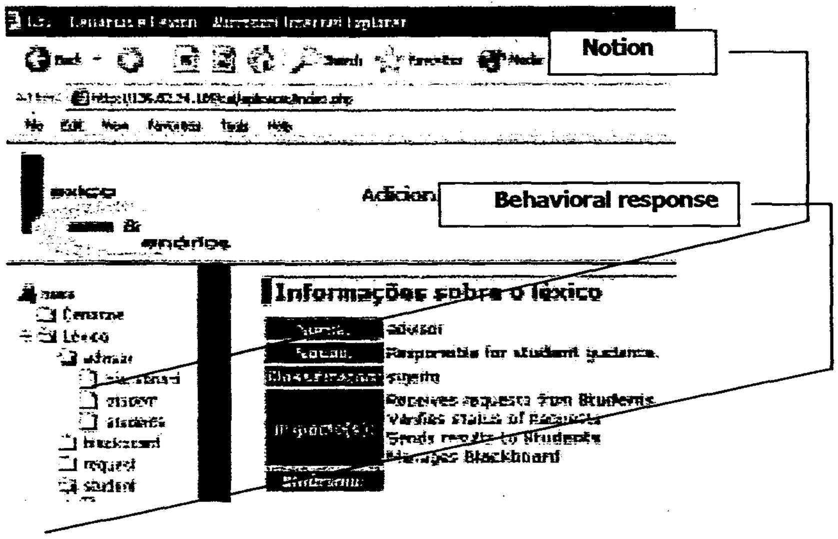
We build ontologies using the lexicon based ontology construction process proposed in [Breitman03]. This process is influenced by our background in requirements engineering and system specification and uses the Extended Lexicon of the Language (LEL) [Breitman03c, Leite93], referred to as Lexicon from here on, as the starting point. We initiate the process by building a Lexicon that captures the vocabulary of our application, i.e., the basic concepts and the relationships that bind them together in an informal way (using natural language). The Lexicon models a series of definitions of the services, objects, agents and components, present in the MAS architecture, and the desired ways in which they should interact. Such definitions evolve from an informal, natural language lexical representation to a formal, machine processable, ontological representation through the application of the lexicon-to-ontology mapping rules defined in [Breitman03].

Table 2. Markup Language Workarounds

	RDF(S)	Daml+Oil	OWL
CONCEPTS			
Instance attributes	+	+	+
Class attributes	W	W	W
Facets			
Type constraints	+	+	+
Cardinality constraints	-	+	+
Procedural Knowledge	-	-	-
CONCEPT TAXONOMIES			
subclass-of	+	+	+
Disjoint decomposition	-	+	+
Exhaustive decomposition	-	W	W
Partition	-	+	W
RELATIONS			
Binary relations	+	+	+
N-ary relations	W	W	W

The Lexicon represents domain information obtained with the help of well known elicitation techniques, e.g. questionnaires, observation, structured meetings. It captures both the denotation and connotation of important domain concepts. Differently from usual dictionaries, that capture the meaning (denotation) of an entry, the Lexicon also captures its connotation, i.e., the behavioral response or impacts that a lexicon entry might have in defining other entries [Leite93].

To build the service specification Lexicon we started with the elicitation of important domain[1] concepts. Those were present in the XML specification, but were not defined to satisfaction. To elicit their meaning, we applied questionnaires and structured interviews with domain experts, i.e., the software engineers involved in the construction of the first specificatio In Figure 2 we show an example of a lexicon entry. We depict the Advisor entry. The Lexicon elicitation and construction process is fully described in [Breitman03].



Informações sobre o léxico

Termo: advisor
Noção: Responsible for student guidance.
Classificação: sujeito
Impacto(s): Receives requests from Students
Verifies status of Requests
Sends results to Students
Manages Blackboard
Sinônimo(s):

Fig. 2 – Screen snapshot of the Lexicon entry Advisor in the C&L tool

To generate the formal ontology we applied the process proposed by Breitman & Leite to the newly built Lexicon. This process consists of a set of rules that map Lexicon entries into the five ontological elements proposed by Maedche, described in section 2.

Lexicon entries are typed in one of subject, object, verb or situation. Depending on the type a different set of rules is applied to the Lexicon entry and will result in its mapping to either an ontology concept or property. The notion of a Lexicon entry is mapped into the description of its correspondent ontology concept. Its behavioral

[1] Please note that we use the term domain in the broad sense, signifying the application domain as a whole. In this case, our domain is the entire multi agent framework, for which we intend to build a service specification, as opposed to its top layer that is incidentally named domain as well.

responses serve as an aid in the identification of ontology properties, concept restrictions and non taxonomical relationships among ontology concepts. Axioms come from the identification of disjoint or generalization relationships held among Lexicon entries. The lexicon based ontology construction process is described in detail in [Breitman03]. This process is supported by C&L, an Open Source tool that automates great part of the lexicon to ontology mapping process. Some design decisions have to be taken by the software engineer and can not be fully automated [Breitman03c]. The tool also provides automated support for the creation and management of Lexicons [Felicíssimo04]. In Figure 3 we show the upper service ontology.

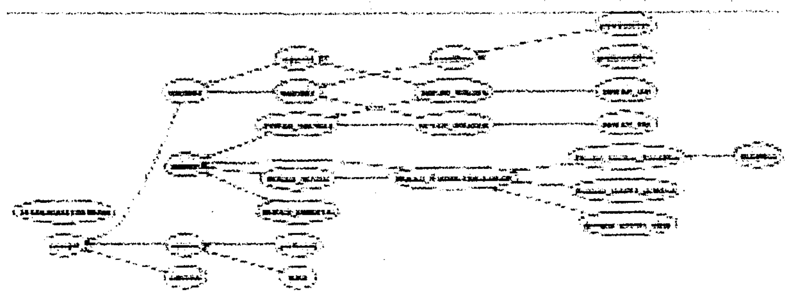


Fig. 3 - Upper service ontology

In this section we described the construction of the upper service ontology. Specific services provided by the agents are specificied in the application ontology, located at the MAS level, as shown in Figure 1. As mentioned before, it is a direct consequence of the multi layer architecture of the Framework that specific agent services are specified as leaves, i.e., placed under the lowest levels of the upper ontology. Evidently, those services are particular to each implementation and can not be provided by the upper ontology. Those specifications must be included by a software engineer, as part of the implementation of the MAS itself, and vary case by case. In the next section we exemplify our approach.

4 Academic Control System: an example

To exemplify our approach we chose an academic control system MAS that tracks the undergraduate student advisement process. We focus on the services provided by the Advisor agent, as illustrated in Figure 4.

In the advising process, a student fills out a registration form with his/her name, student ID, the current semester and the details of the course he/she would like to take. After sending the request, the student receives the final results, either an enabling password or the justification for denying the request. The *Advisor* has the function of taking the student request and to conduct preprocessing, validating the student, verifying the syntactic aspects, checking the viability of the schedule, to direct the request result for the student or providing a request status.

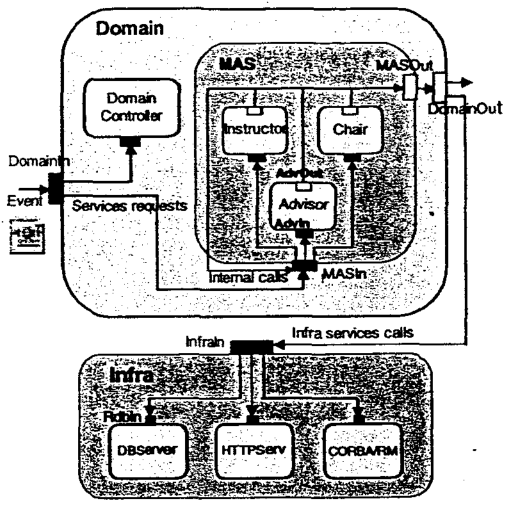


Fig. 4. System generic architecture as proposed in [Haendchen03], instantiated to the academic control MAS example

The agent *Chair* can make a slot available whenever the class is full, and the agent *Instructor* can dismiss pre-requisites for a course. The instructor and chair agents exchange messages with human agents through well-defined and well-structured e-mail messages. The advisor receives the request and verifies syntactic aspects, if the student has the prerequisites to the intended courses and checks to see if there are vacancies in the desired classes. If these conditions are met, the advisor authorizes the request by signing it and gives the student the registration password needed to register for the course. If these conditions are not met, the advisor directs the request according to the arguments of the event to the student, instructor or to the chair. While the process is under way, the student can ask the advisor for information about the progress of the request by e-mail. In any case, the advisor returns the request to the student via e-mail, specifying the result. Based in this information we modeled the Lexicon of the services provided by the system. In the academic control MAS case we used interviews and observation techniques to help elicit lexical information from the domain.

Through a series of refinements, the academic control Lexicon was mapped to its formal ontology. This process was semi automated, for some human input is necessary at specific decision points. The C&L open source tool automates this process and was used to support the construction of the academic system ontology [Silva03].

In Figure 5 we show a screen snapshot of the ontology of services provided by the academic control system. We focus on the the domain_out interface. Please note that, however some restrictions are defined at concept level, there is a great number of other restrictions inherited by its super classes (see the restriction box in the lowest right corner of Figure 5). The super class of class domain_out is indicated by Classes box, namely domain_required[2].) The ontology was implemented using the OilEd, a freeware tool for ontology editon developed at the University of Manchester, that exports to the chosen OWL format [Berchofer01].

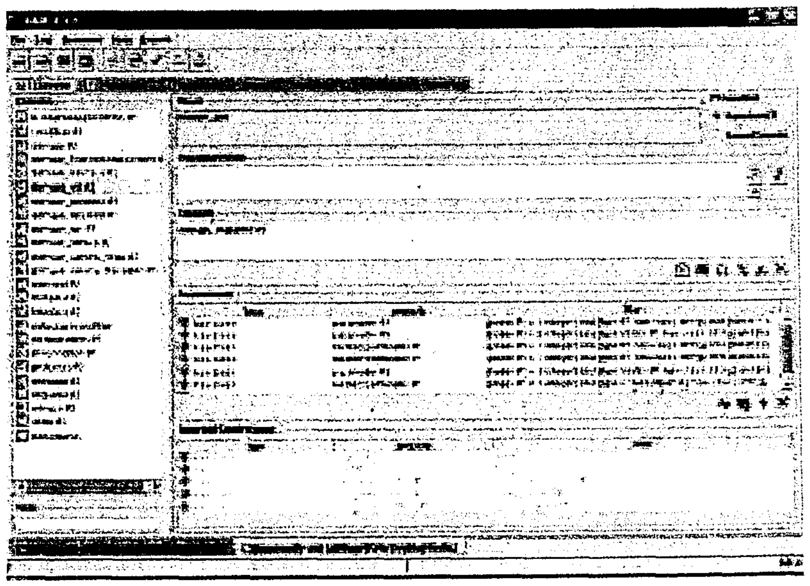


Fig. 5. The ontology concept domain_out implement using the OilEd tool.

We took special care to ensure overall model quality. We have validated the Lexicon with the users and verified using inspections [Kaplan00]. The ontology was verified using the FaCT (fast classification of terminologies) inference engine, publicly available at [FaCT04]. The reasoning services provided by this tool include

[2] The # symbol that appears as a suffix of the classes indicates the namespace of the class, i.e., the name of ontology where the specification of the class resides. OWL and similar mark up languages do not require that all concepts in the ontology are specified in the same document. By using the namespace mechanism, it is possible to reuse concepts defined in other ontologies, provided that a valid path to that document is given.

inconsistency detection, determining subsumption and equivalence (among classes) relationships.

In Figure 6 we illustrate an inconsistency identified with the aid of of the inference mechanism. The ontology has axiom that states that the classes *MAS Security Checking* an *Domain Security Checking* are disjoint, i.e., their intersection is empty. This is is illustrated by the panel in left, that contains the list of axioms for the Academic Control ontology. In the right most panel we depict the ontology, as it was being built. In this process we specified a restriction in which a state would only be reached in the event that both *MAS Security Checking* an *Domain Security Checking* were activated.

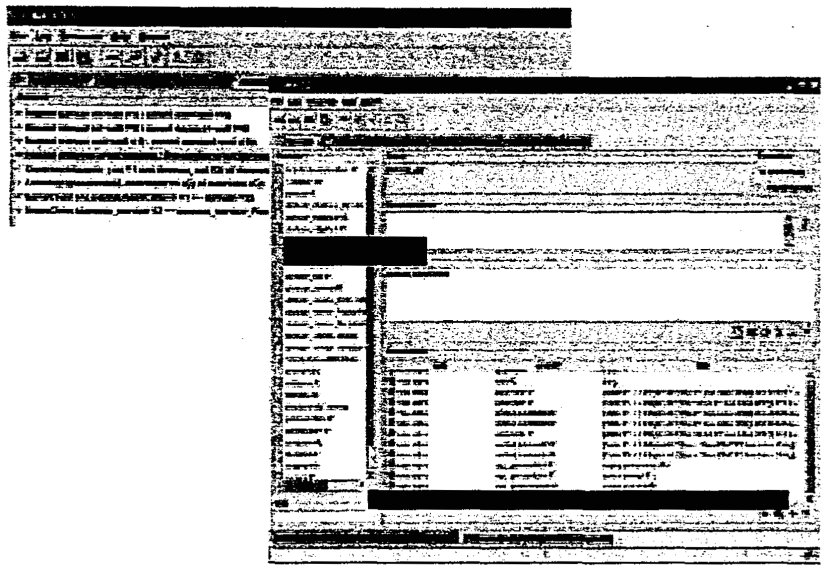


Fig. 6. Inconsistency in class domain_out

This situation is an impossibility, for the classes are forcibly (as explicited by the axiom in the left pane) disjoint. During the construction of the ontology this fact passed noticed by the designers. The consequences this error may bring to the implementation of the MAS are very serious, for that may cause the agent to halt or to enter a dead loop state. This fault was automatically detected with the use of the reasoner, as illustrated in Figure 7. We depict three panes; In the first one we show the interface to the FaCT reasoner. This tool is built in common Lisp and makes inferences over a description logic representation of the ontology. The ontology editor, OilEd translates the ontology to SHIQ (----a description logic language dialect) and sends to the reasoner using a CORBA interface. On the second pane, middle one, we depict the log of the reasoning process. We emphasize that the class domain_out is unsatisfiable, but note that the reasoner also checks for errors in subsumption relationships and class instances. The third pane, rightmost, illustrates the graphical display of the inconsistency in the OilEd tool. Similarly to this case, the reasoner helped us detect other inconsistencies in the ontology. We also performed manual verification, using a process very similar to software walkthroughs: we gathered a group of three designers and revisited the material during a planned meeting. The chief designer of the ontology served as group mediator and conducted the meeting. The errors found we mostly sintactical, e.g., classes, properties and restrictions wrongly named or typos. A few inconsistencies such as the one illustrated in Figure 6 were also found. We noticed that the inheritance mechanism makes it very hard to identify inconsistencies when they are the result of a composition of restrictions that appear in different levels, i.e., one was defined at class level and the other was inherited from a super class. It is important to note that all of this type of inconsistencies were also detected by the reasoner in a later moment. We concluded that manual verification is worthwhile, for it helps identify problems that could not be otherwise detected. Practitioner should, however, focus in the terminology, usage and validation of ontological terms. Inconsistencies are more sistematically detected with the aid of an automatic reasoner.

The reasoner was also useful in the identification of a group of classes that partook a similar setting. To illustrate this situation we present the example of class *alert condition*. This class, as illustrated in Figure 7, is defined if two of its restrictions are true, namely **in!= null** and **security_check = 7)** . We defined this class in the ontology of the type SameClassAs, i.e., this is a necessary and sufficient condition to define any other class that possesses those requirements as a similar to class *alert condition*)

Class *domain_out* of the Academic Control ontology is an example of a class that fullfills this requirement. We depict this class and its restrictions in Figure 8 as follows. Note that one of the restrictions was specified among the class natural restrictions, the second came as an inherited restriction from its super class, domain_required. This mechanism is very interesting to help ensure that some conditons are met across the ontology.

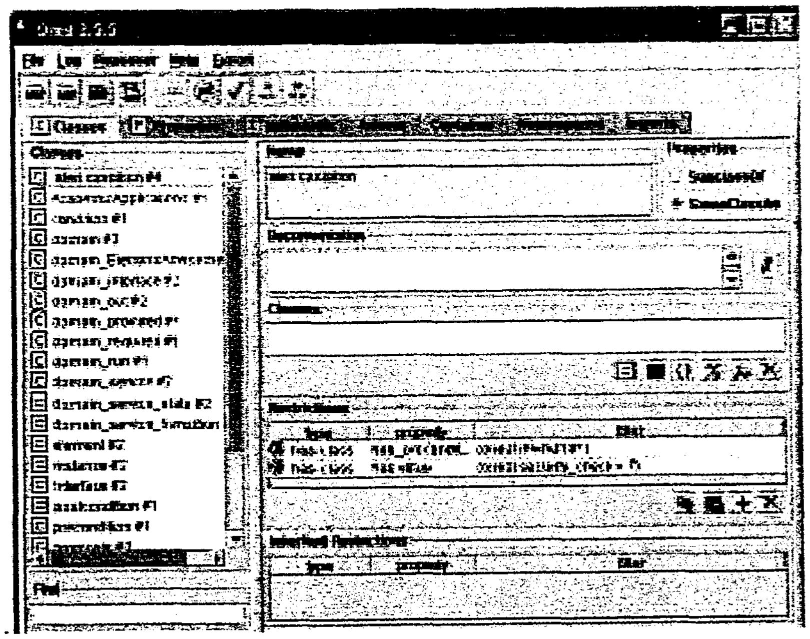


Fig. 7. Class alert condition

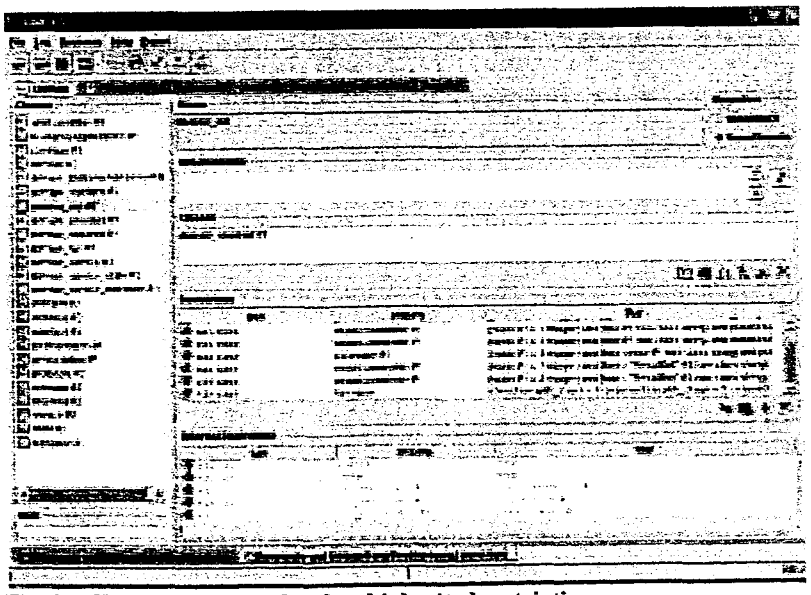


Fig. 8 – Class domain_out, local and inherited restrictions

As an illustration we also show the OWL code for the *Domain_out* class in Table 3. Note the similarity to XML, and the fact that the language uses RDF constructors, e.g., subClassOf. This is intentional and is a direct consequence of the "*wedding cake*" architecture for ontology languages proposed by Tim Berners-Lee [Fensel03]. This model reflects the evolution of ontology mark up languages. Each new gain in semantics resulted in the construction of a new language layers, put on top a XML basis. The first layer was RDF, followed by RDF Schema. Because those were not expressive enough, a new wave of languages, including DAML, OIL and now OWL was proposed and put on top of the RDF layer. The result is that an OWL document contains OWL specific markup as well as primitives imported from layers below, e.g., rdfs: label.

Table 3. Example of OWL code for the domain_out class (partially represented)

```
<owl:Class rdf:about="file:/C:/Documents/AcademicAplications.owl#domain_out">
        <rdfs:label>domain_out</rdfs:label>
        <rdfs:comment><![CDATA[]]></rdfs:comment>
        <oiled:creationDate><![CDATA[2004-04-18T22:20:56Z]]></oiled:creationDate>
        <oiled:creator><![CDATA[Karin]]></oiled:creator>
        <rdfs:subClassOf>
            <owl:Clas
rdf:about="file:/C:/OilED/ontologies/AcademicAplications.owl#domain_required"/>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="file:/C:/Documents/Karin/AcademicAplications.owl#public"/>
                <owl:hasClass>
                    <owl:Thing/>
                </owl:hasClass>
            </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
            <owl:Restriction>
                <owl:onProperty
rdf:resource="file:/C:/Documents/Karin/AcademicAplications.owl#void"/>
```

We must finally remark that a great level of ontology reuse is achieved as a result of our multi level Framework architecture. Generic services provided by every MAS are specified by the upper ontology and need not be specified again. The only services that require a specification effort are those particular to the agent in question. Even so, some of the inputs, pre and post conditions may be inherited from the super class under which the service is to be specified. The reuse of specifications not only reduces overall effort, but also serves to ensure quality because we are making use of a specification that was built by experts (less prone to mistakes), was verified by inspection, and has been tested in other applications[3].

[3] It is important to note that the upper ontology is continuously being refined as a result of reports from practitioners.

5 Lessons Learned

The evolution from the XML service specification to an OWL ontology was an overall positive experience. Our initial concerns related to the power of the ontology representation to convey specification details were lifted as we were able to model every concept in the service specification in the ontology. During this process some workarounds were needed, specially to formalize attributes such as function parameters and transitions.

The use of the FaCT reasoner helped verify the ontology and improve its overall quality. Automatic verification helped detect: inconsistencies (pre and post conditions, parameters, undesireable situations), errors, and some omission. Additional verification mechanisms will have to be used, as strings (e.g. Regular expressions) are processed as a block by the reasoner.

Our experience in building the service ontologies to support our MAS communication exchange has shown that this task is a very complex one. Despite the existence of methods to support ontology construction, it still remains more of a craft than a science [Fernandez-Lopez97, Gruninger95, Noy01-b, Sure03, Ushold96, Breitman03]. The decisions that have to be taken during this process, e.g., decide whether a concept should be mapped into a class or property, are very difficult and require expertise in concept modeling. By the same rule, the workarounds that have to be used in order to represent relevant specification concepts in the ontological representation are not trivial. It requires the ability to identify such concepts and to engender a workaround that maximizes the power of expression of the ontology.

Finally tool support for visualizing ontologies is still very poor. For ontology edition we have been using OilEd and Protégé [Berchofer01, Noy01]. Both tools fulfill our current editing requirements and have proven very reliable and easy to use. Our main concern today is the need for a tool that allows for a better visualization of the ontology, to help in the validation process[4].

6 Conclusion

In this paper we propose to using ontologies as a means to capture and publish the specifications of the services provided by the agents in a MAS. The ontology makes the requirements explicit, centralizes the specification in a single document, at the same that it provides a formal, unambigous representation that can be processed by automated inference machines [Sowa00]. The main contribution of this approach is to put in practice a standardized reference model when specifying new agents, components and object behavior in a MAS. We showed the feasibility of the approach by

[4] Both OilEd and Protégé provide visualization plug-ins. Those are static drawings of the ontology, usually too big and cumbersome. Neither plug in provides the necessary functionality required in the validation process.

means of an example in which we constructed an ontology that specified the services provided within an academic control MAS.

The change from an XML representation to a OWL resulted in real quality gain for the service specification. The current ontological representation is more reliable, for it can be automatically verified. Consistency is thus guaranteed by automatic inference. Furthermore Results from our analysis process (verification and validation) confirm that the OWL specification is more consistent and error free than the previous XML one.

The use of ontologies opens the possibility of interfacing with other MAS environments. As envisioned by James Hendler, the web of the future will be composed of a multitude of websites, network services and databases, each operating with its own local and contextualized ontology [Hendler01]. There is an ongoing effort to support the integration and alignment of different ontologies, in order to support communication and services exchange [Breitman03-d, Bouquet03, McGuiness02]. The ability to align different ontologies will make it possible to probe and request services in truly open ended environments, such as the web [Heflin01].

We are currently experimenting with semantic coordination of MAS ontologies.. We have developed a mechanism to align two different ontologies, CATO, that is publicly available in internet [Felicíssimo04]. We are using this mechanism to help integrate MAS operating in the health care domain. Our current experiment is trying to integrate services provided by a multi agent system used for the diagnoses and treatment of altistic children to similar health care multi agent systems. Our intention is to use the integration process to negotiate among different MAS thus providing new services that were not initially available, e.g., we are currently trying to align our MAS to the Retsina Calendar Agent as to provide appointment services.

The service specification ontology serves us in two ways. Externally of our Framework, the ontology communicates the semantics of the services provided by agents of our domain, thus allowing for exchanges among different MAS and interaction with other agents in Open Ended environments, such as the World Wide Web. Internally to our Framework structural, the ontology serves as a formal specification of the catalog of services provided. Every agent/component operating within our structural model must abide to the specifications dictated by the domain services ontology. The same is true to components and objects.

Future work includes the investigation of a visualization mechanism that would allow for the separation and display of services provided by each layer. The user interface of this mechanism will be inspired in the vision mechanisms of relational databases. At the same time we are considering the development of new plug ins that implement additional verification routines (e.g. lexical and syntactic analysers for strings – parameters, regular expressions), that are not currently covered by the inference mechanisms.

7 References

[Azarmi00]Azarmi N., Thompson S. ZEUS: A Toolkit for Building Multi-Agent Systems. Proceedings of *Fifth Annual Embracing Complexity Conference*, Paris April 2000.

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

[Berners-Lee01]- Berners-Lee, T.; Lassila, O. Hendler, J. – The Semantic Web – Scientific American - http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

[Bosh99] - Bosch, J., Molin P., Mattsson M.; Bengtsson P.; Fayad M. Framework problem and experiences in M. Fayad, Building Application Frameworks, John Willey and Sons, p. 55–82, 1999.

[Bouquet03] –Bouquet, P.; Serafini, L.; Zanobini, S.; - Semantic Coordination A new aporach and an application – in Proceedings of the 2nd. International Semantic Web Conference – Florida, October 2003 – pp. 130-143.

[Breitman03] - Breitman, K.K.; Leite, J.C.S.P - Lexicon Based Ontology Construction - Lecture Notes in Computer Science 2940- Springer-Verlag, 2003, pp.19-34.

[Breitman03b] - Breitman, K.K.; Leite, J.C.S.P. - Semantic Interoperability by Aligning Ontologies - Proceedings of the Requirements Engineering and Open Systems (REOS)-Workshop at RE'03, Monterey, USA, September 2003.

[Breitman03c] - Breitman, K.K., Leite, J.C.S.P.: Ontology as a Requirements Engineering Product, Proceedings of the International Conference on Requirements Engineering, IEEE Computer Society Press, 2003. pp. 309-319

[Evans00] - Evans, R. "*MESSAGE*: Methodology for Engineering Systems of Software Agents". Deliverable 1, July 2000.[FaCT04] - http://www.cs.man.ac.uk/~horrocks/FaCT/

[Fayad99] - Fayad M.E. et al. "Building Application Frameworks". John Wiley & Sons, Inc. New York, 1999.

[Felicíssimo04] - Felicíssimo, C.H.; Leite, J.C.S.P., Breitman, K.K., Silva, L.F.S. - C&L: Um Ambiente para Edição e Visualização de Cenários e Léxicos - XVIIII Simpósio Brasileiro de Engenharia de Software (SBES) - Brasília - 18 a 22 de Outubro de 2004 - to appear.

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

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

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

[Gamma95] - Gamma E. et al. "Design patterns – elements of reusable object-oriented software." Addison-Wesley Longman, Inc., 1995.

[Gelfond93] - Gelfond M. "Representing Action and Change by Logic Programs". The Journal of Logic Programming. Elsevier Science Publishing Co, New York 1993.

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

[Gruber98] – Gruber, T. - A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2):21–66, 1998.

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

[Haendchen03] Haendchen Filho, A.; Staa, A.v.; Lucena, C.J.P. "A Component-Based Model for Building Reliable Multi-Agent Systems". Proceedings of 28th SEW - NASA/IEEE Software Engineering Workshop, Greenbelt, MD. IEEE Computer Society Press, Los Alamitos, CA, 2004, pg 41-50.

[Haendchen04] - Haendchen, A.; Caminada, N.; Haeusler, H.; von Staa, A. - Facilitating the Specification Capture and Transformation Process During the Formal Development of Multi-Agent Systems - Proceedings Of Third NASA/ IEEE Workshop on Formal Approaches to Agent Based Systems, Los Alamitos, California USA. To appear as LNCS, Springer-Verlag.

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

[Heinsohn94] - J. Heinsohn, D. Kudenko, B. Nebel, and H.-J. Profitlich. An empirical analysis of terminological representation systems. Artificial Intelligence, 68:367--397, 1994.

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

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

[Hjem01] - Hejem, J. - Creating the Semantic Web with RDF - Wiley, 2001

[Horrocks01] - Ian Horrocks and U. Sattler. Ontology Reasoning for the Semantic Web. In In B. Nebel, editor, Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI'01), Morgan Kaufmann, pages 199--204, 2001.

[Horrocks02] - I. Horrocks- Reasoning with expressive description logics: Theory and practice- A. Voronkov, editor, Proceedings of the 18th International Conference on Automated Deduction (CADE 2002)

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

[Larman98] - Larman C. "Applying UML and Patterns". Prentice Hall PTR. Upper Saddle River, NJ, USA, 1998.

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

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

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

[Newell, 1982] Newell, A. (1982). The Knowledge Level. Artificial Intelligence, 18:87-127.

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

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

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

[OWL] - http://www.w3.org/TR/owl-ref/

[Paton95] - Paton N.W. "Supporting Production Rules Using ECA-Rules in an Object-Oriented Context". Department of Computer Science. Technical Report. University of Manchester, UK, 1995.

[Pree99] - Pree, W. Hot-spot-driven development in M. Fayad, R. Johnson, D. Schmidt. Building Application Frameworks: Object-Oriented Foundations of Framework Design, John Willey and Sons, p. 379–393, 1999.

[Roberts98] - Roberts D., Johnson R. Evolving frameworks: A pattern language for developing object-oriented frameworks in Martin R.C., Riehle D., Buschmann F. Addison-Wesley,1998.

[Silva03] - Silva, L.F.; Sayão, M., Leite, J.C.S.P.; Breitman, K.K. - Enriquecendo o Código com Cenários - 17 Simpósio de Engenharia de Software (SBES) - Manaus, AM - 2003 - ISBN 85-7401-126-6 - pp.161- 176.

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

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

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

[Vitaglione02] - Vitaglione G., Quarta F., Cortese E. Scalability and Performance of JADE Message Transport System. Proceedings of *AAMAS Workshop on AgentCities*, Bologna, 16th July, 2002.

[Yu00] - Yu L. et al. "A Conceptual Framework for Agent Oriented and Role Based Workflow Modeling." Technical report, Institute for Media and Communications Management, University of St. Gallen, Switzerland, 2000.