ADVANCES IN ONTOLOGIES

Proceedings of the
Australasian Ontology Workshop
Melbourne, Australia, 1 December 2009

Thomas Meyer and Kerry Taylor, Editors

Preprint. To be published as Vol 112 in the Conferences in Research and Practice in Information Technology Series by the Australian Computer Society Inc. ISBN 978-1-920682-91-0
http://crpit.com/

Preface

The series of Australasian Ontology Workshops, begun in 2005, is now in its fifth year with the Australasian Ontology Workshop (AOW 2009), to be held at the University of Melbourne, in Melbourne, Australia on the 1st December 2009. Like most of the previous workshops, AOW 2009 is held in conjunction with the Australasian Joint Conference on Artificial Intelligence, now in its 22nd year as AI'09.

We expanded the scope of the workshop slightly this year, in line with significant growth in semantic web technologies. In addition to the traditional ontology and knowledge representation topics, we specifically requested papers on novel ontology applications, and on linking open data. We also sought papers on contributions of ontologies to e-research: the latter topic being of major interest in Australia at this time due to the knowledge-sharing emphasis of the Government's National Collaborative Research Infrastructure Strategy. Our invited keynote speaker, Professor Peter Fox of Rensselaer Polytechnic Institute in New York State, USA, will address the e-research topic through his presentation on experiences in ontology development primarily for the High Altitude Observatory at the National Center for Atmospheric Research in the US.

Out of ten papers submitted, we accepted six on the basis of three or four reviews each of full papers by our program committee of international standing. Our papers offer an interesting balance of topics with two on formal ontology topics, two on applications of ontologies in software engineering, and two relating to ontologies in sensor networks. Despite being a nationally-titled workshop, located with a national conference, we were very pleased to note the truly international nature of our submitting authors: from Finland, China, South Africa, and the United Kingdom as well as Australia.

For the first time this year, the conference offered a modest best paper award. Unfortunately, we are unable to announce the winner at the time of writing.

Many individuals contributed to this workshop. We thank our contributing authors and our invited speaker, Peter Fox, who will be travelling to Australia especially for this event. We thank our international Program Committee and additional reviewers for their careful reviews in a tight time-frame. We appreciated the support of the organising committee for AI'09, most especially the workshops chair, Christian Guttmann.

We acknowledge the EasyChair conference management system which was used in all stages of the paper submission and review process and also in the collection of the final camera-ready papers. We thank Vladimir Estivill-Castro, John Roddick and Simeon Simoff, the editors of the CRPIT series, for facilitating the formal publication of the AOW 2009 proceedings, and Christian Guttmann for organising pre-prints for the day of the workshop.

We hope that you find this Fifth Australasian Ontology Workshop to be informative, thought-provoking, and most of all, enjoyable!

Thomas Meyer, Meraka Institute, South Africa

Kerry Taylor, CSIRO ICT Centre, Australia

Organisers of AOW 2009

November, 2009

Conference Organization

Programme Chairs

Thomas Meyer
Kerry Taylor

Programme Committee

Franz Baader
Michael Bain
Richard Booth
Arina Britz
Werner Ceusters
Michael Compton
Oscar Corcho
R. Cenk Erdur
Aurona Gerber
Dennis Hooijmaijers
Bo Hu
Renato Iannella
Ken Kaneiwa
C. Maria Keet
Kevin Lee
Laurent Lefort
Constantine Mantratzis
Lars Moench
Deshendran Moodley
Mehmet Orgun
Maurice Pagnucco
Debbie Richards
Rolf Schwitter
Murat Sensoy
Barry Smith
Markus Stumptner
Boontawee Suntisrivaraporn
Sergio Tessaris
Nwe Ni Tun
Ivan Varzinczak
Kewen Wang
Antoine Zimmermann

External Reviewers

Rinke Hoekstra
Rafael Penaloza
María del Carmen Suárez de Figueroa Baonza
Zhe Wang

Table of Contents

Balancing Expressivity and Implementability in OWL Ontologies for Semantic Data Frameworks: The Journey from 2004 to 2009 and Beyond

Peter Fox
Tetherless World Constellation Chair and Professor of Earth
and Environmental Science and Computer Science,
Rensselaer Polytechnic Institute, Troy, NY, USA

pfox@cs.rpi.edu

Abstract

In 2004, a small team of investigators undertook a prototype development effort to explore how semantics could be inserted in several existing scientific data systems being supported by the High Altitude Observatory at the National Center for Atmospheric Research. The problem to be solved was: discovery and access to interdisciplinary and heteroegeneous data sources without very detailed expert knowledge of the domain which included cryptic jargon (mnemonics). Ontology development expertise was included in the team but instead of a bottom-up or top-down approach to ontology development, we used a variant on the use case driven design to formalize vocabulary and relation requirements. We also had to use much of the existing infrastructures. Instead of a prototype the result was a production semantic data framework after about the first 9 months of the project. Several successive releases based on implemented use cases as well as an evaluation study led to some clear lessons in ontology development.

In this talk I will present the setting for this development effort, describe the use cases, experience with the ontology and language encoding choices, including software tools. Since 2007, we have carried these developments to a wider range of disciplines and I will also relate these recent experiences and consequences for ontology development including current and future directions with ontology modularization and OWL-2.

Visualizing and Specifying Ontologies using Diagrammatic Logics

Ian Oliver[1] **John Howse**[2] **Gem Stapleton**[2] **Esko Nuutila**[3] **Seppo Törmä**[3]

[1] Nokia Research, Finland
Email: ian.oliver@nokia.com

[2] Visual Modelling Group, University of Brighton, UK
Email: {John.Howse,g.e.stapleton}@brighton.ac.uk

[3] Helsinki University of Technology, Finland
Email: {esko.nuutila,seppo.torma}@hut.fi

Abstract

This paper proposes a diagrammatic logic that is suitable for specifying ontologies. We take a case study style approach to presenting the diagrammatic logic, and draw contrast with RDF graphs and description logics. We provide specifications of two ontologies and show how to depict instances. Diagrammatic reasoning is used to show that an instance conforms to a specification. We also include examples to show how diagrammatic rules can be used to (a) constrain ontology specifications and (b) define mappings between ontologies. The framework also allows the specification of queries. The positive features of the diagrammatic logic are discussed, supporting a claim that the new logic is intuitive and appropriate for ontology specification. Finally, we discuss the possibilities for developing tools to support the use of the diagrammatic logic, including automated diagram drawing and reasoning procedures.

1 Introduction

The need to specify ontologies frequently arises, with a prominent example being the semantic web area. Specifications can be provided symbolically, but many people find such notations inaccessible. Added to that, ontology construction and conceptualization can be difficult, and is hindered by the inaccessibility of the symbolic syntax available to the user. There are three main tasks that must be performed by users in the context ontology specification: (a) converting their semantic understanding into a specification in their chosen notation, (b) interpreting syntactic specifications created by themselves or others, and (c) reasoning about that specification to further understand its logical consequences. Of course, one can break these tasks down in to subtasks and add further tasks to this specified list. We argue that making the syntax more accessible to the user will aid with all three tasks. Indeed, the provision of a fit-for-purpose, more widely accessible notation specifically designed for specifying and reasoning about ontologies may be helpful to a large community of users.

Diagrammatic notations are potentially a viable alternative to symbolic notations. In the Description Logic Handbook (Baader et al. 2003), Nardi and Brachman state that a "major alternative for increasing the usability of description logics as a modeling language" is to "implement interfaces where the user can specify the representation structures through graphical operations." Moreover, we argue that using diagrammatic (graphical) notations for reasoning, in addition to specification, can bring huge benefits. Currently, some diagrammatic notations have been used for representing ontologies, but typically they are not formalized. For instance, (Brockmams et al. 2004) investigate using the UML for this purpose. However, the UML is not formal (some regard the UML as semi-formal) and was not designed for specifying ontologies. Dau and Ekland proposed using existential graphs for ontology modelling and established that they were capable of representing any DL statement made in $\mathcal{ALC}$ (Dau & Ekland 2008). Existential graphs have the flavour of a first-order logic which uses a minimal set of logical operators ($\wedge$ and $\neg$) along with existential quantification. Restricting to a minimal logic, such as this, can render it hard to use when formulating constraints that are naturally phrased using, for instance, disjunction and universal quantification. Thus, the syntax of existential graphs is very different from the ontology diagrams proposed in this paper.

We further note that standard ontology editors often support a visualization of the specified ontology. For instance, Protégé includes a plug-in visualization package, OWLVis, that provides a visualization in a graph-based form. This visualization shows derived hierarchical relationships between the concepts (classes) in the ontology and, thus, is very limited. We note that automatically generated visualizations from a given symbolic specification help with task (b) described above. However, we want to enable the users to perform the act of specification directly with a graphical notation.

In this paper, we propose a diagrammatic notation for specifying and reasoning about ontologies, called *ontology diagrams*. We consider various aspects of ontology specification, including the need to place constraints on ontologies and relate two or more ontologies. We also consider how ontology diagrams can be used in reasoning tasks. The design of the notation has been strongly informed by the tasks to be performed, including the semantic properties that are to be visualized using them. Of particular note is that we do not design the notation by aiming to have the same expressive power of any particular description logic. In our opinion, it would be inappropriate to design a graphical notation by aiming for the expressive

Oliver, Nuutila and Törmä were supported by the TEKES ICT SHOK DIEM project. Howse and Stapleton acknowledge the support of the EPSRC for the Visualization with Euler diagrams project (EP/E011160/1 and EP/E010393/1); the project web site can be found at www.eulerdiagrams.com. Stapleton further acknowledges EPSRC support for the Defining Regular Languages with Diagrams project (EP/H012311/1). We thank Sergey Boldyrev, Jukka Honkola and Pekka Luoma for helpful discussions on this research.

power of an existing symbolic notation. There is no reason to suppose that a natural design of any given notation should necessarily coincide with any other notation in terms of its expressiveness. Our primary goal is to obtain an effective notation, not a visualization of a particular description logic. Of course, it would be very interesting and important, once the notation has been fully formalized, to identify which description logic statements can be expressed by ontology diagrams.

The notation builds on previous work in the diagrams area. In particular, ontology diagrams use constraint diagrams (Kent 1997) as a basis, but we extend and modify the syntax to make it appropriate for the kinds of tasks described. Moreover, our semantic interpretation of ontology diagrams is not the same as constraint diagrams (Fish et al. 2005, Stapleton & Delaney 2008). We assign semantics that are appropriate for their use in ontology specification. Some of the differences between these notations are mentioned in the paper.

Section 2 illustrates ontology diagrams and how they can be used to specify information about concepts (concepts) and relationships between concepts (roles). Their use to visualize instances is discussed in section 3, where we illustrate how to show an instance is consistent with the specification using diagrammatic inference rules. Section 4 presents a schema for placing constraints on the ontology specification and queries are discussed in section 5. In section 6 we show how ontology diagrams can be used to relate two or more ontologies, allowing agents to reason about multiple ontologies, taking advantage of any semantic similarity between their respective concepts and roles. Further features of ontology diagrams are demonstrated in section 7 and approaches to formalization are discussed in section 8. Section 9 briefly discusses the future development tools to support the use of ontology diagrams. In section 10, we discuss cognitive theories that identify qualities of good diagrammatic notations, from a user perspective, linking the theories with ontology diagrams; this serves to justify their utility for ontology specification to some extent.

2 Ontology Specification

To illustrate our proposed ontology diagrams we present a case study, adapted from (Oliver et al. 2009), in which the notion of meetings, ontologies supporting those concepts and the interaction and mapping between the ontologies and agents were investigated. We will use this case study throughout this paper, illustrating a variety of features of the ontology diagram syntax and semantics.

The case study specifies a meeting ontology, introducing various concepts such as Location and Participant. Briefly, this ontology, called nMeeting, comprises seven concepts:

1. Meeting: this represents the notion of a meeting,
2. Location: every meeting will have a location (in this ontology, exactly one location),
3. Topic: every meeting will have at least one topic,
4. AgendaItem: every meeting will at least one item on its agenda,
5. Participant: every meeting will have at least one host and a set of participants (including the host),
6. Document: each agenda item will have a set of documents (possibly none),
7. Name: each participant will have a name.

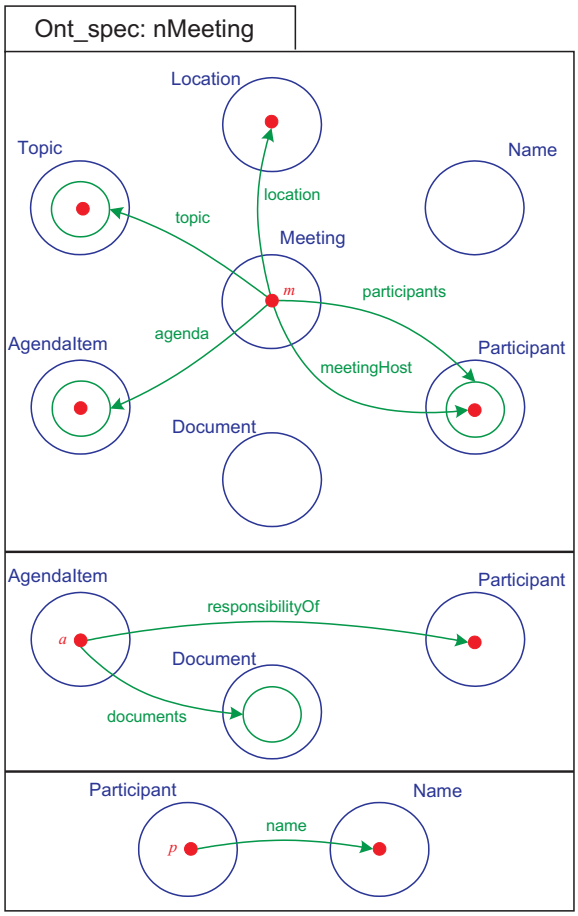


Figure 1: An ontology diagram specifying nMeetings.

Most of the roles in the ontology are evident from the description above. The only exception is that we will require each agenda item to be the responsibility of some participant. All of the concepts are pairwise disjoint.

One could produce alternative specifications of a meetings ontology. The purpose of this paper is to merely use the case study to illustrate some features of ontology diagrams, rather than present an 'ideal' model of such an ontology. This case study is intentionally simple, so that we can focus on pertinent features of ontology diagrams. Consequently, it is not rich enough to exemplify all aspects of the notation. Thus, section 7 presents further examples with features that do not arise from this case study.

The ontology diagram in figure 1 consists of 3 subdiagrams and captures relationships between the concepts, represented by labelled closed curves (circles are examples of closed curves). In each sub-diagram, the curves form an Euler diagram: an Euler diagram is a collection of labelled closed curves whose spatial relationships express semantic relationships between the concepts. Non-overlapping curves assert that the concepts are disjoint; a curve placed inside another asserts a subset relationship. We can immediately see from the top sub-diagram that all the concepts represented are pairwise disjoint since all the labelled curves have pairwise disjoint interiors; given that there are 7 concepts, this would require ${}^7C_2 = 21$ textual assertions (such as Meeting$\sqcap$Topic $\sqsubseteq \bot$), illustrating a certain succinctness of ontology diagrams. An obvious question asks how well ontology diagrams specify ontologies where not all of the concepts are disjoint; examples will be given in section 7.

The unlabelled dots (called unlabelled spiders) assert the existence of elements in the sets represented

by the regions of the diagram in which they are placed (further explanation is given below). The labelled spiders in this diagram are acting as free variables. So, m is a free variable that is 'talking' about meetings. Later, we will use labelled spiders to represent constants; the cases are distinguished by the use of italics for free variables. We note here that constraint diagrams do not include spiders that act as free variables.

The arrows in the ontology diagram are used to make statements about binary relationships between concepts, with their labels being analogous to roles in description logics. In the context of an ontology diagram that is specifying an ontology (like that in figure 1), the arrows are interpreted as providing domain and range (codomain) information; this feature is particular to ontology diagrams in that it is not part of constraint diagrams. In our nMeetings example, the arrow labelled topic informs us that there is a relation (or role) called topic between the concepts Meeting and Topic. Since the target of the arrow is an unlabelled curve that contains an unlabelled spider, we have asserted that each meeting (through the use of the free variable m) has a non-empty set of topics; the unlabelled curve represents the image of the relation topic when the domain is restricted to m. Using description logic syntax, this arrow, its source and target (including the spider inside the targeted curve), tells us:

$$\exists \mathsf{topic}.\top \sqsubseteq \mathsf{Meeting}$$
$$\exists \mathsf{topic}.\neg \mathsf{Topic} \sqsubseteq \bot$$
$$\mathsf{Meeting} \equiv \mathsf{Meeting} \sqcap \exists \mathsf{topic}.\mathsf{Topic}.$$

The location arrow targets an unlabelled spider and tells us that each Meeting has exactly one Location; thus, location is a function with domain Meeting and range Location. We observe that the ontology diagram specifies that (the functional role) meetingHost is a subrole of participants, since the host of the meeting must be one of the meeting participants; equivalently meetingHost $\sqsubseteq$ participants. In section 7 we show how to assert subrole relationships where neither of the roles are necessarily functional.

We now summarize the semantics of the top subdiagram: all represented concepts are pairwise disjoint, and every member, m, has exactly one location, a non-empty set of topics, a non-empty set of agenda items, a meeting host, and a set of participants which includes the meeting host. The other two sub-diagrams make further assertions about binary relations and functions. For instance, the bottom diagram tells us that the function name returns the Name of each Participant.

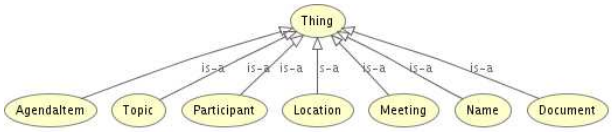


Figure 2: Output from Protégé illustrating the nMeeting hierarchy.

As mentioned in the Introduction, Protégé (*Protege web site* accessed June 2009) includes some support for the visualization of ontologies via OwlViz (Horridge accessed June 2009). Figure 2 illustrates the Protégé output obtained for the nMeeting hierarchy, but it only shows the concepts and the fact that they are all Things. The only semantic information in the output diagram is given by the names of the concepts and the direction of the is-a arrows. This diagram includes no information about the disjointness of these concepts or the properties between them; the disjointness cannot be inferred from the diagram at all, though it does appear so.

3 Instances

An instance of the nMeetings ontology is in figure 3. Here, the spiders are all labelled and they represent specific objects (analogous to constants). Notice that the ontology instance diagram contains shading in the image of the relation topic. Semantically, the shading places an upper bound on set cardinality: the Meeting m_1 is related to all and only the elements represented by the spiders in the unlabelled curve targeted by the topic arrow. In an ontology instance diagram, the arrows are providing information about properties of the relations that their labels represent, and are not interpreted as providing domain and range information (which is their role in an ontology specification diagram).

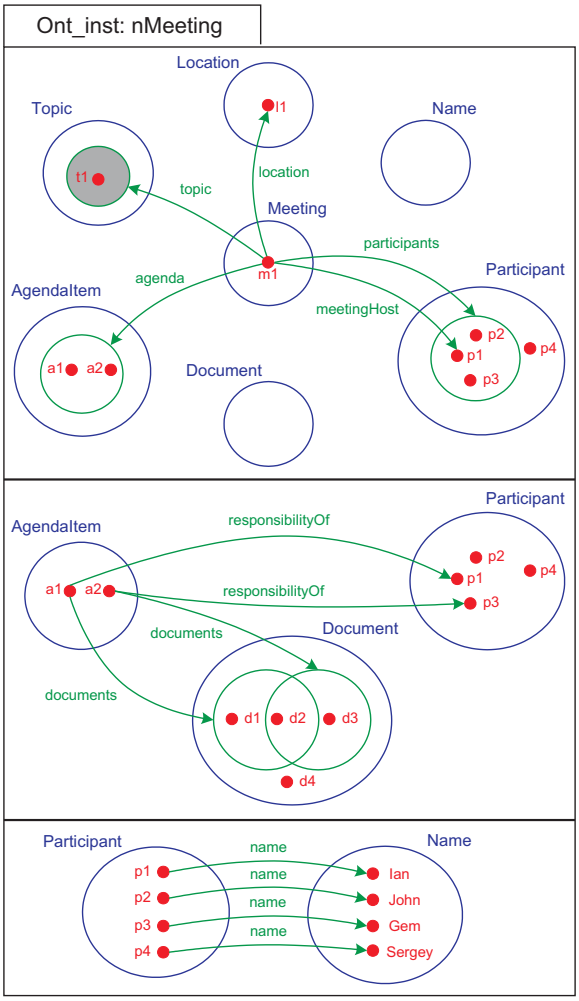


Figure 3: An instance of the nMeeting ontology.

The given ontology instance diagram tells us that the meeting m_1 has exactly one topic, namely t_1. We can also see that m_1 has at least two items on its agenda, namely a_1 and a_2 (distinct spiders are taken to denote distinct objects). The middle sub-diagram shows that the two agenda items have a document, d_2, in common. Moreover, there is a document d_4 that is not in the set of documents associated with a_1 or a_2. The rest of the diagram is similarly interpreted.

The instance diagram is not entirely shaded, so there could be more objects than those represented by the labelled spiders. Consequently, we only have

partial information about the ontology instance. For example, there *may* be more agenda items than the two represented. For comparison purposes, an RDF graph for the same instance can be seen in figure 4. However, we cannot infer, for instance, from the RDF graph that a_1 and a_2 are different agenda items, whereas the ontology diagram in figure 3 does provide this information; this is due to the lack of expressive power of RDF.

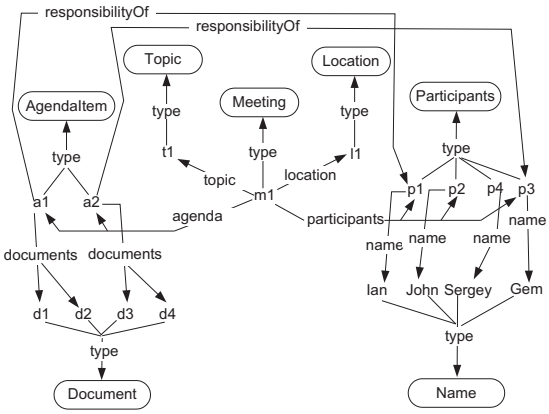


Figure 4: RDF graph of the nMeeting instance

We can prove that this instance is consistent with the specification by using diagrammatic inference rules. To show that it is consistent, we can establish that each sub-diagram of the instance conforms to each respective sub-diagram in the specification. We illustrate how this is done for the top sub-diagram, which is D_1 in figure 5. To do this, we establish that the meeting, m_1, is related to entities as specified in figure 1. We are showing that the only member we know about (m_1) has the properties that a member should have according to the specification.

We first observe that, since shading provides an upper bound on set cardinality, a sound inference rule allows us to delete shading (thus 'forgetting' the upper bound). This is shown as the first step in the proof shown in figure 5, where we derive D_2 from D_1. Next, we notice that if we have a labelled spider, s, placed in a region of the diagram then we can delete that spider (provided the region is not shaded); this is because a labelled spider placed in a region tells us that the represented object (or individual) is in the set represented by that region and, thus, deleting the spider forgets this information. Therefore, we can delete spiders from D_2 to give D_3. Finally, we turn the remaining labelled spiders (except m_1), into unlabelled spiders, giving D_4. This is sound since if we know a specific object has a particular set of properties then there exists an object with those properties. We see that m_1 in D_4 now has all the properties that a member is specified to have in figure 1.

Similar proofs can be constructed for the other two sub-diagrams. Notice that the middle sub-diagram has two AgendaItems and we are required to show each of them has the necessary properties; similarly the bottom sub-diagram has four Participants. In any case, we can use these types of inference rules to prove that the instance conforms to the specification. Inference rules similar to those described here have been defined for constraint diagrams (Stapleton et al. 2005); their formalisation should extend to ontology diagrams but, since the semantics are not identical, we would be obliged to prove their soundness when applied to ontology diagrams.

We can also use diagrammatic inference rules to explore properties of our instance. In addition to being able to delete shading and spiders, we can also delete labelled curves and arrows without increasing the informational content of a diagram (that is, such inference rules are sound). In figure 6, we can deduce D_6 from D_5 (which is the top sub-diagram in figure 3) by deleting syntax. Next, we can use the information from the middle sub-diagram in figure 3 to add pieces of syntax as follows. We can add documents d_1, d_2, d_3 and d_4 to D_6, since the middle sub-diagram tells us that they are Documents, giving D_7. Finally, we can add arrows, again using information in the middle sub-diagram, to yield D_8. Alternative proofs could be constructed to achieve the same outcome; for instance, deleting the shading from D_5 could be delayed until the last step of the proof.

This type of reasoning provides a way for users to explore, and understand, the consequences of an instance. We can also define similar inference rules that allow us to explore the consequences of ontology

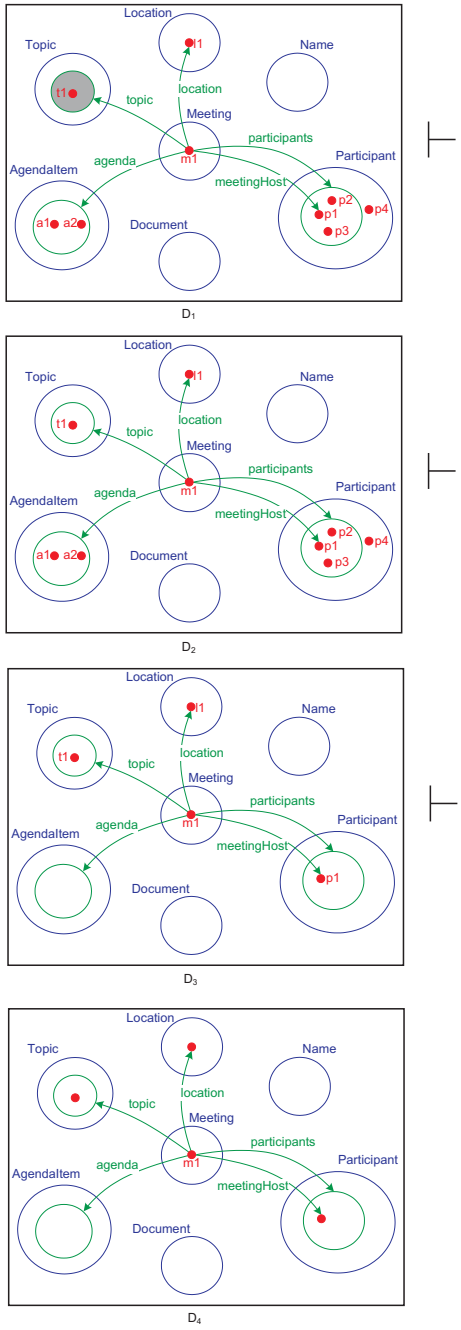


Figure 5: Proving that the instance is consistent with the specification.

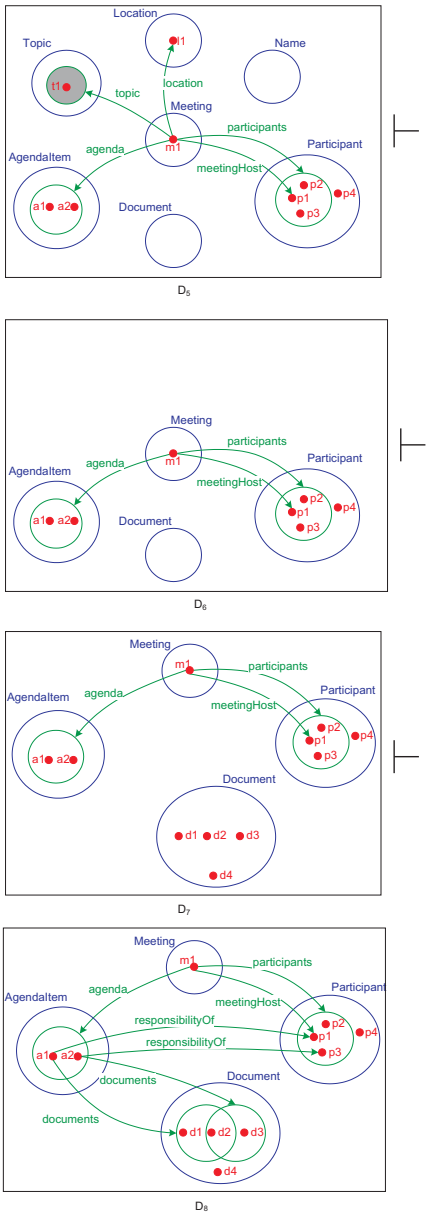


Figure 6: Deductions from an instance.

specification diagrams (like that in figure 1) which is important: understanding the consequences can reveal inconsistencies or unintended consequences. This understanding can lead to refinements or improvements to the specification which is clearly desirable. Of course, we can always convert a reasoning task into a problem stated symbolically, and reason at the symbolic level (provided we have a mechanism to construct an appropriate translation). We believe that applying diagrammatic inference rules to construct reasoned arguments will facilitate a greater understanding in users of why inferences hold (or, even, why intended consequences do not follow). Thus, the motivation for developing diagrammatic inference rules is primarily driven by users.

4 Constraints on Ontologies

In addition to specifying an ontology, we may want to enforce constraints on that ontology. In the nMeeting example, we probably want to assert that if a Participant is responsible for an AgendaItem then that Participant is one of the Meeting participants; this information is not deducible from the ontology specification diagram in figure 1 and must be enforced by a constraint. In order to allow this type of constraint to be imposed on ontologies, we introduce a constraint rule schema.

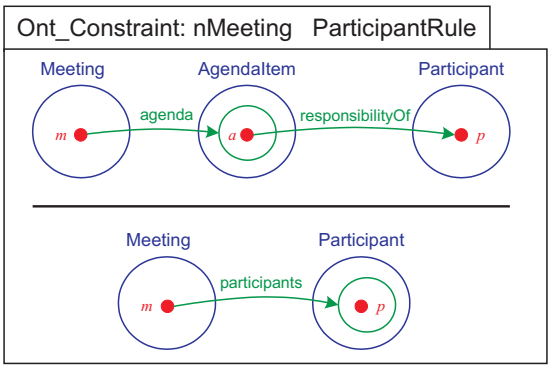


Figure 7: The constraint rule schema.

Figure 7 illustrates the schema (which is following the presentation style of a natural deduction rule), with a constraint called ParticipantRule. The diagram asserts that if a is an AgendaItem, on the agenda of Meeting m and a is the responsibilityOf Participant p (above the line) then it must be the case that the set of m's participants includes p (below the line). In this example, the labelled spiders are acting as free variables, not constants. We may want to use constants in a constraint rule schema. When formalizing the logic, one would know which spider labels are variables and which are constant since these sets would be pre-defined. Recall that, in order to differentiate them in drawn diagrams, we have adopted the convention that free variables, unlike constants, are in italics. Further investigations are needed to establish any relationship between the rules that can be defined in this type of schema and role chains and role subsumption in OWL 2 and any relationship that may exist with OWL with SWRL.

5 Querying Ontologies

It is useful to be able to specify queries over ontologies. Ontology diagrams can also be used for this purpose. As an example, we may want to obtain the set of documents associated with the agenda items for some given meeting. Such a query, called getDocuments is defined in figure 8; the notation getDocuments(m)$\rightarrow D$ provides the name of the query, the required inputs (in this case, m), and the output (in this case D). Notice here that the source of the documents arrow is an unlabelled curve. This unlabelled curve represents the set of agenda items associated with the meeting m; more formally, it represents the image, I, of the relation agenda when its domain is restricted to m. Continuing in this manner, the set D is the image of the relation documents when the domain is restricted to I; the query returns the set D. We show how to represent more complex queries

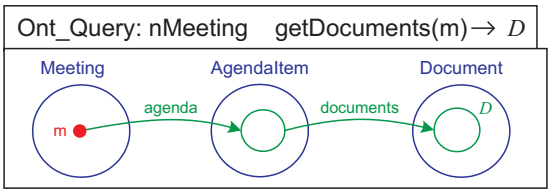


Figure 8: Query.

in section 7. Future work will involve establishing the relationship between these query diagrams and SPARQL (Perez et al. 2006).

6 Ontology Mapping

There are likely to be times when multiple ontologies have been defined that contain similar or identical concepts and roles. If we want agents to be able to reason about these ontologies, taking into account the similar concepts and roles, then we need to be able to specify the commonality. Typically, this is done via ontology mapping: providing a layer from which we can access multiple ontologies. That is, we can take advantage of semantic similarity to enable interoperability between the various ontologies (Budanitsky & Hirst 2006, Lindsey et al. 2007). This is possible by (a) specifying how the concepts and roles in the ontologies relate, and (b) by providing additional constraints that 'tie up' the two ontologies more precisely. Here, we show how ontology mapping is possible using ontology diagrams and, to this end, we introduce a second ontology specification for devices and presentations; see figure 9.

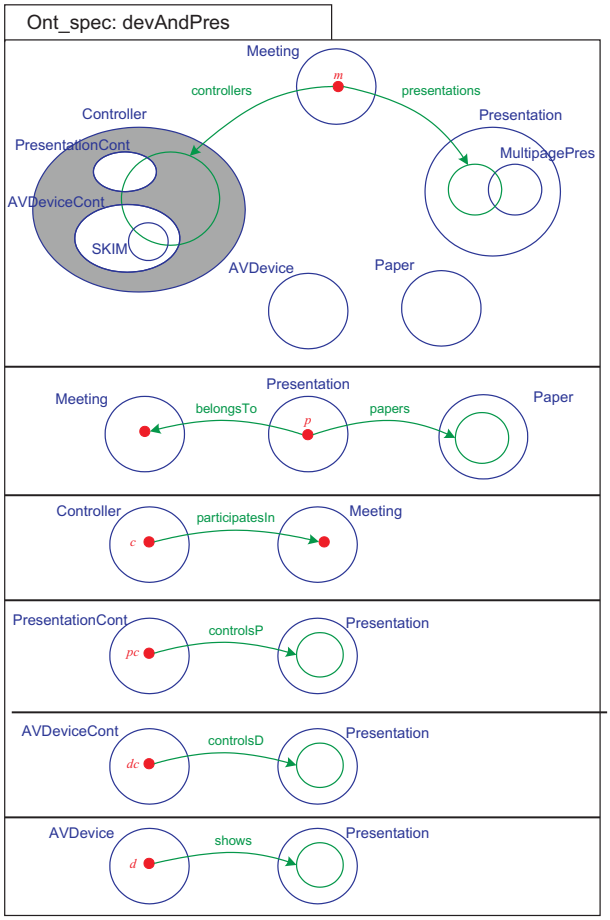


Figure 9: An ontology diagram for devices and presentations.

The concept AVDevice represents things that are able to present audiovisual content, such as a video projector, a display of a laptop, a loudspeaker, a window on a computer desktop, and so forth. The concept AVDeviceController represents a software component that can render output to an `AVDevice`. The concept SKIM is an example subconcept of AVDeviceController. It represents a software component that uses the SKIM viewer (*Skim, PDF reader and note-taker for OS X*. n.d.) to show a PDF document on a display. The concept Controller is just an abstraction of the controller components (i.e. it is an abstract concept), hence the shading inside the respective curve but outside the labelled curves which it contains.

When considering two (or more) ontologies, in order to distinguish the use of common names for concepts or roles (which we may or may not intend to have a common or similar interpretation), we prefix them by the name of the ontology from which they are derived. So, in the case of Meeting, which occurs in both of our example ontology specifications, we write nMeeting:Meeting and devAndPres:Meeting, to differentiate between the two uses of the common concept name. Similarly, we can choose to write nMeeting:AgendaItem, nMeeting:agenda, and so forth, to identify the ontology from which the concept or role derived even though the names do not appear in both of the original ontologies, should this be deemed helpful to us.

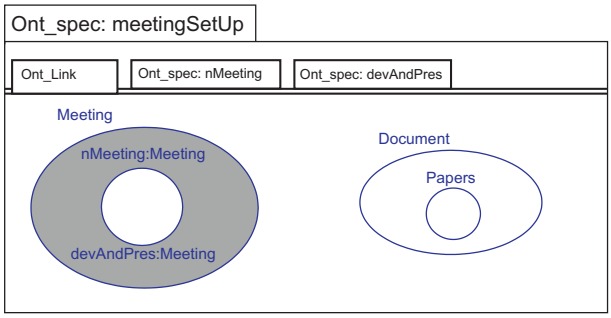


Figure 10: Relating ontologies.

When relating the two ontologies, we may intend for nMeeting:Meeting and devAndPres:Meeting to be interpreted as the same concept. In addition, we may want to assert that Paper (from devAndPres) is a sub-concept of Document (from nMeeting). These two relationships are illustrated in figure 10 which specifies the ontology meetingSetUp which relates the nMeeting and devAndPres ontologies. The 'equality' of nMeeting:Meeting and devicesAndPresentations:Meeting is asserted by drawing their respective curves on top of one another. Notice that we have drawn a curve labelled Meeting (with shading), so that we can simply use Meeting in place of either nMeeting:Meeting or devAndPres:Meeting to simplify matters. We could also choose to express further information about the relationships between other concepts, but for space reasons we have chosen only to define the relationships shown in figure 10. We can also use this framework to specify relationships between roles in the Ont_Link diagram (using the sub-diagram style) where appropriate.

Notice that figure 10 includes tabs, showing that our 'super-ontology' meetingSetUP includes links between ontologies (the diagram shown), and relates the nMeeting and devAndPres ontologies (the tabs with hidden diagrams). With appropriate tool support, one could click on the tab to reveal the original ontologies, since these form part of the meetingSetUp ontology.

Our framework further allows the specification of constraints on the super-ontology, like that in figure 11. The diagram asserts that if m is a meeting whose agenda items are associated with set of documents D then the papers associated with m's presentations form a subset of D. This constraint uses concepts and roles from both of the original ontologies. We can go on to define additional queries, visualize instances, and reason about the specification and instances, using the same approaches as illustrated for the nMeetings ontology.

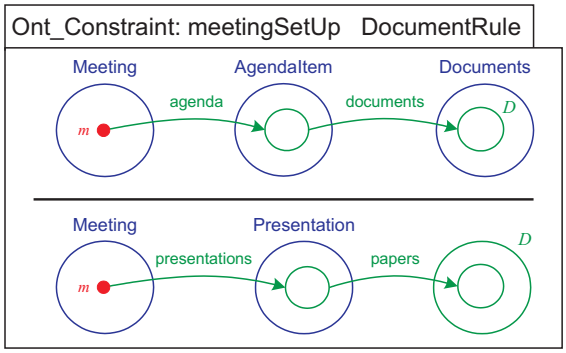


Figure 11: Constraints on the super-ontology.

7 Further Features of Ontology Diagrams

Ontology diagrams include additional syntax not yet demonstrated in the paper. For instance, they include syntax to assert equality between elements, which could be named individuals (i.e. constants), existentially quantified elements, or those elements represented by spiders acting free variables. The syntax used looks very much like an equals sign, which was chosen since people are generally very familiar with this symbol. An example is given in figure 12 which asserts that the functional role f is injective using the constraint rule schema: if x is related to a and y is related to a then $x = y$.

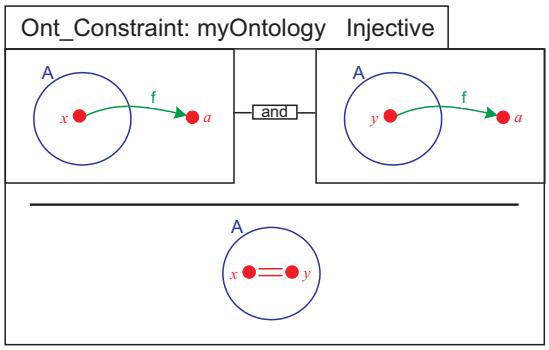


Figure 12: Asserting equality.

We note that description logics have a unique name assumption, that is different names imply different entities. In ontology diagrams different names imply different entities unless connected by an equals sign. By contrast, first order predicate logic assumes that entities may or may not be equal and one must assert either equality or distinctness whenever one or the other is intended. Similarly, OWL does not make the unique name assumption, but includes constructs to express equality or distinctness, namely OWL:sameAs and OWL:differentFrom.

A further feature of ontology diagrams can be seen in figure 13. The spider is placed in more than one (minimal) region of the diagram. Here, x represents an element in $A \cup B$ and, therefore, conveys disjunctive information: $x \in A - B \vee x \in A \cap B \vee x \in B - A$. Disjunctive statements can also be made using logical connectives between diagrams. We have already seen the use of conjunction between diagrams: figure 1 contains three sub-diagrams whose semantics are taking in conjunction. An example of how to represent disjunction, essentially following the style used in (Shin 1994), can be seen in figure 14. In this constraint rule, the two subdiagrams above the line are taken in disjunction. Other logical operators can be used in a similar fashion, although in fact we

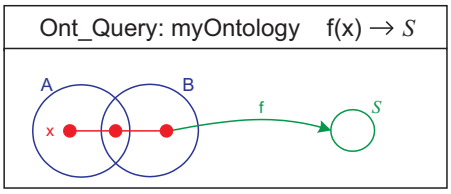


Figure 13: Making disjunctive statements within diagrams.

have been using juxtaposition to assert conjunction between subdiagrams such as in figure 1. The diagram in figure 14 also shows how subrole assertions can be made when the roles are not necessarily functional: the subdiagram below the line asserts that f is a subrole of g, when its domain is restricted to x.

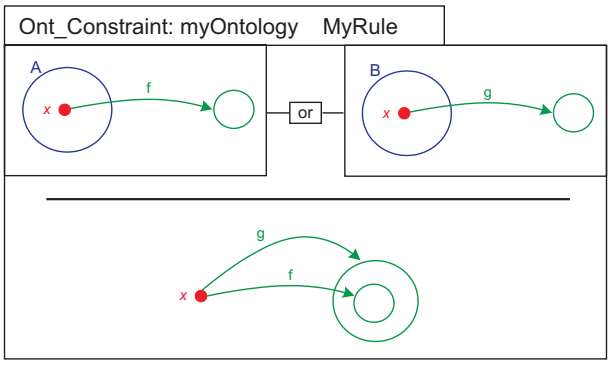


Figure 14: Making disjunctive statements between diagrams.

Figure 15 shows an example of a specification where none of the concepts are disjoint. The arrow between the two sub-diagrams allows us to asserts that there is a relation, g, with domain A and range B ∩ C. Having arrows between sub-diagrams was first explored in (Howse & Stapleton 2008) precisely to allow less cluttered diagrams to be produced. Figure 16 shows an alternative representation of the same information that does not utilize arrows between sub-diagrams. As with any notation, semantically equivalent statements exist and some are more effective at conveying information than others.

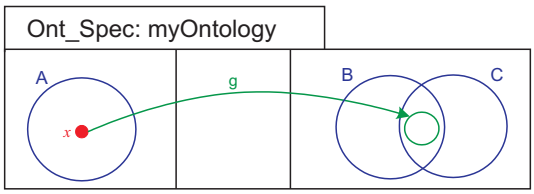


Figure 15: Non-disjoint concepts: arrows between sub-diagrams.

8 Approaches to Formalization

There has been considerable progress, over the last decade or so, on approaches to formalizing visual languages. With respect to formalizing the syntax, we advocate the distinction between concrete and abstract syntax. The concrete syntax captures the physical representation of the diagram whereas the abstract syntax is a mathematical description of the diagram that captures the pertinent (semantically important) features and disregards semantically irrele-

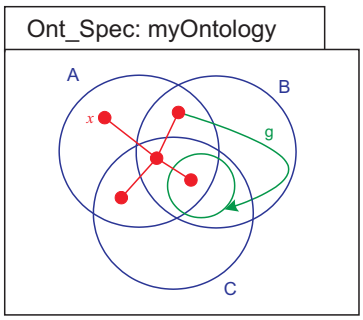


Figure 16: Non-disjoint concepts: a single diagram.

vant geometric properties. One can think of the concrete syntax as being the physical diagram itself and the advantages of this two-tiered approach are discussed in (Howse et al. 2001). The remainder of this section presents preliminary work on the formalization of ontology diagrams for the interested reader and is not necessary for the following sections.

First, we present the abstract syntax for so-called 'unitary' sub-diagrams. These are the diagrams that form the building blocks of more complex diagrams. For space reasons, we do not include a formalization of the syntax of the entire notation. We start by defining various sets:

1. $\mathcal{LC}$ is a set whose elements are used to label curves in concrete diagrams.
2. $\mathcal{UC}$ is a set whose elements correspond to unlabelled curves in concrete diagrams (note that the elements are not themselves curves).
3. $\mathcal{Z}$ is the set of **zones** defined by
$$\mathcal{Z} = \left(\mathbb{P}(\mathcal{LC} \cup \mathcal{UC})\right)^2.$$
4. $\mathcal{CS}$ is a set whose elements are used to label constant spiders in concrete diagrams.
5. $\mathcal{US}$ is a set whose elements correspond to unlabelled spiders in concrete diagrams.
6. $\mathcal{FS}$ is a set whose elements are used to label free spiders (those acting as free variables) in concrete diagrams.
7. $\mathcal{S} = \mathcal{CS} \cup \mathcal{US} \cup \mathcal{FS}$.
8. $\mathcal{AL}$ is a set whose elements are used to label arrows in concrete diagrams.
9. $\mathcal{A}$ is the set of **arrows** defined by
$$\mathcal{A} = \{(s, l, t) : s, t \in \mathcal{LC} \cup \mathcal{UC} \cup \mathcal{S} \wedge l \in \mathcal{AL}\}.$$

We assume that all of the above sets are pairwise disjoint. The definition of a unitary diagram draws its components from these sets; we explain the definition via an example immediately below.

Definition 8.1 *An **abstract unitary ontology sub-diagram** is a tuple $(LC, UC, Z, SZ, CS, US, FS, \eta, \delta, A)$ whose components are all finite sets defined as follows.*

1. *$LC \subseteq \mathcal{LC}$ and $UC \subseteq \mathcal{UC}$*
2. *$Z \subset \mathcal{Z}$ such that each zone $(in, out) \in Z$ satisfies $in \cup out = LC \cup UC$.*
3. *$SZ \subseteq Z$ is a set of **shaded zones**.*
4. *$CS \subseteq \mathcal{CS}$, $US \subseteq \mathcal{US}$ and $FS \subseteq \mathcal{FS}$*
5. *η is a function with domain S, where $S = CS \cup US \cup FS$, and range $\mathbb{P}Z - \{\emptyset\}$.*
6. *δ is a partial function which identifies whether two spiders are joined by $=$. It has domain $(S \times S)$ and codomain $\{0, 1\}$ and is defined precisely for the pairs (s_1, s_2) where $s_1 \neq s_2$ and it is symmetric.*
7. *$A \subseteq \mathcal{A}$ such that for each $(s, l, t) \in A$, s and t are both in $LC \cup UC \cup CS \cup US \cup FS$.*

For example, figure 12 contains three unitary sub-diagram. The top left sub-diagram has abstract syntax comprising (the omitted components of the tuple are all empty):

1. $LC = \{\mathsf{A}\}$,
2. $Z = \{(\{\mathsf{A}\}, \emptyset), (\emptyset, \{\mathsf{A}\}),$
3. $FS = \{x, a\}$,
4. $\eta(x) = \{(\{\mathbf{A}\}, \emptyset)\}$ and $\eta(a) = \{(\emptyset, \{\mathsf{A}\})\}$,
5. $\delta(x, a) = \delta(a, x) = 0$ (the spiders are not joined by $=$), and
6. $A = (x, \mathsf{f}, a)$.

Turning to the semantics, typical approaches in diagrammatic logics mirror those found in more traditional symbolic logics. We start by defining a structure, $(U, \Psi_0, \Psi_1, \Psi_2)$ where U is a universal set,

1. Ψ_0 maps elements of $\mathcal{CS}$ to elements of U,
2. Ψ_1 maps elements of $\mathcal{LC}$ to subsets of U, and
3. Ψ_2 maps elements of $\mathcal{AL}$ to binary relations on U.

Given a structure, one can then define when the structure satisfies a diagram as has been done for constraint diagrams (Fish et al. 2005, Stapleton & Delaney 2008).

A complete formalization of the syntax and semantics of ontology diagrams remains the subject of future work. The initial ideas presented here are based on over a decade of work formalizing Euler, spider and constraint diagrams on which ontology diagrams build. For examples, see (Stapleton et al. 2007), (Gil et al. 1999, Howse et al. 2005, Stapleton, Taylor, Howse & Thompson 2009), and (Stapleton et al. 2005, Stapleton & Delaney 2008, Fish et al. 2005) for formalizations of Euler diagrams, spider diagrams and constraint diagrams respectively.

9 Tool Support

Significant tool support has been developed for using symbolic notations to specify and reason about ontologies, such as (*FaCT++* accessed June 2009, *Protege web site* accessed June 2009), including functionality for visualizing aspects of the ontologies (Horridge accessed June 2009). However, the visualizations available to the users are not as sophisticated as those possible with the notations proposed in this paper. It is possible to provide tool support for ontology diagrams. Key pieces of functionality include:

1. The ability to input ontology diagrams via an editor or sketch recognition system.
2. The ability to automatically translate ontology diagrams into symbolic forms (such as OWL or Description Logics) to enable us to take advantage of the significant tool support that has been developed to date, including highly efficient reasoners. Moreover, it is desirable to support the translation of symbolic statements into ontology diagrams, permitting their visualization.

3. The provision of a proof assistant or automated theorem prover which can be used to allow users to explore the logical consequences of their ontology diagrams.

4. The ability to automatically generate ontology diagrams, in particular to support automated reasoning and visualization of symbolic statements.

In the latter case above, significant research has been directed towards the automated generation and layout of Euler diagrams, which form the basis of ontology diagrams, including (Chow & Ruskey 2003, Flower & Howse 2002, Stapleton, Rodgers, Howse & Zhang 2009, Stapleton, Howse & Rodgers 2009, Verroust & Viaud 2004). These diagram generation tools typically take as input the abstract syntax of the to-be-generated diagram. Already, theorem provers have been developed for Euler diagram (Stapleton et al. 2007) and spider diagrams (Flower et al. 2004). Thus, whilst significant work is required to develop tool support for ontology diagrams, there is already a firm basis on which we can build. We plan to develop tools as part of our future work, possibly as a plug-in for Protégé.

10 Free-rides and Well-matchedness

Some of the benefits of diagrammatic notations are evident in figure 9, where both set intersection, disjointness and containment are represented visually: for example, Meeting and Controller assert the disjointness of the represented concepts since these two curves do not overlap, and PresentationCont represents a subset of Controller asserted by the inclusion of the former curve inside the latter. This diagram has properties that are thought to correlate with areas where diagrams are superior to symbolic notations, from a user interpretation perspective, because it is well-matched to its set-theoretic semantics (Gurr 2001). Extending this observation, using containment to represent set inclusion has the added benefit that the transitive property of the (semantic) subset relation is mirrored by the transitive property of (syntactic) containment. Any notation that is based on Euler diagrams to make such statements about sets is well-matched to its semantics. Thus, Euler diagrams are a good basis for ontology diagrams.

The economy of syntax afforded by diagrams over symbolic notations is also sometimes an advantage. In Fig. 9, the relative placement of the Meeting, PresentationCont and Controller curves gives, for free, that 'PresentationCont is disjoint from Meeting'. This example of a *free ride*, the theory of which is developed by Shimojima (Shimojima 2004), is an instance of where the explicit information in a diagram includes facts that would need to be inferred in the symbolic case. Other types of free rides arise and are not solely an advantage of Euler diagrams; for example, see the discussions on various types of free rides in constraint diagrams that relate to their arrows (Howse & Stapleton 2008); many of the free rides exhibited by constraint diagrams extend to ontology diagrams due to the similarity of their syntax. This type of inferential advantage of diagrams has been noted by several researchers, including (Barwise & Etchemendy 1990, Stenning & Lemon 2001), and is backed up by empirical evidence provided in (Shimojima & Katagiri 2008). The advantages of diagrams in numerous reasoning contexts are further discussed in (Larkin & Simon 1987).

11 Conclusion

In this paper we have proposed a new diagrammatic logic, *ontology diagrams*, for specifying and reasoning about ontologies. We have argued that these diagrams are well-matched to their semantics and, therefore, have advantages over symbolic notations that are currently on offer. The visual nature of the syntax may make ontology diagrams more appealing to non-mathematically minded people who have a need to specify ontologies. Moreover, they may provide a more accessible means of communicating an ontology specification to a variety of stakeholders (not just those who are familiar with current mechanisms to define ontologies).

The expressiveness required of the description logic capable of specifying the two ontologies presented here is relatively simple: in the meeting ontology it is $\mathcal{ALCHQ}_{(\mathcal{D})}$ and the device/presentation ontology it is $\mathcal{ALCF}_{(\mathcal{D})}$. This level of expressiveness means that any reasoning that is required to be made over these ontologies is reasonably simple and does not require the power of OWL[1] reasoner. We note that most current business information models have been specified using entity-relationship models which are easily mapped to simple description logics of much less complexity than OWL (Baader et al. 2003). We note here that, without the arrows present, ontology diagrams are essentially spider diagrams (Howse et al. 2005), a notation which is equivalent in expressiveness to monadic first order logic with equality (Stapleton et al. 2004). For ontology diagrams, it remains the subject of future work to establish their expressive power relative to description logics.

In the future, we plan to formalize the syntax and semantics of ontology diagrams, following the style used for constraint diagrams (Stapleton & Delaney 2008). This formalization will then allow us to define inference rules for ontology diagrams and prove their soundness and, ideally, completeness. We will carefully design the inference rules, using a wide variety of case studies to inform us of the kinds of reasoning that takes place. We would aim to define rules that are intuitive to human users, so that people can better understand why entailments hold. This complements current work on computing justifications (Horridge et al. 2009) which aims to produce minimal sets of axioms from which an entailment holds; finding minimal sets allows users to focus on the information that is relevant to the deduction in question which is important when dealing with ontologies containing very many concepts. Using a visual syntax with which to communicate *why* the entailment holds (i.e. providing a diagrammatic proof) may allow significant insight beyond merely knowing the axioms from which a statement can be deduced.

References

Baader, F., Calvanese, D., McGuinness, D., Nadi, D. & (eds), P. P.-S. (2003), *The Description Logic Handbook*, CUP.

Barwise, J. & Etchemendy, J. (1990), *Logical Reasoning with Diagrams*, OUP, chapter Visual Information and Valid Reasoning.

Brockmams, S., Volz, R., Eberhart, A. & Löffler, P. (2004), Visual modeling of owl dl ontologies using UML, *in* 'International Semantic Web Conference', Springer, pp. 198–213.

[1] http://www.w3.org/TR/owl-features/

Budanitsky, A. & Hirst, G. (2006), 'Evaluating wordnet-based measures of lexical semantic relatedness', *Computational Linguistics* **32**(1), 13–47.

Chow, S. & Ruskey, F. (2003), Drawing area-proportional Venn and Euler diagrams, *in* 'Proceedings of Graph Drawing 2003, Perugia, Italy', Vol. 2912 of *LNCS*, Springer-Verlag, pp. 466–477.

Dau, F. & Ekland, P. (2008), 'A diagrammatic reasoning system for the description logic $\mathcal{ACL}$', *Journal of Visual Languages and Computing* **19**(5), 539–573.

FaCT++ (accessed June 2009)http://owl.man.ac.uk/factplusplus/.

Fish, A., Flower, J. & Howse, J. (2005), 'The semantics of augmented constraint diagrams', *Journal of Visual Languages and Computing* **16**, 541–573.

Flower, J. & Howse, J. (2002), Generating Euler diagrams, *in* 'Proceedings of 2nd International Conference on the Theory and Application of Diagrams', Springer, Georgia, USA, pp. 61–75.

Flower, J., Masthoff, J. & Stapleton, G. (2004), Generating readable proofs: A heuristic approach to theorem proving with spider diagrams, *in* 'Proceedings of 3rd International Conference on the Theory and Application of Diagrams', Vol. 2980 of *LNAI*, Springer, Cambridge, UK, pp. 166–181.

Gil, J., Howse, J. & Kent, S. (1999), Formalising spider diagrams, *in* 'Proceedings of IEEE Symposium on Visual Languages (VL99), Tokyo', IEEE Computer Society Press, pp. 130–137.

Gurr, C. (2001), Aligning syntax and semantics in formalisations of visual languages, *in* 'Proceedings of IEEE Symposia on Human-Centric Computing Languages and Environments', IEEE Computer Society Press, pp. 60–61.

Horridge, M. (accessed June 2009), 'Owlviz', www.co-ode.org/downloads/owlviz/.

Horridge, M., Parsia, B. & Sattler, U. (2009), Computing explanations for entailments in description logic based ontologies, *in* '16th Automated Reasoning Workshop'.

Howse, J., Molina, F., Shin, S.-J. & Taylor, J. (2001), Type-syntax and token-syntax in diagrammatic systems, *in* 'Proceedings FOIS-2001: 2nd International Conference on Formal Ontology in Information Systems, Maine, USA', ACM Press, pp. 174–185.

Howse, J. & Stapleton, G. (2008), Visual mathematics: Diagrammatic formalization and proof, *in* 'International Conference on Mathematical Knowledge Management', Springer, pp. 478–493.

Howse, J., Stapleton, G. & Taylor., J. (2005), 'Spider diagrams', *LMS Journal of Computation and Mathematics* **8**, 145–194.

Kent, S. (1997), Constraint diagrams: Visualizing invariants in object oriented modelling, *in* 'Proceedings of OOPSLA97', ACM Press, pp. 327–341.

Larkin, J. & Simon, H. (1987), 'Why a diagram is (sometimes) worth ten thousand words', *Journal of Cognitive Science* **11**, 65–99.

Lindsey, R., Veksler, V., Grintsvayg, A. & Gray, W. (2007), Be wary of what your computer reads: The effects of corpus selection on measuring semantic relatedness, *in* '8th International Conference on Cognitive Modeling'.

Oliver, I., Nuutila, E. & Törmä, S. (2009), Context gathering in meetings: Business processes meet the agents and the semantic web, *in* 'The 4th International Workshop on Technologies for Context-Aware Business Process Management (TCoB 2009)'.

Perez, J., Arenas, M. & Gutierrez, C. (2006), Semantics and complexity of sparql, *in* 'International Semantic Web Conference', Springer, pp. 30–43.

Protege web site (accessed June 2009), http://protege.stanford.edu/.

Shimojima, A. (2004), Inferential and expressive capacities of graphical representations: Survey and some generalizations, *in* 'Proceedings of 3rd International Conference on the Theory and Application of Diagrams', Vol. 2980 of *LNAI*, Springer, Cambridge, UK, pp. 18–21.

Shimojima, A. & Katagiri, Y. (2008), An eye tracking study of spatial constraints in diagrammatic reasoning, *in* '5th International Conference on the Theory and Application of Diagrams', Springer, pp. 64–88.

Shin, S.-J. (1994), *The Logical Status of Diagrams*, Cambridge University Press.

Skim, PDF reader and note-taker for OS X. (n.d.). http://skim-app.sourceforge.net/.

Stapleton, G. & Delaney, A. (2008), 'Evaluating and generalizing constraint diagrams', *Journal of Visual Languages and Computing* **19**(4), 499–521.

Stapleton, G., Howse, J. & Rodgers, P. (2009), 'A graph theoretic approach to general Euler diagram drawing', *Accepted for Theoretical Computer Science* .

Stapleton, G., Howse, J. & Taylor, J. (2005), 'A decidable constraint diagram reasoning system', *Journal of Logic and Computation* **15**(6), 975–1008.

Stapleton, G., Masthoff, J., Flower, J., Fish, A. & Southern, J. (2007), 'Automated theorem proving in Euler diagrams systems', *Journal of Automated Reasoning* **39**, 431–470.

Stapleton, G., Rodgers, P., Howse, J. & Zhang, L. (2009), 'Inductively generating Euler diagrams', *accepted for IEEE Transactions on Visualization and Computer Graphics* .

Stapleton, G., Taylor, J., Howse, J. & Thompson, S. (2009), 'The expressiveness of spider diagrams augmented with constants', *Journal of Visual Languages and Computing* **20**, 30–49.

Stapleton, G., Thompson, S., Howse, J. & Taylor, J. (2004), 'The expressiveness of spider diagrams', *Journal of Logic and Computation* **14**(6), 857–880.

Stenning, K. & Lemon, O. (2001), 'Aligning logical and psychological perspectives on diagrammatic reasoning', *Artificial Intelligence Review* **15**(1-2), 29–62.

Verroust, A. & Viaud, M.-L. (2004), Ensuring the drawability of Euler diagrams for up to eight sets, *in* 'Proceedings of 3rd International Conference on the Theory and Application of Diagrams', Vol. 2980 of *LNAI*, Springer, Cambridge, UK, pp. 128–141.

Finding $\mathcal{EL}^+$ justifications using the Earley parsing algorithm

Riku Nortje[1,2] **Katarina Britz**[1,2] **Thomas Meyer**[1,2]

[1] Knowledge Systems Group, Meraka, CSIR,
PO Box 395, Pretoria 0001, South Africa

[2] School of Computing, University of South Africa,
PO Box 392, UNISA 0003, South Africa

Email: nortjeriku@gmail.com; {arina.britz;tommie.meyer}@meraka.org.za

Abstract

Module extraction plays an important role in the reuse of ontologies as well as in the simplification and optimization of some reasoning tasks such as finding justifications for entailments. In this paper we focus on the problem of extracting small modules for $\mathcal{EL}^+$ entailment based on reachability. We extend the current notion of (forward) reachability to obtain a bi-directional version, and show that the bi-directional reachability algorithm allows us to transform an $\mathcal{EL}^+$ ontology into a reachability preserving context free grammar (CFG). The well known Earley algorithm for parsing strings, given some CFG, is then applied to the problem of extracting minimal reachability-based axioms sets for subsumption entailments. We show that each reachability-based axiom set produced by the Earley algorithm corresponds to a possible Minimal Axiom Set (MinA) that preserves the given entailment. This approach has two advantages – it has the potential to reduce the number of subsumption tests performed during MinA extraction, as well to minimize the number of axioms for each such test.

1 Introduction

Reasoning tasks such as finding all justifications for an entailment are inherently hard, having at least an exponential worst case complexity. Even for description logics (DLs) such as $\mathcal{EL}^+$ (Suntisrivaraporn 2009) for which many reasoning tasks can be performed in polynomial time, the exponential nature of finding all justifications for an entailment is inescapable. Module extraction is one of the methods that aims to optimize the performance of this process by reducing the size of the ontology to a smaller subset of axioms that contains only the relevant axioms required for the entailment, thereby reducing the search space.

Extracting a minimal module is closely related to computing a deductive conservative extension of an ontology, which has been shown by Grau et al. (2007, 2008) to be intractable. They introduce syntactic locality-based modules, an approximation of minimal modules, that are more tractable and can be computed in polynomial time, whilst preserving all entailments.

Suntisrivaraporn (2009) introduced the notion of reachability-based modules for the DL $\mathcal{EL}^+$. Though the reachability-based module extraction algorithm differs from the syntactic locality-based algorithm, he proves that the modules extracted correspond to minimal syntactic locality-based modules.

The main criticism against reachability-based modules, as raised by Jianfeng Du & Ji (2009), is that, given an entailment $\mathcal{O} \models A \sqsubseteq B$, these methods extract a module for A and all concepts reachable from it without considering the super-concept B, resulting in large modules that in some cases do not reduce the size of the ontology at all. They propose a goal-directed algorithm for extracting just-preserving modules. The algorithm proceeds in two phases; the first, the off-line phase, transforms the ontology into a propositional program which preserves all logical relationships. The second phase, the on-line phase, utilizes the idea of maximally connected components, as used in SAT problem optimization, to extract a justification-preserving module. Experimental results show that modules obtained this way are smaller by an order of magnitude than their locality-based module counterparts.

Once a module has been extracted, various methods are used to find all justifications. A common approach is to systematically remove axioms that do not play a role in the entailment. After every iteration of the axiom removal procedure, a subsumption test determines if the entailment still holds in the resulting axiom set. This process continues until no more axioms can be removed. The resulting axiom set then constitutes a MinA. Subsumption testing is a computationally expensive procedure, even for $\mathcal{EL}^+$. Minimizing the number of subsumption tests during MinA extraction is therefore a primary concern when developing MinA extraction algorithms.

Our approach extends the reachability heuristic as introduced by Suntisrivaraporn (2009) to include backward reachability, thereby obtaining a bi-directional version of reachability. The heuristic allows us to make use of the well known Earley algorithm (Earley 1970) for parsing context free grammars (Jurafsky & Martin 2009) to compute all reachability-based paths between the sub- and super-concepts for an entailment. Each such path constitutes a minimal set of axioms such that reachability is preserved. Every reachability preserving axiom set does not guarantee entailment in itself, and therefore does not necessarily constitute a MinA. We focus on extracting all such minimal reachability preserving axiom sets. A standard subsumption test can then be employed in order to determine if the set constitutes a valid MinA. In this way we hope to reduce the number of subsumption tests performed during MinA extraction, as well as to minimize the number of axioms for each such test.

The Earley algorithm has been studied extensively in the literature and highly optimized software and hardware implementations exists (Chiang & Fu 1984, Pavlatos et al. 2003). At present we restrict our focus to $\mathcal{EL}^+$ which, because of its particular structure, allows us to transform any axiom into reachability preserving CFG production rules. The original Earley algorithm can then be used to extract all parse trees.

The rest of the paper is structured as follows. Section 2 contains the relevant background information

Name	Syntax	Semantics
top	$\top$	Δ^I
bottom	$\bot$	$\emptyset$
conjunction	$C \sqcap D$	$C^I \cap D^I$
existential restriction	$\exists r.C$	$\{x \in \Delta^I \mid \exists y \in \Delta^I : (x,y) \in r^I \wedge r \in C^I\}$
general concept inclusion (GCI)	$C \sqsubseteq D$	$C^I \subseteq D^I$
role inclusion (RI)	$r_1 \circ \ldots \circ r_k \sqsubseteq r$	$r_1^I \circ \ldots \circ r_k^I \subseteq r^I$
transitivity	transitive(r)	$\forall d,e,f \in \Delta^I : (d,e),(e,f) \in r^I \rightarrow (d,f) \in r^I$
reflexivity	reflexive(r)	$\forall d \in \Delta^I : (d,d) \subseteq r^I$
range restriction	range(r)$\sqsubseteq C$	$\{e \in \Delta^I \mid \exists d : (d,e) \in r^I\} \subseteq C^I$
domain restriction	domain(r)$\sqsubseteq C$	$\{d \in \Delta^I \mid \exists e : (d,e) \in r^I\} \subseteq C^I$
role hierarchy (RH)	$r \sqsubseteq s$	$r^I \subseteq s^I$

Table 1: $\mathcal{EL}^+$ syntax and semantics

on description logics, context free grammars, the Earley algorithm, and existing versions of reachability. In Section 3 we introduce a notion of backward (top-down) reachability. We show that a bi-directional reachability-based approach may be used to extract small modules that considers the sub-concept as well as the super-concept in an entailment. We show that modules obtained in this way may be smaller than reachability-based modules that consider only the sub-concept. In Section 4 we provide an algorithm for transforming any $\mathcal{EL}^+$ ontology into a reachability preserving CFG and show how the Earley algorithm can be used to extract a small strong subsumption module for a given entailment. Furthermore, in section 5 we show that the Earley algorithm simultaneously computes all parse trees, where each parse tree corresponds to a possible MinA. Section 6 is a discussion on work in progress, where we discuss possible changes to the standard Earley algorithm specifically aimed at optimizing the MinA discovery process. Section 7 briefly concludes and discusses future work.

2 Preliminaries

2.1 DL terminology

In the standard set-theoretic semantics of concept descriptions, concepts are interpreted as subsets of a domain of interest, and roles as binary relations over this domain. An interpretation I consists of a non-empty set Δ^I (the *domain* of I) and a function $\cdot^I$ (the *interpretation function* of I) which maps each atomic concept A to a subset A^I of Δ^I, and each atomic role R to a subset R^I of $\Delta^I \times \Delta^I$. The interpretation function is extended to arbitrary concept and role descriptions, with the specifics depending on the particular description logic under consideration. We provide the details for $\mathcal{EL}^+$ in Definition 1 below.

A DL knowledge base consists of a *TBox* which contains *terminological axioms*, and an *ABox* which contains *assertions*, i.e. facts about specific named objects and relationships between objects in the domain. For the purposes of this paper we concern ourselves only with Tbox statements.

TBox statements are *general concept inclusions* of the form $C \sqsubseteq D$, where C and D are (possibly complex) concept descriptions. $C \sqsubseteq D$ is also called a *subsumption statement*, read "C is subsumed by D". An interpretation I *satisfies* $C \sqsubseteq D$, written $I \Vdash C \sqsubseteq D$, iff $C^I \subseteq D^I$. $C \sqsubseteq D$ is *valid*, written $\models C \sqsubseteq D$, iff it is satisfied by all interpretations.

An interpretation I satisfies a DL knowledge base $\mathcal{K}$ iff it satisfies every statement in $\mathcal{K}$. A DL knowledge base $\mathcal{K}$ *entails* a DL statement ϕ, written as $\mathcal{K} \models \phi$, iff every interpretation that satisfies $\mathcal{K}$ also satisfies ϕ.

Roughly speaking, DLs are defined by the constructors they provide. There exists a correlation between the expressivity of the DL and the complexity of reasoning over it. We consider the DL $\mathcal{EL}^+$ which is defined as follows:

Definition 1 ($\mathcal{EL}^+$ syntax and semantics) *The syntax and semantics of $\mathcal{EL}^+$ constructors are defined in Table 1.*

We further require that an $\mathcal{EL}^+$ ontology conform to the following ***syntactic restriction*** (Suntisrivaraporn 2009): For an ontology $\mathcal{O}$ and role names r,s, we write $\mathcal{O} \models r \sqsubseteq s$ if and only if $r = s$ or $\mathcal{O}$ contains role inclusions $r_1 \sqsubseteq r_2, \ldots, r_{k-1} \sqsubseteq r_k$ with $r = r_1$ and $s = r_k$. Also, we write $\mathcal{O} \models \mathit{range}(r) \sqsubseteq C$ if there is a role name s such that $\mathcal{O} \models r \sqsubseteq s$ and $\mathit{range}(s) \sqsubseteq C \in \mathcal{O}$. The $\mathcal{EL}^+$ syntactic restriction is as follows: If $r_1 \circ \ldots \circ r_k \sqsubseteq s \in \mathcal{O}$ with $k \geq 1$ and $\mathcal{O} \models \mathit{range}(s) \sqsubseteq C$, then $\mathcal{O} \models \mathit{range}(r_k) \sqsubseteq C$.

Intuitively, the restriction ensures that a role inclusion $r_1 \circ \cdots \circ r_k \sqsubseteq s$ does not induce any new range constraints on the role composition $r_1 \circ \cdots \circ r_k$. Formally, it ensures that if the role inclusion implies a role relationship $(d,\ e) \in\ s^I$ in the model, then the range restrictions on s do not impose new concept memberships on e. Without this restriction reasoning in $\mathcal{EL}^+$ becomes intractable (Suntisrivaraporn 2009, Baader et al. 2008).

Given an ontology $\mathcal{O}$ and an entailment $\mathcal{O} \models \sigma$ with σ a statement of interest, a *justification* for σ is a set of axioms from $\mathcal{O}$ such that the entailment is preserved. A *minimal axiom set (MinA)* is the smallest set of axioms that preserves the entailment.

Definition 2 (Minimal Axiom set) *Let $\mathcal{O}$ be an ontology, and σ a subsumption statement such that $\mathcal{O} \models \sigma$. A subset $S \subseteq \mathcal{O}$ is a minimal axiom set (MinA) for σ w.r.t. $\mathcal{O}$, also written as "S is a MinA for $\mathcal{O} \models \sigma$", if and only if*

1. *$S \models \sigma$, and*
2. *for every $S' \subset S$, $S' \not\models \sigma$.*

Definition 3 (Signature of $\mathcal{O}$) *Let* CN*($\mathcal{O}$) represent the set of all concept names in $\mathcal{O}$,* RN*($\mathcal{O}$) the set of all role names in $\mathcal{O}$. We define the signature of $\mathcal{O}$, denoted as Sig($\mathcal{O}$), as the union of all concept and role names occurring in $\mathcal{O}$ i.e., Sig($\mathcal{O}$) =* CN*($\mathcal{O}$) ∪* RN*($\mathcal{O}$). Similarly for* any *$\mathcal{EL}^+$ statement σ, Sig(σ) is the union of all concept and role names occurring in σ.*

2.2 The Earley algorithm for parsing CFG languages

Context free grammars (CFGs) provide a well-known method for modeling the structure of English and other natural languages. A grammar consists of a set of productions or rules, each of which expresses the ways the symbols (strings) in a language can be grouped together, as well as a lexicon of words or symbols.

Definition 4 (CFG production rules) *Given that X represents a single non-terminal, the symbol "a" represents a single terminal and α and σ represent mixed strings of terminals and non-terminals, including the* null *string. CFG production rules have the form:*

$$X \rightarrow \alpha\sigma \quad (1)$$
$$X \rightarrow a \quad (2)$$

Parsing a CFG string results in a parse tree, assigning syntactic structure to it. The Earley parsing algorithm (Earley 1970) uses a dynamic programming approach applying a single left-to-right, top-down, depth-first parallel search strategy to compute a chart that contains all possible parses for a given input. It accomplishes this in polynomial worst case time (n^3), where n is the size of the input string.

Example 1 *Consider the sample CFG for a subset of English grammar below. The set of symbols $\{that, book, flight\}$ represent terminal symbols and all other symbols represent non-terminals.*

$$\begin{aligned} S &\rightarrow VP \\ S &\rightarrow NP \quad VP \\ VP &\rightarrow VP \quad NP \\ NP &\rightarrow Det \quad Noun \\ VP &\rightarrow Verb \\ Det &\rightarrow that \\ Verb &\rightarrow book \\ Noun &\rightarrow flight \end{aligned}$$

During execution the Earley algorithm generates a state entry for each production rule it operates on. The purpose of the state is to record the progress made during the parsing process.

Definition 5 (Parse states) *Let X and Y represent single non-terminal symbols, let a represent a single terminal symbol, and let α, β and σ represent mixed strings of terminal and non-terminal symbols, including the* null *string. Then for each token (word) in the input string, the Earley algorithm creates a set of states, called a chart. A chart at position k of the input is represented by $\mathcal{C}_k$. Each state consists of a tuple ($X \rightarrow \alpha \bullet \beta$, i) where*

1. *$X \rightarrow \alpha\beta$ is the current production rule,*
2. *$\bullet$ indicates the* dot rule *which represents the current parsing position in the state, with $\alpha \bullet \beta$ indicating that α has previously been parsed and β is expected next, and*
3. *i indicates the starting index of the substring where parsing of this production began.*

The parser consists of three sub-parts, the *predictor*, *scanner* and *completer*. For each state in chart $\mathcal{C}_i$, the tuple ($X \rightarrow \alpha \bullet \beta$, j) is evaluated and the appropriate sub-part executed:

1. **Predictor:** If state = ($X \rightarrow \alpha \bullet Y\beta$, j) , then for every production $Y \rightarrow \sigma$, if $(Y \rightarrow \bullet\sigma) \notin \mathcal{C}_i$ then $\mathcal{C}_i := \mathcal{C}_i + (Y \rightarrow \bullet\sigma, i)$,
2. **Scanner:** If state = ($X \rightarrow \alpha \bullet a\beta$, j), with a the next symbol in the input stream, and if $(X \rightarrow \alpha a \bullet \beta, j) \notin \mathcal{C}_{i+1}$ then $\mathcal{C}_{i+1} := \mathcal{C}_{i+1} + (X \rightarrow \alpha a \bullet \beta, j)$,
3. **Completer:** If state = ($X \rightarrow \gamma\bullet$, j), then for every $(Y \rightarrow \alpha \bullet X\beta, k) \in \mathcal{C}_j$, $\mathcal{C}_i := \mathcal{C}_i + (Y \rightarrow \alpha X \bullet \beta, k)$.

The algorithm executes all states iteratively in a top-down manner until no new states are available for processing, and no state may appear more than once in a given chart (Jurafsky & Martin 2009).

Example 2 *Table 2 shows the output of the Earley algorithm given the input string* 'book that flight' *and the CFG in Example 1. The state*

$$20. \quad S \rightarrow VP\bullet$$

in chart 3 represents a successful parse of the string.

Table 2: Parse for 'book that flight'

Chart 0:	• book that flight
1. S → • NP VP	$j = 0$: Inital State
2. S → • VP	$j = 0$: Inital State
3. NP → • Det Noun	$j = 0$: Predictor 1
4. VP → • VP NP	$j = 0$: Predictor 2
5. VP → • Verb	$j = 0$: Predictor 2
6. Det → • that	$j = 0$: Predictor 3
7. Verb → • book	$j = 0$: Predictor 5
Chart 1:	book • that flight
8. Verb → book •	$j = 0$: Scanner 7
9. VP → Verb •	$j = 0$: Completer 5, 8
10. VP → VP • NP	$j = 0$: Completer 4, 8
11. NP → • Det Noun	$j = 1$: Predictor 10
12. Det → • that	$j = 1$: Predictor 11
Chart 2:	book that • flight
13. Det → that •	$j = 1$: Scanner 12
14. NP → Det • Noun	$j = 1$: Completer 11, 13
15. Noun → • flight	$j = 2$: Predictor 14
Chart 3:	book that flight •
16. Noun → flight •	$j = 2$: Scanner 15
17. NP → Det Noun •	$j = 1$: Completer 14, 16
18. VP → VP NP •	$j = 0$: Completer 10, 17
19. VP → VP • NP	$j = 0$: Completer 4, 18
20. S → VP •	$j = 0$: Completer 2, 18

2.3 Reachability-based module extraction

Given an ontology $\mathcal{O}$ and an entailment $\mathcal{O} \models \sigma$ with σ a statement of interest, extracting a module aims to obtain a small subset $\mathcal{O}'$ of $\mathcal{O}$, such that entailment of σ is preserved, where Sig(σ) is defined as in Definition 3. For the purposes of this paper σ is always a subsumption statement.

Definition 6 (Module for $\mathcal{EL}^+$) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology, and σ a statement formulated in $\mathcal{EL}^+$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a* module for σ in $\mathcal{O}$*(a σ-module in $\mathcal{O}$) whenever: $\mathcal{O} \models \sigma$ if and only if $\mathcal{O}' \models \sigma$. We say that $\mathcal{O}'$ is a module for a signature $\boldsymbol{S}$ in $\mathcal{O}$ (an* **S**-module in $\mathcal{O}$*) if, for every $\mathcal{EL}^+$ statement σ with Sig(σ) $\subseteq \boldsymbol{S}$, $\mathcal{O}'$ is a σ-module in $\mathcal{O}$.*

Definition 7 (Reachability-based modules) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $S \subseteq$ Sig($\mathcal{O}$) a signature. The set of S-reachable names in $\mathcal{O}$ is defined inductively as:*

- *x is S-reachable in $\mathcal{O}$, for every $x \in S$;*
- *for all inclusion axioms $\alpha_L \sqsubseteq \alpha_R$, if x is S-reachable in $\mathcal{O}$ for every $x \in Sig(\alpha_L)$, then y is S-reachable in $\mathcal{O}$ for every $y \in Sig(\alpha_R)$.*

We call an axiom $\alpha_L \sqsubseteq \alpha_R$ S-reachable in $\mathcal{O}$ if every element of $Sig(\alpha_L)$ is S-reachable in $\mathcal{O}$. The reachability-based module for S in $\mathcal{O}$, denoted by $\mathcal{O}_S^{reach}$, consists of all S-reachable axioms from $\mathcal{O}$.

When S is the single concept A, we write A-reachable and $\mathcal{O}_A^{reach}$. An interesting result of reachability is that it can be used to test negative subsumption. That is, if B is not A-reachable in $\mathcal{O}$, then $\mathcal{O} \not\models A \sqsubseteq B$, unless A is unsatisfiable w.r.t $\mathcal{O}$ (Suntisrivaraporn 2009).

Definition 8 (Subsumption module) *Let $\mathcal{O}$ be an ontology, and A a concept name occurring in $\mathcal{O}$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a subsumption module for A in $\mathcal{O}$ whenever: $\mathcal{O} \models A \sqsubseteq B$ if and only if $\mathcal{O}' \models A \sqsubseteq B$ holds for every concept name B occurring in $\mathcal{O}$.*

A subsumption module $\mathcal{O}'$ for A in $\mathcal{O}$ is called strong *if the following holds for every concept name B occurring in $\mathcal{O}$: if $\mathcal{O} \models A \sqsubseteq B$, then every MinA for $\mathcal{O} \models A \sqsubseteq B$ is a subset of $\mathcal{O}'$.*

Theorem 1 *(Suntisrivaraporn 2009). The module $\mathcal{O}_A^{reach}$ is a strong subsumption module for A in $\mathcal{O}$.*

We require that an $\mathcal{EL}^+$ ontology $\mathcal{O}$ be in normal form. We use the same form as Brandt (2004) and Suntisrivaraporn (2009). Any $\mathcal{EL}^+$ ontology $\mathcal{O}$ can be converted to an ontology $\mathcal{O}'$ in normal form in linear time, with at most a linear increase in the size of the ontology.

Let $CN(\mathcal{O})$ represent the set of all concept names in $\mathcal{O}$, $RN(\mathcal{O})$ the set of all role names in $\mathcal{O}$, $CN(\mathcal{O})^\top = CN(\mathcal{O}) \cup \{\top\}$ and $CN(\mathcal{O})^\bot = CN(\mathcal{O}) \cup \{\bot\}$.

Definition 9 (Normal Form) *An $\mathcal{EL}^+$ ontology $\mathcal{O}$ is in* normal form *if the following conditions are satisfied:*

1. *all concept inclusions in $\mathcal{O}$ have one of the following forms:*
$$\begin{aligned} A_1 \sqcap \ldots \sqcap A_n &\sqsubseteq B, \\ A_1 &\sqsubseteq \exists r.A_2, \\ \exists r.A_1 &\sqsubseteq B \end{aligned}$$
where $A_i \in CN^\top(\mathcal{O})$ and $B \in CN^\bot(\mathcal{O})$;
2. *all role inclusions in $\mathcal{O}$ have one of the following forms:*
$$\begin{aligned} \epsilon &\sqsubseteq r, \\ r &\sqsubseteq s, \\ r \circ s &\sqsubseteq t, \end{aligned}$$
where $r, s, t \in RN(\mathcal{O})$ and ϵ is the identity element;
3. *there are no reflexivity statements, transitivity statements or domain restrictions, and all range restrictions are of the form $range(r) \sqsubseteq A$ with A a concept name.*

3 Bi-directional reachability-based module

Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$ and entailment $\mathcal{O} \models A \sqsubseteq B$, as well as the module $\mathcal{O}_A^{reach}$, we have that $\mathcal{O} \models A \sqsubseteq B$ if and only if $\mathcal{O}_A^{reach} \models A \sqsubseteq B$, where A and B are concepts names. $\mathcal{O}_A^{reach}$ preserves entailments for all concept names α such that $\mathcal{O} \models A \sqsubseteq \alpha$.

A criticism raised against reachability-based modules, in terms of finding justifications, is that they contain many irrelevant axioms, and in some cases do not reduce the size of the ontology at all (Jianfeng Du & Ji 2009). This stems from the fact that $\mathcal{O}_A^{reach}$ considers only the sub-concept A in $\mathcal{O} \models A \sqsubseteq B$; the super-concept B is never used to elimate unwanted axioms.

Example 3 *Given the small ontology $\mathcal{O}$ below, as well as $\mathcal{O} \models A \sqsubseteq B$, $\mathcal{O}_A^{reach}$ will consist of axioms 1, 2 and 4. Axiom 4 is irrelevant in terms of finding justifications for $\mathcal{O} \models A \sqsubseteq B$, yet it is included in $\mathcal{O}_A^{reach}$.*

$$\begin{aligned} A &\sqsubseteq \exists r.D && (1) \\ \exists r.D &\sqsubseteq B && (2) \\ E &\sqsubseteq B && (3) \\ A &\sqsubseteq F && (4) \end{aligned}$$

Given the entailment $\mathcal{O} \models A \sqsubseteq B$, reachability can be applied in two directions: The standard bottom-up approach, which extracts $\mathcal{O}_A^{reach}$, and a top-down approach, which extracts $\mathcal{O}_{\overleftarrow{\mathrm{B}}}^{reach}$, and is defined as follows:

Definition 10 (Top-down reachability-based module) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. The set of $\overleftarrow{\mathrm{S}}$-reachable names in $\mathcal{O}$ is defined inductively as:*

- *x is $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$, for every $x \in S$;*
- *for all inclusion axioms $\alpha_L \sqsubseteq \alpha_R$, if x is $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$ for some $x \in Sig(\alpha_R)$, or if $\alpha_R = \bot$, then y is $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$ for every $y \in Sig(\alpha_L)$.*

We call an axiom $\alpha_L \sqsubseteq \alpha_R$ $\overleftarrow{\mathrm{S}}$-reachable in $\mathcal{O}$ if some element of $Sig(\alpha_R)$ is $\overleftarrow{\mathrm{S}}$-reachable or if $\alpha_R = \bot$. The top-down reachability-based module for S in $\mathcal{O}$, denoted by $\mathcal{O}_{\overleftarrow{\mathrm{S}}}^{reach}$, consists of all $\overleftarrow{\mathrm{S}}$-reachable axioms from $\mathcal{O}$.

Besides the direction of application, there is a fundamental difference between the two approaches. When extracting $\mathcal{O}_A^{reach}$, the axiom $\alpha_L \sqsubseteq \alpha_R$ becomes A-reachable only when all $x_i \in \mathrm{Sig}(\alpha_L)$ are A-reachable. When extracting $\mathcal{O}_{\overleftarrow{\mathrm{B}}}^{reach}$, the axiom $\alpha_L \sqsubseteq \alpha_R$ is $\overleftarrow{\mathrm{B}}$-reachable whenever any $x_i \in \mathrm{Sig}(\alpha_R)$ is $\overleftarrow{\mathrm{B}}$-reachable.

By definition of reachability, axioms of the form $\top \sqsubseteq \alpha_R$ and $\epsilon \sqsubseteq r$ are A-reachable, since $\mathrm{Sig}(\top) = \mathrm{Sig}(\epsilon) = \emptyset$, and form part of any module $\mathcal{O}_A^{reach}$ extracted (Suntisrivaraporn 2009).

Further by definition of top-down reachability, axioms of the form $\alpha_L \sqsubseteq \bot$ are $\overleftarrow{\mathrm{B}}$-reachable. Therefore all axioms $\alpha_L \sqsubseteq \bot$ will also always be a part of any module $\mathcal{O}_{\overleftarrow{\mathrm{B}}}^{reach}$ being extracted.

From Theorem 1 we have that $\mathcal{O}_A^{reach}$ preserves all entailments in terms of the sub-concept A. We show in Theorem 2 that a similar result holds for $\mathcal{O}_{\overleftarrow{\mathrm{B}}}^{reach}$ with respect to the super-concept B.

Definition 11 (Top-down subsumption module) *Let $\mathcal{O}$ be an ontology, and B a concept name occurring in $\mathcal{O}$. Then, $\mathcal{O}' \subseteq \mathcal{O}$ is a top-down subsumption module for B in $\mathcal{O}$ whenever: $\mathcal{O} \models A \sqsubseteq B$ if and only if $\mathcal{O}' \models A \sqsubseteq B$ holds for every concept name A occurring in $\mathcal{O}$.*

A top-down subsumption module $\mathcal{O}'$ for B in $\mathcal{O}$ is called strong *if the following holds for every concept name A occurring in $\mathcal{O}$: if $\mathcal{O} \models A \sqsubseteq B$, then every MinA for $\mathcal{O} \models A \sqsubseteq B$ is a subset of $\mathcal{O}'$.*

We show that top-down reachability modules preserves all subsumption relationships i.t.o super-concepts.

Lemma 1 *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and $S \subseteq Sig(\mathcal{O})$ a signature. Then, $\mathcal{O} \models C \sqsubseteq D$ if and only if $\mathcal{O}^{reach}_{\overleftarrow{S}} \models C \sqsubseteq D$ for arbitrary $\mathcal{EL}^+$ concept descriptions C and D such that $Sig(D) \subseteq S$.*

Proof: We have to prove two parts. First: If $\mathcal{O}^{reach}_{\overleftarrow{S}} \models C \sqsubseteq D$ then $\mathcal{O} \models C \sqsubseteq D$. This follows directly from the fact that $\mathcal{O}^{reach}_{\overleftarrow{S}} \subseteq \mathcal{O}$ and that $\mathcal{EL}^+$ is monotonic.

Second, we show that, if $\mathcal{O} \models C \sqsubseteq D$ then $\mathcal{O}^{reach}_{\overleftarrow{S}} \models C \sqsubseteq D$: Assume the contrary, that is, assume $\mathcal{O} \models C \sqsubseteq D$ but that $\mathcal{O}^{reach}_{\overleftarrow{S}} \not\models C \sqsubseteq D$. Then there must exist an interpretation I and an individual $w \in \Delta^I$ such that I is a model of $\mathcal{O}^{reach}_{\overleftarrow{S}}$ and $w \in C^I \setminus D^I$. Modify I to I' by setting $x^{I'} := \Delta^I$ for all concept names $x \in \text{Sig}(\mathcal{O}) \setminus (S \cup \text{Sig}(\mathcal{O}^{reach}_{\overleftarrow{S}}))$, and $r^{I'} := \Delta^I \times \Delta^I$ for all roles names $r \in \text{Sig}(\mathcal{O}) \setminus (S \cup \text{Sig}(\mathcal{O}^{reach}_{\overleftarrow{S}}))$. I' is a model of $\mathcal{O}^{reach}_{\overleftarrow{S}}$ since it does not change the interpretation of any symbol in its signature. For each $\alpha = (\alpha_L \sqsubseteq \alpha_R) \in \mathcal{O} \setminus \mathcal{O}^{reach}_{\overleftarrow{S}}$, we have $\alpha_L^{I'} \subseteq \alpha_R^{I'}$ since α is not $\overleftarrow{S}$-reachable and thus $\alpha_R^{I'} = \Delta^I$. Therefore I' is a model for $\mathcal{O}$. But I and I' correspond on all symbols $y \in \text{Sig}(D) \subseteq S$ and $C^I \subseteq C^{I'}$, therefore we have that $w \in C^{I'} \setminus D^{I'}$, contradicting the assumption.

In order to show that $\mathcal{O}^{reach}_{\overleftarrow{B}}$ contains all MinAs for the entailment $\mathcal{O} \models A \sqsubseteq B$, we show that $\mathcal{O}^{reach}_{\overleftarrow{B}}$ is a strong top-down subsumption module:

Theorem 2 *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology and B a concept name occurring in $\mathcal{O}$. Then $\mathcal{O}^{reach}_{\overleftarrow{B}}$ is a strong top-down subsumption module for B in $\mathcal{O}$.*

Proof: That $\mathcal{O}^{reach}_{\overleftarrow{B}}$ is a top-down subsumption module follows directly from Lemma 1 above. To show that it is strong, assume that $\mathcal{O} \models A \sqsubseteq B$, but there is a MinA S for $\mathcal{O} \models A \sqsubseteq B$ that is not contained in $\mathcal{O}^{reach}_{\overleftarrow{B}}$. Thus, there must be an axiom $\alpha \in S \setminus \mathcal{O}^{reach}_{\overleftarrow{B}}$. Define $S_1 := S \cap \mathcal{O}^{reach}_{\overleftarrow{B}}$. S_1 is a strict subset of S since $\alpha \notin S_1$. We claim that $S_1 \models A \sqsubseteq B$, which contradicts the fact that S is a MinA for $\mathcal{O} \models A \sqsubseteq B$.

We use proof by contradiction to show this. Assume that $S_1 \not\models A \sqsubseteq B$ i.e., there is a model I_1 of S_1 such that $A^{I_1} \not\subseteq B^{I_1}$. We modify I_1 to I by setting $y^I := \Delta^{I_1}$ for all concept names y that are not $\overleftarrow{B}$-reachable, and $r^I := \Delta^{I_1} \times \Delta^{I_1}$ for all roles names r that are not $\overleftarrow{B}$-reachable. We have $B^I = B^{I_1}$ since B is $\overleftarrow{B}$-reachable, and $A^I = A^{I_1}$ if A is $\overleftarrow{B}$-reachable, or $A^I = \Delta^{I_1}$ otherwise. Therefore $A^I \not\subseteq B^I$. It remains to be shown that I is indeed a model of S, and therefore satisfies all axioms $\beta = (\beta_L \sqsubseteq \beta_R)$ in S, including $A \sqsubseteq B$. There are two possibilities:

- $\beta \in S_1$. Since $S_1 \subseteq \mathcal{O}^{reach}_{\overleftarrow{B}}$, all symbols in $\text{Sig}(\beta_L)$ and one or more symbols in $\text{Sig}(\beta_R)$ are $\overleftarrow{B}$-reachable. Consequently, I_1 and I coincide on the names occurring in β_L and since I_1 is a model of S_1, we have that $(\beta_L)^I = (\beta_L)^{I_1}$ and $(\beta_R)^{I_1} \subseteq (\beta_R)^I$. Therefore $(\beta_L)^I \subseteq (\beta_R)^I$.
- $\beta \notin S_1$. Since $S_1 = S \setminus \mathcal{O}^{reach}_{\overleftarrow{B}}$, we have that β is not $\overleftarrow{B}$-reachable. Thus no $x \in \text{Sig}(\beta_R)$ is $\overleftarrow{B}$-reachable. By the definition of I, $(\beta_R)^I = \Delta^{I_1}$. Hence $(\beta_L)^I \subseteq (\beta_R)^I$.

Therefore I is a model for S.

Example 4 *Extracting $\mathcal{O}^{reach}_{\overleftarrow{B}}$ from the sample ontology in Example 3, we see that it will consist of axioms 1, 2 and 3. This correctly differs from $\mathcal{O}^{reach}_{A}$ in that axiom 4 is not $\overleftarrow{B}$-reachable. Similar to $\mathcal{O}^{reach}_{A}$ though, it contains an axiom that is irrelevant in terms of finding justifications for $\mathcal{O} \models A \sqsubseteq B$, axiom 3 in this case.*

It is clear that when extracting modules for finding justifications, $\mathcal{O}^{reach}_{\overleftarrow{B}}$ opens itself to the same criticism as $\mathcal{O}^{reach}_{A}$. Given the entailment $\mathcal{O} \models A \sqsubseteq B$ we see that where $\mathcal{O}^{reach}_{A}$ considers only the sub-concept A, $\mathcal{O}^{reach}_{\overleftarrow{B}}$ considers only the super-concept B.

Both reachability module extraction methods preserve all entailments; $\mathcal{O}^{reach}_{A}$ entailments in terms of the sub-concept A and $\mathcal{O}^{reach}_{\overleftarrow{B}}$ entailments in terms of the super-concept B. Given the entailment $\mathcal{O} \models A \sqsubseteq B$, we may now extract the module $(\mathcal{O}^{reach}_{A})^{reach}_{\overleftarrow{B}}$, or similarly $(\mathcal{O}^{reach}_{\overleftarrow{B}})^{reach}_{A}$, such that it considers both the sub- and super concepts in the entailment. The resulting module is a *bi-directional reachability-based module* denoted by $\mathcal{O}^{reach}_{A \leftrightarrow B}$.

Definition 12 (Bi-directional reachability-based module) *A bi-directional reachability-based module, denoted $\mathcal{O}^{reach}_{A \leftrightarrow B}$, is defined as the set of all axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$ such that for every $x_i \in Sig(\alpha_L)$, x_i is A-reachable, and for some $y_i \in Sig(\alpha_R)$, y_i is $\overleftarrow{B}$-reachable.*

Example 5 *Extracting $\mathcal{O}^{reach}_{A \leftrightarrow B}$ from the sample ontology in Example 3, we see that it will consist of axioms 1 and 2. The previous irrelevant axioms 3 and 4 are no longer present.*

From Theorem 1 and Theorem 2 we have that bi-derectional reachabiliy modules preserves all MinAs for the entailment $\mathcal{O} \models A \sqsubseteq B$.

Corollary 1 *$\mathcal{O}^{reach}_{A \leftrightarrow B}$ preserves all MinAs for $\mathcal{O} \models A \sqsubseteq B$.*

4 Reachability preserving CFG

We show how an $\mathcal{EL}^+$ ontology $\mathcal{O}$ can be transformed into a reachability preserving CFG. In the discussion that follows we assume that we have an $\mathcal{EL}^+$ ontology $\mathcal{O}$ in normal form, and an entailment $\mathcal{O} \models A \sqsubseteq B$, where A and B are single concept names.

From the definition of $\mathcal{O}^{reach}_{A \leftrightarrow B}$ above , we have that every axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}^{reach}_{A \leftrightarrow B}$ has the following two properties:

1. every $x_i \in \text{Sig}(\alpha_L)$ is A-reachable, and
2. some $y_i \in \text{Sig}(\alpha_R)$ is $\overleftarrow{B}$-reachable.

Every CFG production rule we introduce must preserve bi-directional reachability. By Property 1 above, A-reachability of the axiom $\alpha_L \sqsubseteq \alpha_R$ is solely dependent on symbols in α_L. Similarly, by Property 2, $\overleftarrow{B}$-reachability is solely dependent on symbols in α_R.

From the previous section we know that there exists special cases in which A- and $\overleftarrow{B}$-reachability hold. For A-reachability these axioms have one of the forms:

- $\top \sqsubseteq \alpha_R$, or
- $\epsilon \sqsubseteq \alpha_R$

For $\overleftarrow{\text{B}}$-reachability these axioms have the form:

- $\alpha_L \sqsubseteq \bot$

In the steps that follow, all production rules we introduce have the form $y_i \rightarrow \sigma$. Each rule is read as: any $\overleftarrow{\text{B}}$-reachable symbol y_i is A-reachable only if all symbols $x_i \in \sigma$ are A-reachable. This clearly conforms to the definition of bi-directional reachability. We further note that the symbols on the rhs of CFG production rules have a fixed order, whereas the conjunction of $\mathcal{EL}^+$ concepts and roles are symmetric, i.e. $A \sqcap B = B \sqcap A$, and thus order is unimportant. We therefore place no restrictions on the order of the symbols on the rhs of production rules and thus consider production rules differing only in the order of symbols on the rhs as identical.

The conversion process below proceeds in a step by step manner until all axioms in $\mathcal{O}$ have been processed.

Step 1: All axioms $\alpha_L \sqsubseteq \alpha_R$ in $\mathcal{O}$ such that $\text{Sig}(\alpha_R) = \emptyset$ are $\overleftarrow{\text{B}}$-reachable by definition. By Property 2 above, in order to preserve both $\overleftarrow{\text{B}}$-reachability as well as bi-directional reachability, $\overleftarrow{\text{B}}$-reachability depends solely on α_R. For each such axiom the implicit $\overleftarrow{\text{B}}$-reachability of α_R is made explicit by introducing the following production rule:

$$B \rightarrow Sig(\alpha_L)$$

Step 2: All axioms $\alpha_L \sqsubseteq \alpha_R$ in $\mathcal{O}$ such that $\text{Sig}(\alpha_L) = \emptyset$ are (implicitly) A-reachable. By property 1 above, in order to preserve both A-reachability as well as bi-directional reachability, A-reachability depends solely on α_L. For each such axiom the implicit A-reachability of α_L is made explicit by introducing the production rule:

$$y_i \rightarrow A$$

for each $y_i \in Sig(\alpha_R)$.

Step 3: For each axiom $\alpha_L \sqsubseteq \alpha_R$ in $\mathcal{O}$ such that $|\text{Sig}(\alpha_L)| \geq 1$ and $|\text{Sig}(\alpha_R)| \geq 1$, introduce the production rule:

$$y_i \rightarrow Sig(\alpha_L)$$

for each $y_i \in Sig(\alpha_R)$. Axioms of this kind do not have any implicit reachability concerns like those in Steps 1 and 2 above, and bi-directional reachability is preserved trivially.

We note that it follows from the normal form in Definition 9 that, for every rule introduced in Steps 2 and 3 above where $|\text{Sig}(\alpha_R)| = 2$, α_R has the form $\exists rC$. By property 2 above, $\overleftarrow{\text{B}}$-reachability is preserved if either one of r or C is $\overleftarrow{\text{B}}$-reachable. Bi-directional reachability is therefore preserved by the two rules:

$$r \rightarrow Sig(\alpha_L)$$

$$C \rightarrow Sig(\alpha_L)$$

in Step 3 above, and similarly for Step 2. We therefore define the reachability preserving CFG for an $\mathcal{EL}^+$ ontology $\mathcal{O}$ as:

Definition 13 (Reachability preserving CFG) *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology in normal form, and $\mathcal{O} \models A \sqsubseteq B$ an entailment, then the reachability preserving CFG, denoted $\mathcal{CFG}_\mathcal{O}$, is the minimal set of CFG production rules such that for each axiom $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}$:*

- *if $Sig(\alpha_R) = \emptyset$, the rule $B \rightarrow Sig(\alpha_L) \in \mathcal{CFG}_\mathcal{O}$;*
- *if $Sig(\alpha_L) = \emptyset$ the rule $x_i \rightarrow A \in \mathcal{CFG}_\mathcal{O}$ for each $x_i \in Sig(\alpha_R)$;*
- *for all other axioms the rule $x_i \rightarrow Sig(\alpha_L) \in \mathcal{CFG}_\mathcal{O}$ for each $x_i \in Sig(\alpha_R)$;*

where the symbol A represents the only terminal symbol and the set $Sig(\mathcal{O})\backslash A$ represent the set of non-terminals.

Example 6 *All production rules for $\mathcal{CFG}_\mathcal{O}$ may be obtained from the ontology in Example 3 above as follows:*

r	$\rightarrow$	A	*(Step 3 applied to axiom 1)*
D	$\rightarrow$	A	*(Step 3 applied to axiom 1)*
B	$\rightarrow$	$r\ D$	*(Step 3 applied to axiom 2)*
B	$\rightarrow$	E	*(Step 3 applied to axiom 3)*
F	$\rightarrow$	A	*(Step 3 applied to axiom 4)*

Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$, with $\mathcal{O}'$ being $\mathcal{O}$ in normal form, and n the number of axioms in $\mathcal{O}'$, we see that there can be at most $2 \times n$ production rules in $CFG_{\mathcal{O}'}$. This follows directly from the definition of $CFG_{\mathcal{O}'}$, and the fact that there is a one-to-one correspondence between all CFG production rules introduced and axioms in $\mathcal{O}'$, except for axioms $\alpha_L \sqsubseteq \alpha_R \in \mathcal{O}'$ where $\alpha_R = \exists r.C$; for these axioms two production rules are introduced.

5 Earley as a MinA extraction algorithm

Given an $\mathcal{EL}^+$ ontology $\mathcal{O}$, an entailment $\mathcal{O} \models A \sqsubseteq B$ and the resulting CFG $\mathcal{CFG}_\mathcal{O}$, the Earley algorithm may be applied to extract all possible parse trees.

Before the algorithm is executed we introduce the start state $S \rightarrow B$, where S $\notin$ Sig($\mathcal{O}$). The algorithm now proceeds in a top-down manner, starting with this state, and then proceeds to find all production rules $\sigma_L \rightarrow \sigma_R$ such that σ_L is $\overleftarrow{\text{B}}$-reachable and all $x_i \in \sigma_R$ are A-reachable.

From the definition of $\mathcal{CFG}_\mathcal{O}$ we have that the symbol A is the only terminal symbol and all other symbols are considered to be non-terminals. The standard algorithm requires an input string to parse. While in terms of reachability there is no explicit input string, it is implicit from the definition of $\mathcal{CFG}_\mathcal{O}$ that the input string consists of a finite string of As.

The standard Earley algorithm may now be applied in order to extract all possible reachability paths between the concepts A and B. However, since the Earley algorithm is not explicitly bound by an input string, the situation arises that it may never terminate. The algorithm inherently handles cycles, but the absence of an explicit input-sentence may lead to non-termination. This occurs when a cycle is right-recursive, where the recursive part is not A-reachable. For example let the rules:

$$B \rightarrow C\ D$$

$$C \rightarrow A$$

$$D \rightarrow B$$

represent such a cycle. Then D can never be A-reachable, however since C is A-reachable the Earley algorithm will go into an infinite loop.

There are a few ways to remedy this. The first is to introduce a depth bound n, where n is the sum total of production rulesl symbols appearing on the rhs of production rules. The choice of n stems from the fact that the Earley algorithm exhaustively searches

for parse trees, each node represented by some production rule. In chart k, the Early algorithm searches trees with the length of the longest branch in the tree equal to $k+1$. This branch then represents the longest chain of bi-directional reachability preserving production rules, with each rule appearing at most j times, where j is the sum total of times the lhs symbols of the production rule appears on the rhs of other production rules. Thus the longest branch of any parse tree cannot exceed n. The standard Earley algorithm will in this case run in $\mathcal{O}(n^3)$ worst case time.

A different approach is to first extract $\mathcal{O}_A^{reach}$ and running the Earley algorithm on it. This guarantees that all production rules are A-reachable and the above cycle can never occur, this however does not gurantee that no other bad cycles exists. A third approach involves changing the algorithm itself similar to the method used in Section 6.

Both $\mathcal{O}_A^{reach}$ and $\mathcal{O}_{\overleftarrow{\mathrm{B}}}^{reach}$ can be extracted in linear time, and hence so can $\mathcal{O}_{A \leftrightarrow B}^{reach}$. The standard Earley algorithm is therefore non-optimal by two orders of magnitude in terms of module extraction. The benefit gained from the Earley algorithm however is that all parse trees are computed simultaneously.

Each parse tree computed by the Earley algorithm corresponds to a set of production rules, starting with the state $S \rightarrow B$, such that for each rule $\sigma_L \rightarrow \sigma_R$, we have that σ_L is $\overleftarrow{\mathrm{B}}$-reachable and σ_R is A-reachable. Each branch of a parse tree corresponds to a minimal set of productions rules such that B is A-reachable and A is $\overleftarrow{\mathrm{B}}$-reachable, removing any rule from this set would cause reachability to be lost for that branch, and hence the whole tree would not preserve bi-directional reahcability. Each parse tree therefore corresponds to a possible MinA, dependent only upon a positive subsumption test.

It must be noted that in the worst case, there is an exponential number of parse trees. The Earley algorithm computes all parse trees in parallel in polynomial time. However, extracting an exponential number of parse trees will run in exponential time.

6 Work in progress

In this section we outline some modifications to the Earley algortihm to improve its efficiency in terms of MinA extraction.

During its search for parse trees, the predictor procedure expands all production rules for a non-terminal symbol it encounters. Terminal symbols are not expanded and are handled by the scanner procedure. We note that, in our case, when a concept C becomes A-reachable, future expansions of production rules for C are unneccesary. When a specific reachability path between the concept A and C has been found, we never need to traverse that path again, and the symbol C effectively becomes a terminal symbol.

The algorithm may therefore be improved by introducing a dynamic terminal set. That is, initially only the symbol A is a terminal symbol. When any symbol becomes A-reachable we add it to the set of terminals and remove it from the set of non-terminals.

The completer procedure forms the core of marking parse trees. It keeps track of the production rules responsible for completions; for every symbol in a production rule, it maintains a list of pointers to other states responsible for completing it. Having a dynamic terminal set complicates this bookkeeping process and requires changes to the data-structures used, as well as the completer and predictor procedures.

1. **Data structures:** We introduce an array such that, for each concept/symbol that becomes A-reachable, we maintain a set of pointers to states, responsible for completing the symbol. Each pointer records the state in which the symbol becomes A-reachable, i.e. whenever the completer is run, a pointer to this state entry is recorded in the array of pointers for the symbol being completed.
2. **Predictor:** The predictor procedure normally expands all relevant production rules for a symbol and adds new states to the current chart. It never adds the same state more than once to the current chart. In a different chart however it may expand the same symbol again. We restrict the procedure so that it may never introduce a production rule more than once, irrespective of the chart it which it occurs.
3. **Completer:** For each state completed the completer stores a pointer to the state in the array above for the symbol being completed. Every symbol completed also gets marked as a terminal symbol. The changes in the predictor further neccesitates that once a new symbol D becomes a terminal, that the completer completes all states $\alpha \rightarrow \sigma \bullet D$ in any prior chart, and that the scanner be called for each state $\alpha \rightarrow \sigma \bullet D$ in the current chart.
4. **Production states:** Production states no longer require an index to mark their originating charts, because, for each state, the new completer procedure will scan all previous charts for symbols to the right of the dot to complete, and not only those from the chart the state originated from.

These optimizations have the potential for a more efficient algorithm. The problem of non-termination described earlier is no longer relevant, since every production rule can only ever be introduced once by the predictor. From this we have that the only way a production rule occurs more than once in any chart is by virtue of the scanner or completer procedures. Each of these advances the dot in some way and new symbols may need to be expanded, but these expansions can only be done by the predictor which would never expand any production rule more than once.

The proposed modified Early algorithm is listed in Table 3. Before the algorithm is executed we obtain $\mathcal{CFG}_{\mathcal{O}}$. For the entailment $\mathcal{O} \models A \sqsubseteq B$ the appropriate substitutions have been made and the start state $S \rightarrow B$ added to $\mathcal{CFG}_{\mathcal{O}}$ and chart[0]. The set TERMINALS represent the set of all terminals, initialised to $\{A\}$, NONTERMINALS the set of non-terminals initialised to $\mathrm{Sig}(\mathcal{O}) \backslash A$, and REF[$\alpha$] an initially empty array which will contain pointers to all states where the symbol α has been completed.

Once the algorithm terminates all MinAs still need to be extracted. The process is similar to the method used to extract the parse trees from the original Earley algorithm. The algorithm proceeds in a standard depth-first manner. Starting with the completion references for the symbol S, select the production rule referenced. Let this state be $S \rightarrow \alpha$. Then for each symbol $x_i \in \alpha$ choose a production rule from $\mathrm{REF}(x_i)$; this process continues recursively. Once no new production rules can be added, the set of all states represent a possible MinA. Mapping back to the origional axioms in normal form the set can be tested for subsumption, and if subsumption holds the MinA is valid. More parse trees may be extracted by backtracking and making alternate choices where $|\mathrm{REF}(x_i)| > 1$.

We use the standard example in the literature (Brandt 2004), showing that there exists an ontology $\mathcal{O}$ such that it contains exponentially many MinAs

Table 3: Modified Earley algorithm

function EARLY-PARSE **returns** chart
cIndex = 0
do
  for each state **in** chart[cIndex] **do**
    if next symbol ∈ NONTERMINALS **then**
      PREDICTOR(state, cIndex)
    elseif next symbol ∈ TERMINALS **then**
      SCANNER(state, cIndex)
    else
      COMPLETER(state, cIndex)
  end
while(hasNextChart)
return chart

procedure ENQUEUE(state, chart-entry)
if state not in chart-entry **then**
  PUSH(state, chart-entry)

procedure SCANNER($(A \rightarrow \alpha \bullet B\beta)$, cIndex)
if $B \in$ TERMINALS **then**
  ENQUEUE($(A \rightarrow \alpha B \bullet \beta)$, chart[cIndex+1])

procedure PREDICTOR($(A \rightarrow \alpha \bullet B\beta)$, cIndex)
if $B \in$ NONTERMINALS **and**
if no B-productions have been expanded **then**
  ENQUEUE($(B \rightarrow \bullet\alpha\beta)$, chart[cIndex])
    for all production rules for B

procedure COMPLETER($(B \rightarrow \gamma\bullet)$, cIndex)
REF[B] += Pointer($B \rightarrow \gamma\bullet$)
TERMINALS += B
NONTERMINALS -= B
for each $(A \rightarrow \alpha \bullet B\beta)$ **in** chart[0 → cIndex-1] **do**
  ENQUEUE($(A \rightarrow \alpha B \bullet \beta)$, cIndex)
if B is a new terminal **then**
  for each $(A \rightarrow \alpha \bullet B\beta)$ **in** chart[cIndex] **do**
    SCANNER($(A \rightarrow \alpha \bullet B\beta)$, cIndex)

for an entailment, and show how the improved Earley algorithm can be used to extract all MinAs.

Example 7 *Let $\mathcal{O}$ be an $\mathcal{EL}^+$ ontology consisting of the axioms:*

$\alpha_1 : A \sqsubseteq P_1 \sqcap Q_1$ $\quad \alpha_4 : P_2 \sqsubseteq B$
$\alpha_2 : P_1 \sqsubseteq P_2 \sqcap Q_2$ $\quad \alpha_5 : Q_2 \sqsubseteq B$
$\alpha_3 : Q_1 \sqsubseteq P_2 \sqcap Q_2$

$\mathcal{O}$ in normal form is:
$\omega_1 : A \sqsubseteq P_1$ $\quad \omega_2 : A \sqsubseteq Q_1$ $\quad \omega_3 : P_1 \sqsubseteq P_2$
$\omega_4 : P_1 \sqsubseteq Q_2$ $\quad \omega_5 : Q_1 \sqsubseteq P_2$ $\quad \omega_6 : Q_1 \sqsubseteq Q_2$
$\omega_7 : P_2 \sqsubseteq B$ $\quad \omega_8 : Q_2 \sqsubseteq B$

Then $\mathcal{CFG}_\mathcal{O}$ for the entailment $\mathcal{O} \models A \sqsubseteq B$ is:

$\sigma_1 : S \rightarrow B$ $\quad \sigma_4 : B \rightarrow Q_2$ $\quad \sigma_7 : B \rightarrow P_2$
$\sigma_2 : Q_2 \rightarrow Q_1$ $\quad \sigma_5 : Q_2 \rightarrow P_1$ $\quad \sigma_8 : P_2 \rightarrow Q_1$
$\sigma_3 : P_2 \rightarrow P_1$ $\quad \sigma_6 : Q_1 \rightarrow A$ $\quad \sigma_9 : P_1 \rightarrow A$

The chart returned by the algorithm consist of only two chart entries for this problem as shown in Table 4. With the final completion reference list shown in Table 5. Extracting all parse trees using a depth first search results in all the MinAs being extracted for the problem as shown in Table 6.

Table 4: Solution chart

Chart 0

1.	$S \rightarrow \bullet B$	Initial State
2.	$B \rightarrow \bullet Q_2$	Predictor from 1
3.	$B \rightarrow \bullet P_2$	Predictor from 1
4.	$Q_2 \rightarrow \bullet P_1$	Predictor from 2
5.	$Q_2 \rightarrow \bullet Q_1$	Predictor from 2
6.	$P_2 \rightarrow \bullet P_1$	Predictor from 3
7.	$P_2 \rightarrow \bullet Q_1$	Predictor from 3
8.	$P_1 \rightarrow \bullet A$	Predictor from 4 and 6
9.	$Q_1 \rightarrow \bullet A$	Predictor from 5 and 7

Chart 1

10.	$P_1 \rightarrow A\bullet$	Scanner from 8
11.	$Q_1 \rightarrow A\bullet$	Scanner from 9
12.	$P_2 \rightarrow P_1\bullet$	Completer (10-6)
13.	$Q_2 \rightarrow P_1\bullet$	Completer (10-4)
14.	$Q_2 \rightarrow Q_1\bullet$	Completer (11-5)
15.	$P_2 \rightarrow Q_1\bullet$	Completer (11-7)
16.	$B \rightarrow P_2\bullet$	Completer (12-3), (15-3)
17.	$B \rightarrow Q_2\bullet$	Completer (13-2), (14-2)
18.	$S \rightarrow B\bullet$	Completer (16-1), (17-1)

Table 5: Completed reference list

REF[S]	=	[18]	REF[B]	=	[16, 17]
REF[Q_2]	=	[13, 14]	REF[P_2]	=	[12, 15]
REF[Q_1]	=	[11]	REF[P_1]	=	[10]
REF[P_1]	=	[10]			

Table 6: Extracted MinAs

$MinA_1$:	18	16	12	10
$MinA_2$:	18	16	15	11
$MinA_3$:	18	17	13	10
$MinA_4$:	18	17	14	11

Mapping back to the origional axioms we have:

$MinA_1$:	α_4	α_2	α_1
$MinA_2$:	α_4	α_3	α_1
$MinA_3$:	α_5	α_2	α_1
$MinA_4$:	α_5	α_3	α_1

7 Conclusion and future work

The combinatorial nature of MinA extraction makes it an inherently hard problem, with most approaches extracting a module based on reachability or syntactic locality. The set of axioms within this module is then systematically reduced by various methods, after which subsumption tests determine if the desired entailment still holds. Though these aproaches work well, the cost of repetitive subsumption testing is prohibitive. It is therefore desirable to eliminate as many axioms as possible before each subsumption test is performed. To this end partition methods can be employed, with the hope of eliminating large chunks of axioms that do not play a role in an entailment.

The Earley algorithm presented, based on bidirectional reachability, aims to extract all reachability based paths for an entailment directly, without first extracting smaller modules. Each parse tree extracted by the algorithm corresponds to a minimal axiom set such that reachabilty between the sub- and super-concepts in an entailment is preserved. A standard subsumption test is then performed to test

whether the axiom set is a valid MinA. This has the potential to reduce the number of subsumption tests drastically since for each parse tree the Earley algorithm extracts, the set of axioms extracted is minimal. No additional procedures need to be employed to further reduce the set of axioms and only a single subsumption test is neccesary in order to determine if the set represents a valid MinA.

We require two mapping layers, the first mapping between the original axioms in the ontology and the axioms in the normal form, the second between the normal form axioms and the production rules. Though these mappings may seem to introduce a high memory and computational overhead, in our opinion they perform an important function in debugging ontologies. Consider the axiom $A \sqsubseteq B \sqcap C$ which forms part of a MinA for some entailment, where only concept C actually plays a role in the entailment. The mappings allow us to identify exactly which concepts play a role in the entailment. Therefore instead of just presenting whole complex axioms for debugging, we have the ability to highlight exactly which concepts within the axioms are relevant to the entailment.

There are two possible problems with our approach: The first being that parse trees are minimal bi-directionally preserving axioms sets, and since they are minimal, it may occur that that all such sets are only subsets of a MinA. Thus not all MinAs may be obtained as parse trees. This boils down to the completeness question of the algorithm i.t.o. finding justifications. The second issue is that there does not exist a one-one correspondence between the axioms in the different mapping layers. Therefore when mapping back from a minimal parse tree to original axioms, we may find that the set of axioms is not a MinA anymore, in that it contains extra axioms. Though we do not directly address these issues in the current paper, we believe that the ideas presented in this paper, are both interesting and promosing, and as such warrant further investigation.

For future work we intend to implement the algorithm as a plugin for the widely used ontology editor Protégé[1] in order to test its usefulness in practise on large scale ontologies, as well as to optimize it as much as possible. If the algorithm proves useful we will investigate the possibility of extending it towards more expressive DLs. We also aim to investigate the possible link between our appraoch and automata-based pinpointing approaches (Peñaloza 2008).

References

Baader, F., Brandt, S. & Lutz, C. (2008), Pushing the $\mathcal{EL}$ envelope further., *in* K. Clark & P. F. Patel-Schneider, eds, 'OWLED 2008 DC Workshop on OWL: Experiences and Directions'.

Brandt, S. (2004), Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and – what else?, *in* R. L. de Mántaras & L. Saitta, eds, 'ECAI-2004: Proceedings of the 16th European Conference on Artifcial Intelligence', IOS Press, pp. 298–302.

Chiang, Y. & Fu, K. (1984), 'Parallel parsing algorithms and VLSI implementation for syntactic pattern recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(3), 302–314.

Earley, J. (1970), 'An efficient context-free parsing algorithm', *Communications of the Association for Computing Machinery* **13**(2), 94–102.

[1] http://protege.stanford.edu/

Grau, B. C., Horrocks, I., Kazakov, Y. & Sattler, U. (2007), Just the right amount: Extracting modules from ontologies, *in* C. Williamson & M. Zurko, eds, 'WWW '07: Proceedings of the 16th International Conference on WWW', ACM, New York NY, USA, pp. 717–726.

Grau, B. C., Horrocks, I., Kazakov, Y. & Sattler, U. (2008), 'Modular reuse of ontologies: Theory and practice', *Journal of Artifcial Intelligence Research* **31**, 273–318.

Jianfeng Du, G. Q. & Ji, Q. (2009), Goal-directed module extraction for explaining OWL DL entailments, *in* K. Thirunarayan, ed., 'ISWC'09: Proceedings of the 18th International Semantic Web Conference'. To appear.

Jurafsky, D. & Martin, J. (2009), *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*, 2 edn, Prentice Hall.

Pavlatos, C., Koulouris, A. & Papakonstantinou, G. (2003), Hardware implementation of syntactic pattern recognition algorithms, *in* M. Hamza, ed., 'IASTED International Conference on Signal Processing and Pattern Analysis', Acta Press, pp. 360–365.

Peñaloza, R. (2008), Automata based pinpointing for DLs, *in* F. Baader, C. Lutz & B. Motik, eds, '21st International Workshop on Description Logics', CEUR Workshop Proceedings.

Suntisrivaraporn, B. (2009), Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies, PhD thesis, Technical University of Dresden.

Reflecting on Ontologies Towards Ontology-based Agent-oriented Software Engineering

G. Beydoun[1], B. Henderson-Sellers[2], J. Shen[1], G. Low[3]

[1]{beydoun, jshen} @uow.edu.au, School of Information Systems and Technology, University of Wollongong, Wollongong

[2] brian@it.uts.edu.au, School of Software, University of Technology, Sydney

[3] g.low@unsw.edu.au, School of Information Systems, Technology and Management, University of New South Wales, Sydney

Abstract

"Ontology" in association with "software engineering" is becoming commonplace. This paper argues for the need to place ontologies at the centre of the software development lifecycle for multi agent systems to enhance reuse of software workproducts as well as to unify agent-based software engineering knowledge. The paper bridges the state-of-the-art of ontologies research from Knowledge Engineering (KE) within Artificial Intelligence and Metamodelling within Software Engineering (SE). It presents a sketch of an ontology-based Multi Agent System (MAS) methodology discussing key roles on ontologies and their impact of workproducts, illustrating these in a MAS software development project for an important application that utilizes dynamic web services composition.

Key words: *Software Development Lifecycle (SDLC), Ontologies, Agents, Multi Agent Systems (MAS), Services*

1 Introduction

This paper promotes ontology-based software development with a current focus on methodologies for building a multi agent[1] system (MAS). Substantial integration between ontologies and software engineering has been achieved e.g. in ODE of (Falbo *et al.*, 2005) and Onto (Leppänen, 2007). This paper is part of an ongoing effort to place ontologies at the centre of the software development lifecycle (SDLC) for MASs to enhance the reuse of MAS workproducts as well as to unify agent-based software engineering knowledge.

In a MAS composed of a heterogeneous collection of agents with distinct knowledge-bases and capabilities, coordination and cooperation between agents facilitate the achievement of global goals that cannot be otherwise achieved by a single agent working in isolation (Wooldridge, 2000). The unique characteristics of a MAS have rendered most standard systems development methodologies inapplicable, leading to the development of Agent Oriented Software Engineering (AOSE) methodologies. Several AOSE methodologies exist (Henderson-Sellers and Giorgini, 2005). Indeed any one of the extant methodologies has limited applicability (Tran and et al, 2005) e.g. to a specific domain or a specific type of software application. This limits adoption of AOSE. Furthermore, a review (Tran and Low, 2005) of sixteen prominent AOSE methodologies revealed that most ignore system extensibility, maintenance, interoperability and reusability issues. This imposes a second barrier to the adoption of AOSE. This paper outlines a path towards resolution of both of these barriers through the use of ontologies during the software development lifecycle. Given that the "fixed costs" associated with learning or configuring methodologies to suit the requirements of a given project are high, it is critical to address these concerns and protect the various facets of investments associated with using a MAS including: interoperability of systems, reuse of their components, reuse of human skills acquired and reuse of designs generated during development.

As a first step towards using ontologies as a central software engineering construct throughout the whole development lifecycle of a MAS, this paper reviews the state-of-the-art of ontology research in two key communities: the Artificial Intelligence (AI) community and the Information Systems (IS)/Software Engineering (SE) community. Much of our understanding of ontologies has been derived from the AI community; in contrast, the IS/SE community have focussed on the use of a systematic relationship and understanding of models and metamodels. To illustrate how ontologies can be central to MAS development, we use an example application that also highlights the power of agents. The example chosen is a MAS Peer to Peer system constructed to allow dynamic composition of web services in highly distributed and heterogeneous environments.

The rest of the paper is organised as follows: Section 2 provides a conceptual analysis bridging software engineering concepts and existing ontology research emanating largely from the knowledge engineering community. Section 3, using the grounded position on what an ontology can do to the SDLC, provides an argument placing ontologies at the heart of SDLC specifically tailored for Agent Oriented Software Engineering. Section 4 develops this into a sketch of an AOSE ontology-based methodology. Section 5 illustrates key concepts in an application. Section 6 concludes with a summary and discussion of future work.

2 Background: Bridging Ontologies in KE to Models and Metamodels in SE

In SE, terms such as model, metamodel and ontology are often used with disparate meanings across the literature even within the same sub-domain of SE. To pin down the appropriate usage of an ontology within the SDLC of a methodology, it is important to describe how an ontology may be linked to a model and/or a metamodel and,

[1] Agents are highly autonomous, situated and interactive software components. They sense their environment and respond accordingly.

indeed, how models and metamodels are defined and inter-related. This leads to the need to understand the relationship between ontologies and a metamodelling hierarchy such as that of the (OMG, 2005a) or (ISO/IEC, 2007). (Favre *et al.*, 2007) note the lack of a general, systematic technique to map between metamodels and ontologies which is the focus of this section.

In SE, additional characteristics for an ontology are required. It is widely agreed that it needs to be formal e.g. (Corcho *et al.*, 2006; OMG, 2005b; Guizzardi, 2005; Rilling *et al.*, 2007). However the meaning of 'formal' is not very well agreed. For example, (Corcho *et al.*, 2006) suggests it to mean 'understandable by a computer', OMG suggests it to mean underpinned by a metamodel and (Guizzardi, 2005) uses "formal" to mean "having form" rather than precise or mathematical. A second required characteristic is that it should represent *shared* knowledge e.g. (Gruber, 1993; Noy and McGuinness, 2001) and a third characteristic is that an ontology is represented by a vocabulary (Gruber 1993; Guarino 1998). This last notion is used to differentiate between an ontology linked to a representation in a specific vocabulary *but with a common conceptualization (*Guarino 1998). Following (Guarino, 1998), (Ruiz and Hilera, 2006; Guizzardi, 2005) identify four general kinds of ontologies: *high-level ontologies* (or upper level ontologies)[2], domain ontologies, task ontologies, and application ontologies. This is a scheme that will underpin our ontology-centric SDLC to be detailed in Section 4. This is in accordance with (Ruiz and Hilera, 2006) as shown in Figure 1 which also compares two classification schemes (of (Guarino, 1998 and Fensel, 2004)) and differentiates between domain-independent ontologies and domain-dependent ontologies (a discrimination also adopted in this paper).

To link ontologies to metamodels in current SE, two stacked architectures are commonly used. It is worth noting the OMG architecture based on strict metamodelling wherein the only inter-level relationship permitted is "instance of" (in Figure 2). This is not universally accepted within SE, for instance, the architecture used in ISO/IEC 24744 (ISO/IEC, 2007) (Figure 3) uses the powertype pattern (Gonzalez-Perez and Henderson-Sellers, 2006), which permits both instance-of and generalization relationships between levels. Indeed, as observed in several papers summarized in (Gonzalez-Perez and Henderson-Sellers, 2008) application of the four layer hierarchy used by the OMG to methodologies results in several contradictory situations – hence the creation of the newer architecture in Figure 3. For our purpose, we can say that a metamodel describes a domain that is representative of more than one instance in a less abstract domain and, importantly, each model/metamodel describes a domain of discourse, the *language used* for a metamodel domain and a model domain (although relative) is distinct.

[2] Uschold (2005) suggests that, while an upper-level ontology is important, it is less important *which* such ontology is used. In fact, we omit upper level ontology from our methodological sketch in Section 4.

We can now ask which 'metamodels' or 'models' (or both) are useful, both theoretically and pragmatically, to link our SE-defined "ontology" definition. (Atkinson *et al.*, 2006) suggest that ontologies and models may be different technologies since they appear to be derived from different subfields of computing and knowledge representation and there appear to be several projects, for example within the OMG and W3C, aimed at producing a bridge between the technologies. Their conclusion is that ontologies are a subset of models since ontologies fulfil the criteria for being models but have additional characteristics i.e. they are specializations in the object-oriented (OO) sense.

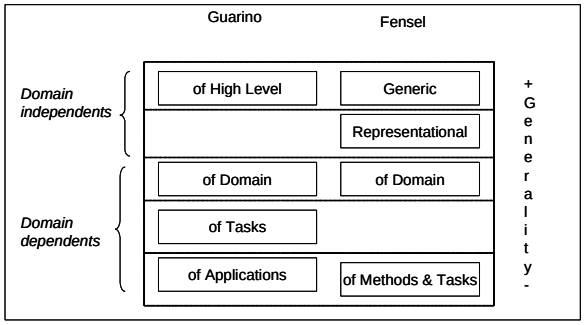


Figure 1 Ontologies by generality level (after (Ruiz and Hilera, 2006))

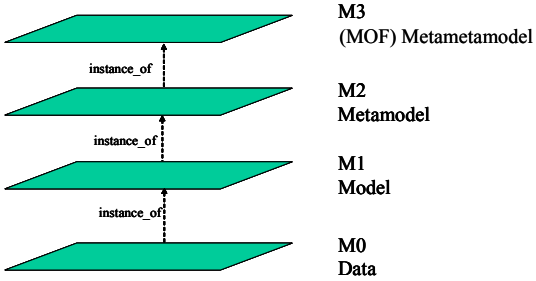


Figure 2 The 4 layer hierarchy of the OMG- based on (ANSI, 1989) (after (Henderson-Sellers and Unhelkar, 2000)) ©Pearson Education Limited

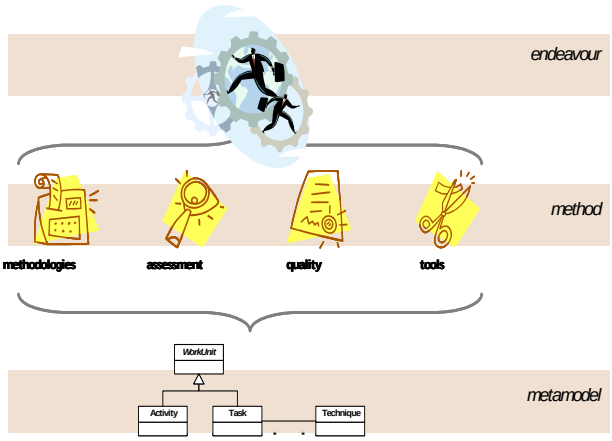


Figure 3 Three layer architecture of ISO/IEC 24744 International Standard (after (Henderson-Sellers, 2006))

While noting that much of ontology design originated in OO design, (Noy and McGuinness, 2001) suggest that OO stresses operational rather than the structural properties of classes, which are the focus of ontology design. This suggests an alignment with data models. On the other hand, the equivalencing of models with database-focussed models, as is done by (Ruiz and Hilera, 2006), unnecessarily restricts the meaning of model for

such a comparison to be useful here. In contrast, (OMG, 2005b) takes a broader meaning to the term "conceptual model". It notes some missing concepts in the UML – in particular, the treatment of disjoint classes, set intersection and set complement. They argue that ontology instances may also be required without the prior definition of a class (not permissible using UML).

Many other authors equate ontologies with models despite noting the difference in intent i.e. that an ontology is descriptive and a model typically (but not always) prescriptive e.g. (Wand and Weber, 2005; Ruiz and Hilera, 2006). For example, (Gruber, 1993) states that "Ontologies are also like conceptual schemata in database systems" which "provide a logical description of shared data"; and (Guarino, 1998) clearly indicates that he regards an ontology as belonging to the model domain and not the metamodel domain. (Ruiz and Hilera, 2006) suggest differences based on arguing that an ontology is descriptive whereas a metamodel is prescriptive, belonging to the solution domain.

In the context of agent modelling languages, (Guizzardi and Wagner, 2005a) propose a unified foundational ontology (UFO). The UFO is categorized as an upper level ontology (a.k.a. foundational ontology), and an application to business modelling is given in (Guizzardi and Wagner, 2005b). (Guizzardi, 2005) states that a foundational ontology is a *meta-ontology*. Since he, and others, effectively equates "ontology" with "model", then we must conclude that a meta-ontology can be effectively equated with metamodel, at least in the OMG sense. Indeed, in (Guizzardi and Wagner, 2005a) it is clearly stated that a foundational ontology can be represented as a MOF (Metaobject Facility) model, MOF being a language for defining modelling languages i.e. it is used as a metamodelling language. In other words, a foundational ontology is at the metamodel level in that it is equivalent to the UML or the ER definition. This means that we need to reassess Figure 1 because "domain independence" is also seen as a feature of a meta-ontology whilst, in contrast (see Figure 1) a generic model is widely recognized as *not* being at this meta level. In a section entitled *"combining metamodels and ontologies to achieve semantic interoperability"* – words suggesting that ontologies belong to the metalevel – (Karagiannis *et al.*, 2008) go on to describe "semantic mappings between metamodel elements and ontology concepts". Arguably this latter statement, at odds with the former, can be interpreted as ontology concepts being the classes in the ontology metamodel – as for instance documented in the OMG's Ontology Defintion Metamodel (ODM) (OMG, 2005b).

Contrasting several chapters from the same book (Calero *et al.*, 2006), we see that while the software maintenance ontology of (Anquetil *et al.*, 2006) and the software development environment ontology of de (Oliveira *et al.*, 2006) clearly discuss a domain ontology, the ontology for software measurement of (Bertoa *et al.*, 2006) and the ontology for software development methodologies and endeavours are all clearly defined in terms of a metamodel. Indeed, (OMG, 2005b) clearly differentiates between the OWL *metamodel* that allows users to define ontology models and the ontology that is "generally specified as a system of classes and properties (the structure) which is populated by instances (the extents)". Hence, the UoD is described by a set of ontologies where ontologies are used to enhance the target system and be complementary to UML modelling artefacts. In other words, ontologies belong to the M1 level (Figure 2) or Method Domain (Figure 3) since an ontology is a conceptual model (OMG, 2005b), sharing characteristics with more traditional data models. This OMG ODM approach suggests a multi-level ontology architecture (Figure 4). Here, the "M2" level is equivalent with (Guizzardi and Wagner, 2005a)'s foundational ontology, with the OMG's ODM and with the term "upper level ontology". The "M1" level includes not only domain-specific ontologies (such as that for, say, a banking domain) but also a domain-independent generic ontology (cf. Figure 1). Instances of elements of a domain-specific ontology (Figure 4) are discussed in (Noy and McGuinness, 2001) where it is argued that the depth in the ontology hierarchy at which this occurs is context dependent, making no attempt to align with the strict metamodelling architecture of Figure 2.

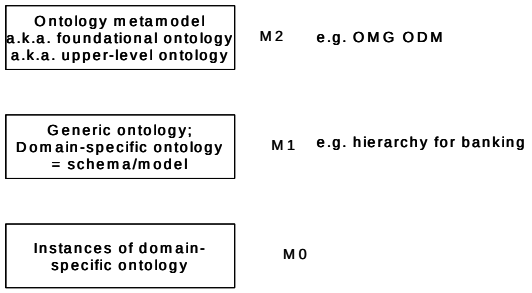


Figure 4. Three level ontology architecture suggested by OMG.

If an ontology refers to a universe of discourse and to conceptualization, as according to (Gruber, 1993), then the term "ontology" would appear to be equally applicable to either M1 or M2 (although not both simultaneously), in just the same way that the term "model" can be applied to a M1 UML visualization (e.g. a system design) or to a M2 visualization (e.g. the UML metamodel). This may explain the ambiguity regarding whether an ontology is an M1 or M2 thing. In some contrast to the notion of ontologies being focussed at their specification level i.e. the metamodel, most "ontologies" found by a web search and documented, for example in Protégé, are hierarchies of terms in a specific (often commercial) domain. For instance, we have located an ontology for newspaper publishing containing elements such as editor, journalist and printing press; an ontology for health care with concepts including doctor and patients. Such an ontological hierarchy bears a good correspondence to a UML model (M1) that might be constructed if one were building software as opposed to the ontological usage of knowledge.

3 Why Must Agent-Oriented SE be Ontology-Centric?

Many of the IS/SE focussed application areas are brought together in (Green and Rosemann, 2005), a volume on business systems analysis. However, there remain only a small number of existing MAS methodologies that include ontologies in their workproducts and processes. This support is generally confined to the early phases of

the development (the *analysis* phase). For example, (Girardi and Serra, 2004) specify how a domain model that includes goal and role analyses is developed from an initial ontology. Another example (DiLeo et al., 2002) uses ontologies to mediate the transition between goal and task analyses. An ontology-based methodological framework that can be used to build new ontology-centric AOSE methodologies from scratch, or a repository of add-on methodological elements that can be added to an existing AOSE methodology to enhance it with new support for ontology-based AOSE, would be a significant innovation in the support for ontology-based AOSE.

In addition, while existing methodologies suffer from other deficiencies (Tran and Low, 2005), there is a growing realization that some form of consolidation is needed. To merge this existing body of agent-oriented software engineering knowledge into a more effective methodological approach, we consider two key issues: how easy it is for software developers to actually apply the outcome (usability) and how feasible is the merging approach (realisability). We identify the following three candidate approaches:

Approach 1: An ad-hoc approach consisting of merging existing methodologies one at a time, with an arbitrary methodology as a starting point, and without guidance on attaching methodologies, beyond avoiding repetition and inconsistent use of terms.

Approach 2: A metamodelling method engineering approach characterised by having a formal unifying formal language (a metamodel) to express various methodology fragments from different sources.

Approach 3: A feature-identification-guided approach to identify AOSE development steps and modelling concepts from existing AOSE methodologies to produce a unified methodological framework that in turn can be used to easily generate methodologies as required.

Approach 1 does not offer any guide on the scope of software development lifecycle concepts and can lead to one of two types of errors: assuming differences of concern when none exists, or falsely assuming similarity of concern because of the common use of terms. The first type of error may lead to repetition and to an unnecessarily large and cumbersome methodology, rendering it less accessible to developers. Tolerating errors of the first type, a successful unification effort would result in a *large* methodology with its bulk concerned with a collection of 'exceptional cases' without common structures. We find that this is exactly what happened with UML (in a slightly different domain but nevertheless providing a highly relevant parallel). The second type of error can create inconsistencies because of inconsistent interpretations of terms. Tolerating such errors, the resultant methodology would produce inconsistent models and lower its usability, as software developers subsequently struggle to deal with problems resulting from inconsistencies and would most likely lead to its abandonment (Bernon et al., 2004).

Approach 2 requires a formal language, a metamodel, whose units serve to generate methodology fragments with similar concerns, but with a different flavour according to the context of the development project. This approach has been the focus of (Beydoun et al., 2006b; Beydoun et al, 2009). In this approach, the development project decides the concern and the flavour of the methodology generated rather than subjective 'interpretations' skewed towards a forced merging between methodologies and their fragments (as in Approach 1). Such interpretations are avoided, preventing any inconsistencies. However, to avoid inconsistencies only a select subset of the rewritten components of methodologies can be integrated at any one time. For example, in every given object-oriented development project (Brinkkemper et al., 2001), a customised integration of selected components is required. For an emerging area of application such as MASs, development experience is limited and the criteria of selection are not yet easily discerned. Hence, the benefit in the applicability of this approach does not outweigh the added effort required for assembling selected method components. Consequently, to balance the work on Framework Agent Modelling Language (FAML) (e.g. (Beydoun et al., 2009)), in this paper Approach 3 is pursued as an alternative, and potentially complementary, approach to a method engineering approach (Henderson-Sellers, 2003) with the aim to explore cross fertilisation between the two approaches. For example, the ontology techniques developed for Approach 3 will be used to enhance the method engineering repository of Approach 2. Approach 3, guided by feature-identification, does not require the cumbersome re-writing of existing methodologies using a common formal language (metamodel) as in Approach 2. It is sufficient to validate and refine the set of candidate steps and modelling concepts and overlay these on top of existing methodologies. Hence, this approach requires much less effort and it is the most realizable as it does not require the collaboration of the creators of the existing methodologies. Crucially, this approach rids developers of the highly specialised and difficult task of the merging of methodology components on a per project basis. The approach instead relies on using explicit ontologies as a focal point during development to facilitate combining features from different AOSE methodologies. This will use ontologies as a means for semantic mappings to convert software work products to suit various development steps. This can substantially support integration of processes and products; and, for the finally implemented MAS, this can support its inter-operation with other systems.

Using off-the-shelf domain ontologies as a starting point of system development, will become the focus of our efforts on the applied use of ontologies in an AOSE methodology (not their actual creation). This will enable the transfer and adaptation existing techniques for ontologies (e.g. techniques for mapping and translating between multiple ontologies) to obtain a more economical approach to MAS development, addressing interoperability and work product reuse. Not only will an ontology-based AOSE methodology be complete and consistent and produce systems that can easily be evolved to new contexts but, in addition, it can have a highly developed maintenance phase to guide developers in

reusing existing systems and components previously developed (using an ontological approach). This will foster wider deployment of agent-based systems by industry by focussing on the commercial success of the technology.

At least three significant contributions to the state-of-the-art in AOSE are identified: firstly, designers will have a tested and verified framework to handle interoperability issues in an heterogeneous environment at design time by allowing a MAS to be formed from loosely coupled components connected through ontological mappings. Thus, they will be inherently flexible and their actual design and architecture will be reusable across applications and in different settings. Secondly, ontological commitments related to a MAS will be explicitly integrated with its actual design and development. In exploring the currently overlooked ontology-related interactions between the analysis and design phases of software development for MAS, iterative verification during the design and development of the system will become possible, increasing the likelihood of producing a correct system. Thirdly, all key concerns of AOSE practitioners will be combined into one methodological framework. The first two contributions are actually interrelated: The explicit and extensive support for *ontology-based* MAS development will address the interoperability concerns in heterogeneous environments.

4 From ontologies in SE to Ontology-based Agent Oriented SE

Inclusion of ontologies into a specific SE methodology for the development of MAS permits the long term reuse of software engineering knowledge and effort and can produce reusable MAS components and designs. (Beydoun et al., 2006a) argue that using ontologies in developing a MAS is complicated by having to simultaneously provide knowledge requirements to different Problem Solving Methods[3] that are still required to share results using a common terminology. This is even further complicated because individual PSMs may operate at different levels of abstraction of the domain, they may be complementary, and they may have varying degrees of prescription to the domain requiring various degrees of adjustment to suit the domain. A set of six requirements were proposed for developing a MAS using an ontology-based software engineering approach. In this section, we present the methodology sketch motivated by the original drive for using ontologies for *reuse* (as also discussed in (Beydoun et al, 2006a)). Specifically, we propose the unification of and reuse of AOSE knowledge (as outlined briefly in the previous section). As targeted by this methodology, the role of ontologies during the SDLC is detailed. Similar to KBS development, it is assumed that the choice of PSM may be made independently of domain analysis. Moreover, it is also assumed that a domain ontology describing domain concepts and their relationships is available. Such an ontology may be available from an existing repository e.g. (DARPA, 2000) or a domain analysis may be considered the first stage of developing the system. The purpose of such a domain analysis would only be to identify concepts and their relationships.

There is inter-play between the role of reuse and other roles of ontologies in a MAS. Various reuse roles cannot be smoothly accommodated (e.g. interoperability at run-time) without careful consideration of run-time temporal requirements. For example, an ontology's role in reasoning at run-time is based on fulfilling PSM knowledge requirements at design time. This requires scoping domain analysis for each individual agent at design time. The key to ontology-based design of a MAS is the appropriate allocation of a PSM to individual agents in order to match system requirements. Towards this, we note that goal analysis is the usual way to express requirements e.g. (Giunchiglia et al, 2003; Wooldridge et al, 2000) and we suggest associating PSMs (using PSM libraries) and system goals in the early stages of a MAS design. The ontologies provides a conceptualization and the basis upon which a machine accessible definition of PSMs may be created (similar to (Fensel, 1997)).

We envisage that the MAS development starts with a domain ontology, an application ontology and a collection of task ontologies used to identify goals and roles of the agents in the system. This in turn is used to index an appropriate set of problem solving capabilities from an appropriate existing library of capabilities. Individual ontologies corresponding to the requirements of each capability are then extracted from the initial common ontology in order to provide knowledge representation and allow reasoning by individual agents. Those ontologies will form the basis for an iterative process to develop a common communication ontology between all agents and verify the knowledge requirements of chosen capabilities. Individual localised ontologies may also require incremental refinement during the iterative process. Appropriate ontology mappings are needed between local ontologies and the communication ontology. To be complete, the methodology needs technical guidelines to develop the various ontology mappings, operators to extract localized agent ontologies from the domain ontology, operators for consistency checking between related ontologies and support for managing reuse tasks in the maintenance phase of the methodology.

The SDLC requires three related ontologies (shown in Figure 5): First is a domain ontology to describe the domain knowledge for the problem and the requirements for a solution to the problem. Domain ontologies may be unique to the problem itself or may be adapted from previous problems in similar domains. Second is problem-type ontology to describe types of problems to which PSMs have been developed to solve. The problem-type ontology is necessary for defining the PSM interface (capabilities and preconditions). In the construction of a PSM library, the problem-type ontology is necessary for indexing suitable PSMs. Third is a PSM ontology to describes knowledge required for the tasks, control structure, and PSM dependencies. An agent that seeks to dynamically select a PSM (or its coded implementation) to solve a problem needs to know this ontology.

[3] PSMs are high-level structures that describe a reasoning process employed to solve general problems (Rodríguez et at., 2003)

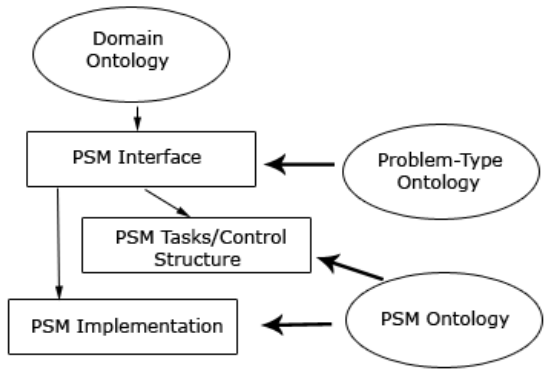


Figure 5 illustrates the role that the ontologies play in PSM implementations. We omit upper level ontologies, we domain ontologies, application ontologies (Problem-type), task ontologies (or PSM ontology)

The collection of all PSMs for local goals should also be verified for completeness against stated system goals. These goals should also be checked against cooperation potential. (A form of distributed goal interaction evaluation could be done using existing approaches e.g. (van Lamsweerde et al, 1998)). Most current methodologies view the decision of problem-solving mechanisms as a low level design step. In our current view, paralleling KBS development, ontology-based design and development requires elevating this to an early design phase and making it central to a later decision on the communication and interface requirement of each agent (rather than the other way around as in many other methodologies e.g. (Giunchiglia et al, 2003; Wooldridge et al, 2000)).

Chosen problem solving capabilities for different agents in a given MAS do not necessarily have the required degree of domain dependence. Hence, for a PSM chosen for some agents, the ontology required may need to be adapted. For this, the domain ontology and the problem-type ontology (application ontology) are again the most convenient reference point. Ontology mapping (between portions of these two ontologies and the local agent's knowledge) is required to ensure that all PSMs have their knowledge requirement available to their reasoning format (adaptors of (Fensel, 1997) may be useful here).

Agents need to communicate their results and instigate cooperation using a common language. For this purpose, we recommend a global communication ontology (as in (Esteva et al, 2002)), rather than many-to-many individual mappings between agents. Such a communication ontology is most conveniently based on the domain ontology available, and it depends on the individual ontology of each agent. In some cases, an ontology mapping may be required between PSM ontologies and the communication ontology. The same adaptation between the reasoning and domain ontology can be used to map the result of reasoning back to a common communication ontology. Our work so far is geared towards 'extendable closed' systems. In the case of 'open systems', introducing new agents may require *runtime* extension of the communication ontology or some local ontologies to allow cooperation with new agents. This is currently beyond our current scope. It is worth noting, that we never assume that local ontologies for agents are complete from the perspective of the agent. This is a considerable step in the right direction towards implementing completely 'open systems'.

Hierarchical ontologies are one way to have flexible domain ontology refinement for agents according to their PSMs, and to accommodate differences in strength of the PSM of agents. A common hierarchical domain ontology can be used as a starting point for verification during development and for multiple access at multiple abstraction levels depending on the individual knowledge requirement of each agent PSM.

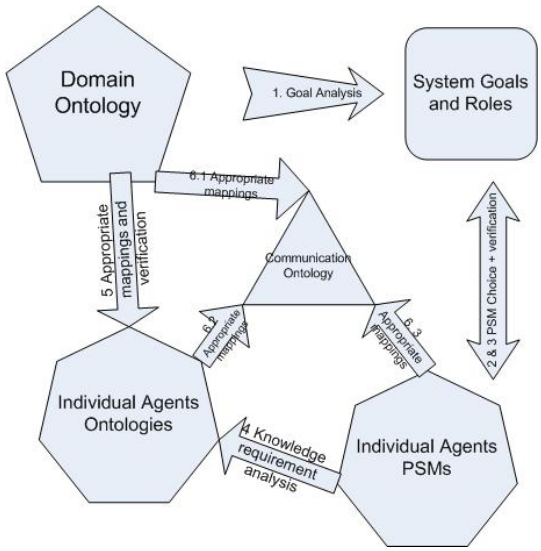


Figure 6. 1. Ontology-based MAS development: Domain Ontology produces Goal Analysis 2. Goal analysis produces a collection of PSMs (using a PSM bank) 3. Knowledge requirement analysis (4). can then be used to delineate local ontologies that can be verified against the domain ontology (step 5). Finally, in step 6 the communication ontology (language) can then be derived using appropriate mappings.

Figure 6 provides the methodological sketch accommodating the observations of this section. The MAS development process starts with a domain and an application ontology (domain-type ontology). These are used to identify goals and roles and to create appropriate interfaces to index an appropriate set of PSMs from a bank of PSMs (see Figure 5 in combination with Figure 6). Appropriate individual ontologies for each PSM are extracted from the initial task ontology. These individual ontologies are used for reasoning by individual problem solvers and may be used to represent results communicated by the individual problem solver. They are next verified against the knowledge requirement of chosen PSMs. The collection of the individualised task ontologies, in combination with the application and domain ontologies, is then used to develop a common communication ontology. Appropriate mappings may be required between individual local ontologies and the communication ontology, to facilitate communicating results between individual agents. Verification between problem solvers and the communication ontology is undertaken, which may result in further localized ontology mappings.

5 Case Study of Ontologies in a MAS application: MAS for Dynamic Web Services Compostion

To illustrate how ontologies can be central to MAS development, we use an example application that also highlights the power of cooperative agents. The application example is a MAS P2P system to allow dynamic composition of web services in highly distributed and heterogenous computing environment and is adapted from (Shen et al, 2007) to highlight how ontologies can be used[4] (using semantically driven composition of services as is often advocated e.g (Souza et al, 2009)). The system will provide, to both service requestors and service providers, Quality of Service (QoS) evaluation. The system will identify service providers' capability and performance so as to enhance the service composition for service clients over the real distributed service network. Due to the complexity of QoS metrics, well-defined QoS service description does not actually exist. With a P2P architecture the QoS is gauged by a service client through cooperative interactions with other peers that can potentially provide the service. The scope of using ontologies in this MAS development is available given that most of the current work focuses on the definition of QoS ontology, vocabulary or measurements and to a lesser extent on a uniform evaluation of qualities, however. Furthermore, a Problem Solving Method unit of analysis nicely corresponds to a service carried by an agent. In this application, the agents themselves will dynamically select PSM implementations that best suit the service or the QoS required. This selection will be made using a P2P searching mechanism to locate appropriate services from other peer agents. Cooperative communication between agents about their existing services, their past services requests and their performance will enable service requesters to locate the service with the most suitable QoS. An ontology-based approach described here will complement existing service repositories, which will provide PSM implementations that may be used in both the design and implementation phases (Figure 7).

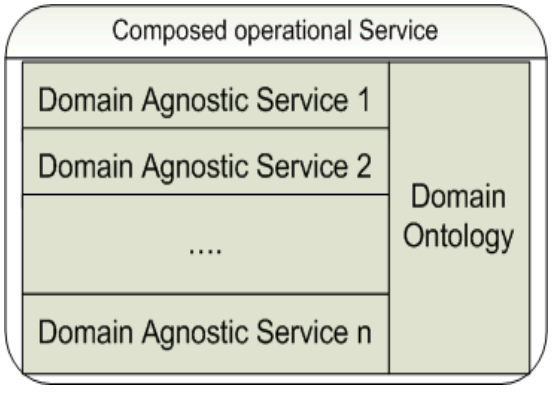


Figure 7. Ontologies can be used to give a dynamic interface to services to agents within a MAS.

When an agent receives a service request that it cannot fulfil, it seeks out a service from another agent or repository of services. This may happen as follows:

1. Identify the corresponding domain of the request
2. Use the domain knowledge to map to the service interface in order to index the PSM corresponding to the service requested.
3. Map its domain knowledge to the individual PSM tasks and perform the tasks to fulfil the service request.

For example, suppose that an agent is interested in engaging in a specific negotiation with another opponent agent. Assuming it is aware of the negotiation protocol, with limited domain knowledge and information about its opponent's preferences, it needs a method to model the opponent and a method to devise a strategy to act. By mapping its domain knowledge to the PSM library, it identifies and employs a suitable coded implementation for a model and strategy. As the negotiation commences, the agent feeds information to the PSM model interface, the model updates, the agent feeds the output of the model (along with the negotiation protocol) to the strategy interface, and follows the recommended course of action. The agent has no fixed automated negotiation approach but, rather, has the capacity to dynamically select the approaches that best suit its circumstance.

In this P2P service evaluation and exchange application, ontologies at various levels of abstractions and details have been developed. This offers a unique workbench to test the reuse of ontologies and places them at the centre of the SDLC. For instance, OWL-S is an ontology to describe Web services with rich semantics. It will allow individual software agents to discover, invoke, compose and monitor Web services with a high degree of automation under dynamic circumstances. The use of this ontology has also been delineated to easily identify problem solving methods of individual agents, bypassing problems identified in (Beydoun et al, 2006a). In fact, OWL-S (OWL-S Coalition, 2006) ontology consists of three main components: the services profile, the process model and the grounding. The services profile is for advertising and discovering Web services. The process model is used to describe detailed operations of services and define composite Web services. The grounding is used to map the abstract definition of services to concrete specifications of how to access the services.

The services profile component of the ontology (corresponding to Task/PSM ontology in Section 4) can be detailed and refined to allow detailed services' description and evaluation. Basically, the service profile does not mandate any representation of services; rather, using the OWL subclass it is possible to create specialised representations of services that can be used as service profiles. OWL-S provides one possible representation through the class "Profile". An OWL-S "Profile" describes services individually as a combination of three basic types of information: what organisation provides each service, what functions each service computes, and a host of features that specify characteristics of each service. In this way, the complementary descriptions about Web services

[4] As a reviewer noted, existing methodologies for creating PSMs are often inadequate. In this example, this problem is by-passed as services do exist and they are typically used to describe atomic tasks within a business process.

including the QoS can be extended in the services profile, so that we can improve the automation and reliability of Web services' composition in dynamic circumstance.

QoS is an important criterion for e-service selection in dynamic environment. In general, QoS refers to the capability of a network to provide better service to selected network traffic over various technologies. As for P2P-based network, the dynamic and unpredictable nature in e-service processes always affects the service's composition and performance significantly. In addition, the dynamic e-business vision calls for a seamless integration of business processes, applications, and e-services over the Web space and time. In other words, QoS properties such as reliability and availability for an e-service process are in high demand. Furthermore, changes and delay in traffic patterns, denial-of-service attacks and the effects of infrastructure failures, low performance in executions, and other quality issues over the Web are creating QoS complications in a P2P network. Quite often, unresolved QoS issues cause critical transactional applications to suffer from unacceptable performance degradation. Consequently, there is a need to distinguish e-services using a set of well-defined QoS criteria.

With the large number of e-services, consumers definitely would like to require a means to distinguish between 'good' and 'bad' service providers. In such a case, QoS is the means to select a 'better' e-service among various providers. From another aspect, the different collaborating e-services applications will compete for network resources in an unreasonable and uncontrollable manner if their interactions are not coordinated by any agreements or specification on QoS differentiation. Naturally, these factors will force service providers to understand and achieve QoS-aware services to meet the demands. Also, a better QoS specification for e-service will become more significant by being a unique selling point for a service provider. Fundamentally, the Web services QoS requirement refers to the quality, both functional and non-functional, aspects of an e-service. This includes performance, reliability, integrity, accessibility, availability, interoperability, and security (Mani and Nagarajan, 2002). The properties become even more complex when adding transactional features to e-services.

How to properly design and integrate QoS criteria in P2P-based e-service process is an important innovation for e-business development in decentralised network. It particularly lends itself to ontology based development, as services correspond to tasks that can be indexed using a task ontology. In a dynamic environment, higher level ontologies (application and domain ontologies) can be used by agents to locate appropriate providers of services and undertaking dynamic evaluation through appropriate communication between agents. (Greco et al., 2004) present an ontology-driven framework to build complex process models that can be reused in this application. More specifically, a web services modelling ontology is described in detail in (Roman et al., 2005) and a "Generic Negotiation Ontology (GNO)" in (Ermolayev and Keberle, 2006) as an upper level negotiation ontology for software agents. All these can be reused in this application.

6 Summary and Conclusions

This paper promotes ontology-based software development with a focus on methodologies for MAS development. The paper first provides a conceptual analysis bridging software engineering concepts (models, modelling, metamodels etc.) and existing ontology research emanating largely from the knowledge engineering community. This provides a grounded position on what an ontology can do to the SDLC and to launch a methodological sketch of an ontology-based multi agent system methodology. Key concepts and roles of an ontology in a SDLC are illustrated in an application, which is amenable to both the deployment of agents and ontologies. Whilst this is a preliminary illustration, it does clearly argue for enhanced reuse by using ontologies as a central software workproduct.

Much work remains to refine the concepts presented in this and to ensure that they are applicable to areas where the use of ontologies is less obvious than the domain discussed in this paper. Towards this, the first step is to develop required ontological techniques. These include ontology-based techniques for consistency checking across products and processes, and ontology-based techniques for testing completeness of products and processes within and across methodologies. Underlying complex issues need to be resolved, e.g. as how to reconcile requirements from multiple sources and multiple versions of ontologies. Another issue is how do candidate Problem Solving Methods get identified to be reused. Moreover, if new Problem Solving Methods are needed for the system and if creating these is too cumbersome, then this could certainly lead to the ontology-based approach to be abandoned (as one reviewer pointed out). It may well turn out that an ontology-based approach is most suited to areas of applications where the set of possible agent actions are well specified in advance e.g. in modelling service oriented systems.

Acknowledgement

This research is supported by the Australian Research Council. This is Contribution number 09/07 of the Centre for Object Technology Applications and Research.

References

Anquetil, N., de Oliveira, K.M. and Dias, M.G.B. (2006): Software maintenance ontology. In *Ontologies for Software Engineering and Software Technology*. 153-173. Calero, C., Ruiz, F. And Piattini M. (eds). Springer Berlin Heidelberg.

ANSI (1989): *Information Resource Dictionary System* X3.138, American National Standards Institute. New York.

Atkinson, C., Gutheil, M. and Kiko, K. (2006): On the relationship of ontologies and models. *Proc. WoMM Workshop on Meta-Modelling,* Karlsruhe, Germany,

2:47-60, Springer. Lecture Notes in Informatics (LNI) 96 GI 2006.

Bernon, C., Cossentino, M., Gleizes, M., Turci, P. and Zambonelli, F. (2004): A study of some multi-agent meta-models. *Proc. AOSE 2004 International Workshop on Agent-Oriented Software Engineering*, New York, USA, 5:62-77, Springer Berlin/Heidelberg. Lecture Notes in Computer Science (LNCS) 3382.

Bertoa, M.F., Vallecillo, A. and García, F. (2006): An ontology for software measurement, In *Ontologies for Software Engineering and Software Technology*. 175-196. Calero, C., Ruiz, F. And Piattini M. (eds). Springer Berlin Heidelberg.

Beydoun, G., Tran, N., Low, G.C. and Henderson-Sellers, B. (2006a): Foundations of ontology-based MAS methodologies. *Proc. AOIS 2005 International Bi-Conference Woekshop*, Utrecht, Netherland, **7**:111-123, Springer Berlin/Heidelberg. Lecture Notes in Artificial Intelligence (LNAI) 3529.

Beydoun, G., González-Pérez, C., Henderson-Sellers, B. and Low, G.C. (2006b): Developing and evaluating a generic metamodel for MAS work products. In *Software Engineering for Multi-Agent Systems IV: Research Issues and Practical Applications*, 126-142. Garcia, A., Choren, R., Lucena, C., Giorgini, P., Holvoet, T. And Romanovsky, A. (eds). Springer Berlin/Heidelberg. Lecture Notes in Computer Science (LNCS) 3914.

Beydoun G., Low G.C., Henderson-Sellers B., Mouratidis H., Gomez-Sanz J.J., Pavón J. and González-Pérez C. (2009): FAML: A generic metamodel for MAS development, *IEEE Transaction on Software Engineering*, **35** (forthcoming).

Brinkkemper, S., Saeki, M. & Harmsen, F. (2001): A method engineering language for the description of systems development methods. *Proc. CAiSE'01 International Conference on Advanced Information Systems Engineering*, Interlaken, Switzerland, **13**:473-476, Springer Berlin/Heidelberg. Lecture Notes in Computer Science (LNCS) 2068.

Calero, C., Ruiz, F. and Piattini, M. (eds) (2006): *Ontologies for Software Engineering and Software Technology*, Berlin Heidelberg, Springer-Verlag.

Corcho, O., Fernández-López, M. and Gómez-Pérez, A. (2006): Ontological engineering: principles, methods, tools and languages. In *Ontologies for Software Engineering and Software Technology*. 1-48. Calero, C. Ruiz, F. And Piattini, M. (eds). Springer Berlin Heidelberg.

DARPA (2000): Ontology Repository, http://www.daml.org/ontologies/. Accessed 1 July 2009.

DiLeo, J., Jacobs, T. and DeLoach, S. (2002): Integrating ontologies into multiagnet systems engineering. *Proc. AOIS 2002 at CAiSE'02 International Bi-Conference Workshop on Agent-Oriented Information Systems*, Toronto, Ontario, Canada, **4**:1-15.

Ermolayev, V. and Keberle, N. (2006): A generic ontology of rational negotiation. *Proc. ISTA 2006 International Conference on Information Systems Technology and its Applications*, Klagenfurt, Austria, **5**:51-65. Lecture Notes in Informatics (LNI) 84 GI 2006.

Esteva, M., de la Cruz, D. and Sierra, C. (2002): ISLANDER: an electronic institutions editor. *Proc. AAMAS 2002 International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy, **1**: 1045-1052, ACM Press.

Falbo, R.A., Ruy, F.B. and Moro, R.D. (2005): Using ontologies to add semantics to a software engineering environment. *Proc. SEKE 2005 International Conference on Software Engineering and Knowledge Engineering*, Taipei Taiwan, Republic of China, **17**:151-156.

Favre, J.-M., Gašević, D., Lämmel, R. and Winter, A. (eds), (2007): 3rd International Workshop on Metamodels, Schemas, Grammars and Ontologies. In *Models in Software Engineering*, 52-55. Kühne, T (ed). Springer Berlin/Heidelberg.

Fensel, D. (2004): *Ontologies: a silver bullet for knowledge management and electronic commerce*, second edition, Berlin Heidelberg, Springer-Verlag.

Fensel, D. (1997): The tower-of-adaptor method for developing and reusing problem-solving methods. *Proc. EKAW European Workshop on Knowledge Acquisition, Modeling and Management*, Sant Feliu de Guíxols, Catalonia, Spain, **10**: 97-112. Springer Berlin/Heidelberg. Lecture Notes in Artificial Intelligence (LNAI) 1319.

Girardi, R. and Serra, I. (2004): Using ontologies for the specification of domain-specific languages in multi-agent domain engineering. *Proc. AOIS 2004 at CAiSE'04 International Bi-Conference Workshop on Agent-Oriented Information Systems*, Riga, Latvia, **6**: 295-308, Faculty of Computer Science and Information Technology, Riga Technical University.

Giunchiglia, F., Mylopoulos, J. and Perini, A. (2003): The tropos software development methodology: processes, models and diagrams. In *Agent-Oriented Software Engineering III*. 162-173. Giunchiglia, F., Odell, J. And Weiß, Gerhard. (eds). Springer-Verlag.

González-Pérez, C. and Henderson-Sellers, B. (2006): A powertype-based metamodelling framework, *Software and Systems Modeling*, **5**(1), 72-90.

González-Pérez, C. and Henderson-Sellers, B. (2008): *Metamodelling for software engineering*, Chichester, UK, John Wiley & Sons.

Greco, G., Guzzo, A., Pontieri, L. and Saccà, D. (2004) An ontology-driven process modeling framework. *Proc. DEXA International Conference on Database and Expert Systems Applications*, Zaragoza, Spain, 15:13-23, Springer Berlin/Heidelberg. Lecture Notes in Computer Science (LNCS) 3180.

Green, P. and Rosemann, M. (2005): *Business systems analysis with ontologies*, Hershey, PA, USA, IGI Publishing.

Gruber T.R. (1993): A translation approach to portable ontology specifications. *Knowledge Acquisition*, **5**(2): 199-220.

Guarino, N. (1998): Formal ontology and information systems. *Proc. FOIS 2008 International Conference on Formal Ontology in Information Systems*, Trento, Italy, **1**:3-15, IOS Press.

Guizzardi, G. (2005). Ontological Foundations for Structural Conceptual Models. Ph.D. thesis. Centre for Telematics and Information Technology. Enschede, The Netherlands. CTIT PhD Thesis Services No. 05-74. Telematics Instituut Fundamental Research Series No. 015 (TI/FRS/015).

Guizzardi, G. and Wagner, G. (2005a): Towards ontological foundations for agent modelling concepts using the unified foundational ontology (UFO). *Proc. AOIS 2004 International Bi-Conference Workshop*, Riga, Latvia, **6**: 110-124, Springer Berlin/Heidelberg. Lecture Notes in Computer Science (LNCS) 3508.

Guizzardi, G. and Wagner, G. (2005b): Some applications of a unified foundational ontology in business modeling. In *Business Systems Analysis with Ontologies*. 345-367, Green, P. And Posemann, M. (eds). IGI Publishing.

Henderson-Sellers, B. (2003): Method engineering for OO systems development. *Communications of the ACM* **46** (10): 73-78.

Henderson-Sellers, B. and Giorgini, P. (eds) (2005): Agent-Oriented Methodologies. Hershey, USA, IGI Publishing.

Henderson-Sellers, B. (2006): Method engineering: theory and practice. *Proc. ISTA 2006 International Conference on Information Systems Technology and its Applications*, Klagenfurt, Austria, **5**:13-23. Lecture Notes in Informatics (LNI) 84 GI 2006.

Henderson-Sellers, B. and Unhelkar, B. (2000): *OPEN modeling with UML*, Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc.

ISO/IEC (2007): 24744: *Software Engineering - Metamodel for Development Methodologies*, Geneva, Switzerland, International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC).

Karagiannis, D., Fill, H.-G., Höfferer, P. and Nemetz, M. (2008): Metamodelling: some application areas in information systems. Proc. UNISCON 2008 Information Systems and e-Business Technologies, International United Information Systems Conference, Klagenfurt, Austria, **2**:175-188. Springer-Verlag Berlin Heidelberg. Lecture Notes in Business Information Processing (LNBIP) 5.

van Lamsweerde, A., Darimont, R. and Letier, E. (1998): Managing conflict in goal-driven requirements engineering. *IEEE Transactions on Software Engineering* **24**(11):908-926.

Leppänen, M. (2007): An Ontological framework of method engineering: an overall structure. *Proc. EMMSAD 2007 Workshop on Exploring Modeling Methods for Systems Analysis and Design at CAiSE'07 Conference on Advanced Information Systems,* Trondheim, Norway, **12**:47-57, Tapir Academic Press.

Mani, A. and Nagarajan, A. (2002): Understanding Qality of Service for Web Services, http://www.ibm.com/developerworks/library/ws-quality.html. Accessed 1 June 2008.

Noy, N.F. and McGuinness, D.L. (2001): *Ontology development 101: a guide to creating your first ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.

Oliveira, K.M. de, Villela, K., Regina Rocha, A. and Horta Travassos, G. (2006): Use of ontologies in software development environments. In *Ontologies for Software Engineering and Software Technology.* 275-309. Coral, C., Ruiz, F. And Piattini, M. (eds). Springer Berlin Heidelberg.

OMG (2005a): *Unified Modeling Language: Superstructure*, Version 2.0, formal/05-07-04. http://www.omg.org/spec/UML/2.0/Superstructure/PDF.

OMG (2005b): *Ontology Definition Metamodel*, ad/2005-08-01.

OWL-S Coalition (2006): Semantic Markup for Web Services. http:// www.daml.org/services/owl-s/. Accessed 1 Nov 2009.

Rilling, J., Zhang, Y., Meng, W.J., Witte, R., Haarslev, V. and Charland, P. (2007): A unified ontology-based process model for software maintenance and comprehension. In *Models in Software Engineering*. 56-65. Kűhne, T. (ed). Springer Berlin/Heidelberg. Lecture Notes in Computer Science (LNCS) 4364.

Rodríguez, A., Palma, J. and Quintana, F. (2003): Experiences in Reusing Problem Solving Methods - An Application in Constraint Programming. *Proc. KES'2003 International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, Oxford, U.K., **7**:1299-1306. Lecture Notes in Computer Science (LNCS) 2774.

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C. and Fensel, D. (2005): Web Service Modeling Ontology, *Applied Ontology* **1**(1): 77-106.

Ruiz, F. and Hilera, J.R. (2006): Using ontologies in software engineering and technology. In *Ontologies for Software Engineering and Software Technology*. 49-102. Calero, C., Ruiz, F. And Piattini M. (eds). Springer Berlin Heidelberg.

Shen J., Yang Y., Yan J. (2007): A P2P based Service flow system with advanced ontology-based service profiles. *Advanced Engineering Informatics* **21**(2)**:**221-229.

Sousa, J.P.P., Carrapatoso, E., Fonseca, B., Pimentel, M.G.C. and Bulcão-Neto, R.F. (2009): Composition of Context-Aware Mobile Services Using a Semantic Context Model, *IARIA International Journal on Advances in Software* **2**(2):1-13.

Tran, Q.N.N. and Low, G.C. (2005): Comparison of Methodologies. In Agent-Oriented Methodologies. 341-367. Henderson-Sellers, B. And Giorqini, P. (eds). IGI Publishing.

Tran, Q.N.N., Low, G.C. and Beydoun, G. (2006): A methodological framework for ontology centric oriented software engineering. International Journal of Computer Systems Science and Engineering **21**(2):117-132.

Uschold, M. (2005): An ontology research pipeline. *Applied Ontology* **1**(1):13-16.

Wand, Y. and Weber, R. (2005): Introduction: setting the scene. In *Business Systems Analysis with Ontologies*. xii-xv, Green, P. And Posemann, M. (eds). IGI Publishing.

Wooldridge, M., Jennings, N.R. and Kinny, D. (2000): The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems* **3**(3):285-312.

An Approach to Customizing Requirements Goal Model based on Metamodel for Ontology Registration

Chong Wang[1], Chi Zhang[2], Keqing He[1], Jingbai Tian[3], Jian Wang[1], Cheng Zeng[1]

[1]State Key Lab. Of Software Engineering, Wuhan University, Wuhan, 430072, China

[2] International School of Software, Wuhan University, Wuhan, 430079, China

[3] School of Computer Science, Hubei University of Technology, Wuhan, 430068, China

cwang@sklse.org, chzhcn88@gmail.com

Abstract

To speed up creation of personalized requirements models for users, it is necessary to systematically model domain knowledge and specify a reuse mechanism to customize personalized requirements models. Based on a unified framework for requirements metamodeling named RGPS (Role-Goal-Process-Service), a goal oriented and ontology based approach is proposed in this paper to customization of requirements goal models based on domain knowledge. Particularly, Metamodel for ontology registration is introduced as a common facility to register goal models and promote semantic interoperation between them. Accordingly, a series of rules are designed to construct, refine and ultimately confirm requirement goal models. In order to demonstrate how our approach works, a case study in urban transportation domain is illustrated step by step to provide details of how to customize requirements goal models for users. In this way, well-modeled and registered domain goal models will be the foundation for constructing high-quality requirements goal models in a normative way.

Keywords: Domain modeling, requirements customization, goal-oriented requirements analysis, ontology

1 Introduction

With the rapid growth of Internet, the software development environment is shifting from centralized and closed local network to open, dynamic, complex, and evolving Internet (Fuqing 2005), software development is now facing challenges of providing better products and services to discriminating customers by reusing existing information resources，which leaves two key problems to be resolve. One is to promote interoperation between heterogeneous resources; the other is to fill in the gap between users' requirements described from different viewpoints and those resources.

Goals are often used as descriptive statements of users' intention, or objectives the system under consideration should achieve (Lamsweerde 2001). Thus, goal models satisfying specific requirements are capable of carrying users' intention to describe functional and non-functional requirements at different levels of granularity. As an active branch of requirements engineering, famous goal-oriented approaches to requirements description and analysis, such as KAOS (Lamsweerde 2001, Dardenne 2003) and i*(Yu 1997), can a) characterize and classify requirements that are viewed as goals, and help developers capture real motivation and intention of users in an accurate and precise way; b) generate operational goals for developers by decomposing and refining abstract goals. Goal-oriented approaches enable a smooth transition from users's abstract descriptions to specified softwares, but rare effective mechanisms for non-functional requirements have been issued. Tropos (Castro 2002) and NFR (Non-Functional Requirements Framework) (Chung 2000, Mylopoulos 1992) define "softgoal" to express nonfunctional requirements. Yet, it differs from non-functional goals in that it can express both nonfunctional requirements and extra functional expectations, which brings difficulty for technicians to analyse and process requirements described by softgoals. On the other hand, ontology is able to capture the semantics of information from various sources and give them a concise, uniform and declarative description, therefore have brought up significant attention in academia and industry (Fensel 2001). So it offers a common semantic foundation to support consistent expression of both user's requirements and information resources.

To perform requirements modeling in a comprehensive and user-friendly way, a framework for requirements metamodeling named RGPS was proposed (Jian 2007,

Jian 2008). It supports requirements modeling from four aspects, namely, role, goal, process and service, so that interweaved requirements in a specific domain can be organized into orderly and structured requirements specifications. Compared with the requirements modeling methods motioned above, Goal metamodel in RGPS is a suitable choice for goal-oriented requirements modeling because a) it makes a clear distinction between functional goals, nonfunctional goals and operational goals as well as defines respective description facilities for them; b) in RGPS, four decomposition manners and two constraints are designed to help goal modeling more flexible for users to perform goal decomposition; c) it specifies relationships among goal, role and process respectively to facilitate reuse of domain knowledge of multi-granularity. In this paper, we will take goal metamodel in RGPS as the foundation and the cases in urban transportation domain as an example to demonstrate how to customize requirements goal model(RGM) from RGPS-based domain goal models (DGM). In RGPS, DGM is described with OWL. So Metamodel for Ontology Registration (MOR) is introduced in this paper, which can register and manage goal models in a specific domain. On one hand, MOR is able to promote interoperation between heterogeneous domain assets. On the other hand, customization of RGM will be addressed as how to create local ontologies based on a given reference ontology. Meanwhile, in terms of internal relationships between local ontology and the reference ontology in MOR, which part of a DGM is reused and how frequently it is reused can be summarized to benefit the process of merging individual requirements into requirements of a specific user group.

The rest of this paper is organized as follows: section 2 gives a brief introduction of MOR; section 3 explains the details of the Goal metamodel in RGPS and shows how to perform goal modeling in the urban transportation domain with the modeling tool we developed; section 4 proposes a solution to customize requirements goal model based on domain goal model created in section 3 and its registration information based on MOR; section 5 is the related work, followed by the summary and future work.

2 Brief Introduction of MOR

MOR is a key member of Metamodel Framework for Interoperability (ISO/IEC 19763) (ISO 2007), whose main objective is to register and manage administrative information with respect to the structure and semantics of ontologies. Since the differences in ontology descriptive languages and ontology development techniques add difficulties in promoting semantic interoperations between ontologies, MOR illustrates a comprehensive solution for this problem (Yangfan 2005, Chong 2006). The overall structure of MOR is depicted in Fig. 1.

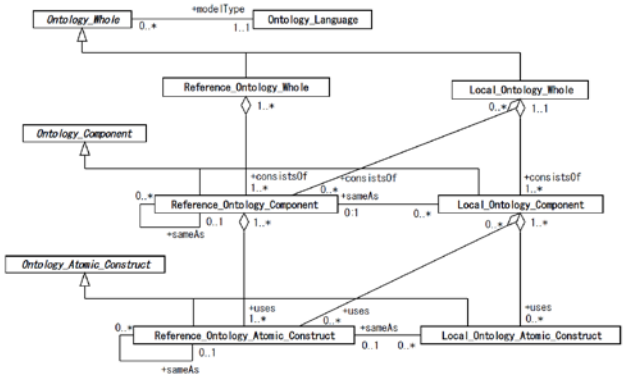


Fig.1. Overall structure of MOR

Examining Fig. 1 from the top down, MOR defines "*Ontology_Whole-Ontology_Component-Ontology_Atomic_Construct*" to register common information of ontologies. This three-layer structure implies that an ontology consists of ontology components and each ontology component is composed of ontology atomic constructs, the smallest component of an ontology. Moreover, it only emphasizes the language-independent information of ontologies and ignores their differences caused by representative notations. So for any ontology to be registered, ontology components and ontology atomic constructs will respectively represent sentences and non-logical symbols (such as concepts, instances) of the ontology.

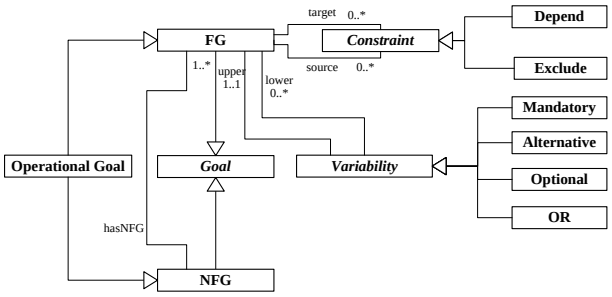
Viewing Fig. 1 from left to right, MOR also specifies ontologies of two different types, *Reference_Ontology_Whole (RO)* and *Local_Ontology_Whole (LO)* to distinguish different roles that ontology plays in different cases. *RO* is responsible of representing common ontologies in domains, which is in usual created and maintained by authorities and/or relevant domain experts to guarantee its suitability. Different from *RO*, *LO* is designed for particular information systems, which reuses some elements of *RO*s and adopts changes to meet different needs. As a result, two *LO*s derived from the same *RO* suggest there is some inherent semantic relationship between them. Thus, the information systems adopting these two *LO*s can interoperate with each other on the basis of the parent *RO*. In addition, *LO* can also reuse some parts of *RO*, modify the reused part and add

new elements in MOR. This idea is illustrated where *Reference_Ontology_Component(ROC)* forms *RO* but *LO* is composed of two ontology components, i.e. *ROC* that comes from *RO* directly and Local_Ontology_Component*(LOC)* that is special for LO and generated by modifying or adding operations. Likewise, *Reference_Ontology_Atomic_Construct(ROAC)* forms *ROC* while *LOC* consists of both *Local_Ontology_Atomic_Construct(LOAC) and ROAC.*

3 Domain Goal Modeling based on RGPS

3.1 Goal Metamodel in RGPS

RGPS is designed for service-oriented requirements metamodeling. It consists of four metamodels which are interconnected with each other. Role Metamodel describes organizations, roles and the interactions between them in a given requirements problem space. Goal Metamodel is to perform goal decomposition by specifying constraints between them. Process Metamodel defines basic constructs of a Process and the connections between them. Service Metamodel provides available services and is in general bound zero-to-many with a process.



NOTE Metaclasses whose names are italicized are abstract metaclasses.

Fig. 2. Overall Structure of Goal Metamodel.

Fig.2 shows Goal Metamodel in RGPS. Considering the system functionality that should be achieved and the global constraints that have to be followed, goals can be classified as *Functional Goal(FG)* and *Nonfunctional Goal(NFG)*. *FG* describes the functions that a system must achieve, and *NFG* explains how these *FG*s are exercised and will affect or restrict the achievement of which *FG* to some degree.

As mentioned before, a goal is a high-level and general statement. With Goal Metamodel in RGPS, it can be refined as a concrete and operational description of the software-to-be. Goal refinement is a process in which a high-level goal is decomposed into sub-goals, using feature decomposition strategies in FODA (Kang 1990). In RGPS, *Variability* decomposition indicates whether a goal is variable with respect to its upper-goal during the process of goal refinement. In detail, *Variability* decomposition relationships that characterize the relationship between the upper goal and lower goal set can be divided into *Mandatory*, *Optional*, *OR* and *Alternative* sub-goals during goal decomposition. A *Mandatory* goal is a goal that is common to all the software systems in domain, while the other three are dependent on particular systems. *Optional* goals are those whose existence depends on the requirements of individual cases. The difference between the *OR* goal and the *Alternative* goal is that exactly one *Alternative* goal can be chosen from a sub-goal set, while more than one *OR* goals can be selected from the set. Additionally, the *Constraint* among goals is either *Depend* or *Exclude*. The former means that the realization of a certain goal depends on the realization of other ones, and the latter implies that two goals cannot be satisfied simultaneously.

3.2 Goal Modeling in Urban Transportation Domain

In this paper, we will take urban transportation domain as the typical application domain for modeling DGMs based on RGPS. To facilitate domain modeling based on RGPS, a domain modeling toolkit named O-RGPS was developed to import domain ontologies of a specific domain and perform domain knowledge modeling based on RGPS. Supposing we want to arrange a trip plan, a corresponding domain goal model will be created by the goal modeling tool in the toolkit, shown in Fig.3.

We can find that FG “PrepareTripPlan” is composed of four mandatory FGs, i.e. “GenerateTripPreference”, “QueryTripInfo”, “ArrangeTrip” and “DisplayTripPlan” and one optional goal “PerformBooking”. Moreover, “DisplayTripPlan” depends on “ArrangeTrip”, which depends on “QueryTripInfo”. FG “PerformBooking” has a NFG “AvailabilityisGreaterThan95”. “QueryBusInfo”, “QueryHotelInfo” and “QueryParkingInfo” are three OR sub-goals of “QueryTripInfo”. “DisplayTripPlan” also has two *OR* subgoals, i.e. “DisplayTripbyVideo” and “DisplayTripbySMS”. In addition, operational goal “QueryBusInfobyStation” and “QueryBusInfobyRoute” are two alternative subgoals of “QueryBusInfo”.

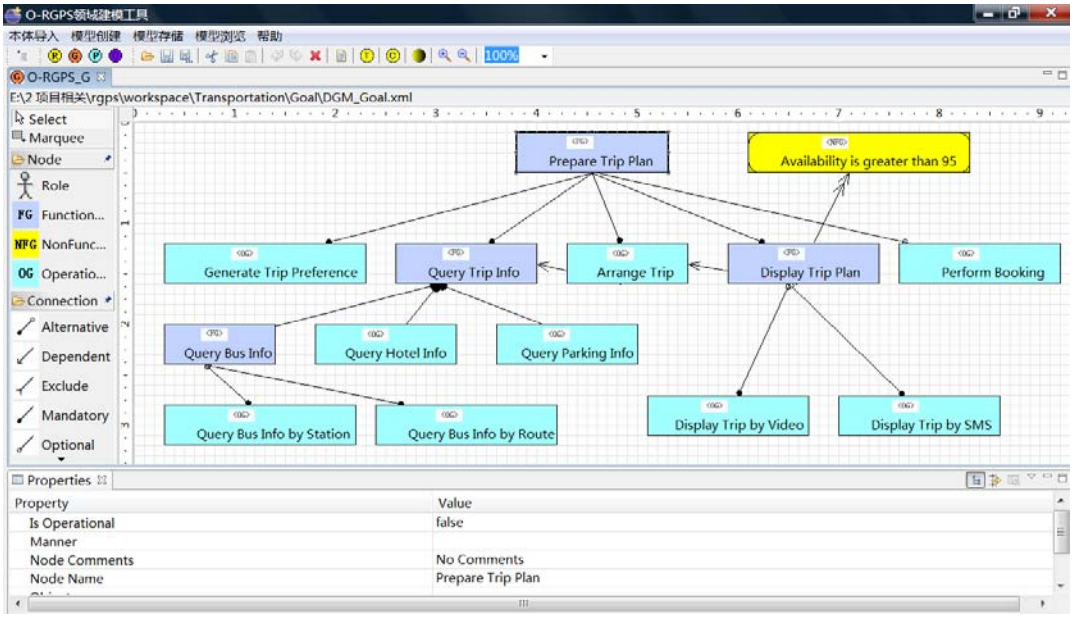


Fig. 3 Screenshot of the domain goal model for "Prepare Trip Plan".

4 Customizing Personalized RGM

The basic idea of RGM customization is to reuse common goals from RGPS-based domain goal models and add personal goals for user preference. The process of customizing RGM includes the following steps:

Step1: register DGM as an instance of RO according to the three-layer structure defined in MOR.

Step2: select a requested set of goals from registered DGM, add relevant goals to the set and supplement associations between them, and then organize them as a predefined RGM (PreRGM) that is in the form of DGM.

Step3: simplify the associations between goals in PreRGM and convert it into RGM.

Step4: complete RGM by adding personalized goals and detecting potential conflicts between goals, register RGM as LO and specify "sameAs" relation from RGM to DGM.

In this way, a unified registration facility is introduced to promote sharing of domain assets and interoperation between them by registering and managing both common and personalized requirements in a domain. Furthermore, "sameAs" relations between RGM and DGM states which goals are frequently reused by which users. This can help requirements evolution from individuals to user group.

4.1 Goal Metamodel in RGPS

Before registering DGM based on MOR, we will introduce graph theory, a formal description of DGM, in this section.

Def.. $DGM = \langle V, E \rangle$ denotes a domain goal model, in which V denotes a set of goals and E denotes relationships between them, where $E = V \times V$ and $\forall e \in E, \exists u, v \in V \rightarrow e = \langle u, v \rangle \cap w(e) = \{M, O, A, OR, D, Ex, hasNFG\}$. M, O, A, OR, D and Ex denote *Mandatory*, *Optional*, *Alternative*, *OR*, *Depend* and *Exclude* respectively. The domain goal model in Fig.2, for example, can be formalized into followings:

$DGM_{prepareTrip} = \langle V_{prep\ areTrip}, E_{prepareTrip} \rangle$

$V_{prepareTrip}$

$= \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}\}$

$=$ {PrepareTripPlan, GenerateTripPrefernce, QueryTripPlan, ArrangeTrip, DisplayTripPlan, PerformBooking, QueryBusInfo, QueryhotelInfo, QueryParkingInfo, QueryBusInfobyStation, QueryBusInfobyRoute, DisplayTripbyVideo, DisplayTripbySMS, AvaliablityisGreaterThan95}

$E_{prepareTrip}$

$= \{\langle v_1, v_2 \rangle, \langle v_1, v_3 \rangle, \langle v_1, v_4 \rangle, \langle v_1, v_5 \rangle, \langle v_1, v_6 \rangle, \langle v_4, v_3 \rangle, \langle v_5, v_4 \rangle, \langle v_5, v_{14} \rangle, \langle v_3, v_7 \rangle, \langle v_3, v_8 \rangle, \langle v_3, v_9 \rangle, \langle v_7, v_{10} \rangle, \langle v_7, v_{11} \rangle, \langle v_5, v_{12} \rangle, \langle v_5, v_{13} \rangle\}$

Take a further look at $E_{prepareTrip}$, it is found that

$w(\langle v_1, v_2 \rangle) = w(\langle v_1, v_3 \rangle) = w(\langle v_1, v_4 \rangle) = w(\langle v_1, v_5 \rangle) = M,$

$w(\langle v_1, v_6 \rangle) = Op,$

$w(\langle v_4, v_3 \rangle) = w(\langle v_5, v_4 \rangle) = D,$

$w(\langle v_3, v_7 \rangle) = w(\langle v_3, v_8 \rangle) = w(\langle v_3, v_9 \rangle) = OR$

$w(\langle v_7, v_{10}\rangle) = w(\langle v_7, v_{11}\rangle) = w(\langle v_5, v_{12}\rangle) =$
$w(\langle v_5, v_{13}\rangle) = A,$

$w(\langle v_5, v_{14}\rangle) = \text{hasNFG}$

After registering DGM, the whole goal model is registered as an instance of RO. Individual goals and the constraints between them in the goal model is viewed as ROAC. The instance of ROC can be any subgraph consisting of some goals and the corresponding relations.

Table 1: Mapping Goal metamodel to MOR.

Elements in Goal metamodel	Metaclasses in MOR
$DGM = \langle V, E\rangle$	RO
$subDGM = \langle V', E'\rangle, V' \subseteq V, E' \subseteq E$	ROC
$\forall v \in V$	ROAC
$\forall e \in E \rightarrow \exists u, v \in V \cap e = \langle u, v\rangle$	ROAC

(a)

(b)

Fig. 4 Registration information of (a) ROC and (b) ROAC based on the Ontology Registration Platform.

We have developed a platform for ontology registration based on MOR. Following mappings from extended Goal metamodel to MOR in Table1, we can get registration information of DGM created in section 4.1. Fig.4(a) shows registration information of ontology component and Fig.4(b) illustrates that of ontology atomic construct.

4.2 Customizing PreRGM

Given a set of goals, V_{user} is the only input when users try to customize a satisfying RGM. It includes V_{common} and $V_{personal}$, denoting part of goals within DGM and a set of personalized goals beyond DGM, respectively. Actually, V_{common} is the starting point of customizing PreRGM. The basic idea is that for $\forall v_i \in V_{common}$, we should find a $subDGM_i = \langle V_i, E_i\rangle$ whose root is v_i and in which 1) V_i represents a set of goals that is directly or indirectly related to v_i and 2) E_i denotes a set of associations connecting goals in V_i. The algorithm below shows how to customize PreRGM.

```
// GoalCollection denotes a set of
selected goals.
// Boolean Goal.hasRelatedGoal(Goal
goal) is used as a method to tell
whether a goal has related goals.
// GoalCollection
Goal.listRelatedGoal() is used to list
all the goals related to a given goal
GoalCollection
completeGoals(GoalCollection gc1){
gc = new GoalCollection ();
  for(int i=0; i<gc1 ; i++){
    gc.add(gc1[i]);
    if(gc1[i].hasrelatedgoal()){
      gc2=gc1[i].listRelatedGoal();
          for(int j=0; j<gc2.length();
j++){
            gc.add(gc2[j]);
    }
  }
}
Return gc;
}
```

For example, we suppose:

V_{user} = {ArrangeTripPlan, DisplayTripPlan, PerformBooking, ReponseTimeisLessThan5sec.}

By matching V_{user} to the registration information of RGM, we can get $V_{common} = \{v_4, v_5, v_6\}$ and $V_{personal}$ = {"ReponseTime is less than 5 sec."}

After applying the improvement algorithm above, goals and relevant associations contained in $subDGM_{arrange}$,

$subDGM_{display}$ and $subDGM_{perform}$ are generated as follows. Note that the process of customizing RGM will not change the weight of the edges.

(1) Search goals that are related to V_{user}

$subDGM_{arrange} = \langle V_{arrange}, E_{arrange} \rangle$

$V_{arrange} = \{v_4, v_3, v_7, v_8, v_9, v_{10}, v_{11}\}$

$E_{arrange}$

$= \{\langle v_4, v_3\rangle, \langle v_3, v_7\rangle, \langle v_3, v_8\rangle, \langle v_3, v_9\rangle, \langle v_7, v_{10}\rangle, \langle v_7, v_{11}\rangle\}$

$subDGM_{display} = \langle V_{display}, E_{display} \rangle$

$V_{displ\ ay} = \{v_5, v_4, v_{12}, v_{13}, v_{14}\}$

$E_{display} = \{\langle v_5, v_4\rangle, \langle v_5, v_{14}\rangle, \langle v_5, v_{12}\rangle, \langle v_5, v_{13}\rangle\}$

$subDGM_{perform} = \langle V_{perform}, E_{perform} \rangle$

$V_{perform} = \{v_6\}$, $E_{perform} = \emptyset$

(2) Search goals linking to V_{user}

$V'_{arrange} = V_{arrange} \cup \{v_1, v_5\}$,

$E'_{arrange} = E_{arrange} \cup \{\langle v_1, v_4\rangle, \langle v_1, v_3\rangle, \langle v_5, v_4\rangle\}$,

$V'_{display} = V_{display} \cup \{v_1\}$, $E'_{display} = E_{arrange} \cup \{\langle v_1, v_5\rangle\}$

$V'_{perform} = V_{perform} \cup \{v_1\}$,

$E'_{perform} = E_{perform} \cup \{\langle v_1, v_6\rangle\}$

(3) Generate PreRGM

$PreRGM = \langle V_{common}, E_{common} \rangle$

$V_{common} = V'_{arrange} \cup V'_{perform} \cup V'_{display}$

$= \{v_4, v_3, v_7, v_8, v_9, v_{10}, v_{11}, v_5, v_{12}, v_{13}, v_{14}, v_6, v_1\}$

$E_{common} = E'_{arrange} \cup E'_{perform} \cup E'_{display}$

$= \{\langle v_4, v_3\rangle, \langle v_3, v_7\rangle, \langle v_3, v_8\rangle, \langle v_3, v_9\rangle, \langle v_7, v_{10}\rangle, \langle v_7, v_{11}\rangle,$
$\langle v_5, v_4\rangle, \langle v_5, v_{14}\rangle, \langle v_5, v_{12}\rangle, \langle v_5, v_{13}\rangle, \langle v_1, v_4\rangle, \langle v_1, v_3\rangle,$
$\langle v_1, v_5\rangle, \langle v_1, v_6\rangle\}$

4.3 Refining PreRGM to RGM

Since all the goals involved in RGM will explicitly describe user's requirements, this paper specifies that only *Mandatory, OR* and *Depend* are allowed in RGM. For this purpose, it should expect users not only to further select appropriate goals from PreRGM, but refine PreRGM into RGM with corresponding transformations on *Optional, Alternative* and *Exclude*l. That is, we need to transform *Optional* and *Alternative* association into *Mandatory* as well as delete *Exclude* relation.

The inference rules of *Optional*, *Alternative* and *Exclude* are fundamental for refining PreRGM into RGM. To define the SWRL-based rules, we suppose that *hasGoal* denotes a goal set whose elements are selected by users; *hasNegationGoal* presents a goal set whose elements are beyond the choice of users; *hasMandatory*, *hasOptional*, *hasAlternative* and *hasExclude* denote four kinds of goal sets whose elements are respectively connected with *Mandatory*, *Optional*, *Alternative* and *Exclude*; *differentFrom* implies that the involved goals are quite different with each other. The rules are defined as follows:

Optional Rule: *hasGoal(?x,?y)* $\wedge$ *hasOptional(?z,?y)* $\rightarrow$ *hasGoal(?x,?z).*If user x has a goal y and y is an optional goal of z, then goal z is the mandatory goal of users.

Alternative Rule: *hasGoal(?x,?y)*$\wedge$*hasAlternative(?y,?z)*$\wedge$*hasAlternative(?y,?a)*$\wedge$*differentFrom(?a,?z)*$\wedge$ *hasGoal (?x,?a)* $\rightarrow$ *hasNegationGoal(?x,?z).* Given user x has a goal y, which has two different alternative goals named x and z. If a is selected by user x, then z must be contained in the negation goal set of x.

Exclude Rule: *hasGoal(?x,?y)* $\wedge$ *hasExclude(?y,?z)* $\rightarrow$ *hasNegationGoal(?x,?z).* Given user x has a goal y. If goal y and z exclude each other, then y resides in the negation gal set of x.

Table 2: Transformation rules for refining PreRGM into RGM.

Rule name	Operation
Change_Op_To_M	e= $\langle u, v\rangle, w(e) = Op$ if $u, v \in V'_{common} \cap e \in E'_{common}$ then $w(e) := M$
Change_A_To_M	$e_{12} = \langle v_1, v_2\rangle, e_{13} = \langle v_1, v_3\rangle, w(e_{12}) =$ if $v_2 \in V'_{common}$ then { $w(e_{12}) = M$; $w(e_{13}) := Ex$;} else if $v_3 \in V'_{common}$ then { $w(e_{13}) = M$; $w(e_{12}) := Ex$; }
Delete_Ex	e= $\langle u, v\rangle, w(e) = Ex$ if $u \in V'_{common}$ then {delete v; if $\forall x, y \in V'_{common}$ then { delete e= $\langle v, x\rangle$; delete x; delete $e = \langle y, v\rangle$; } }

Accordingly, Table 2 illustrates how to handle *Optional*, *Alternative* and *Exclude* to perform transformation from PreRGM to RGM. Before refining RGM, we suppose that goals in V'_{common} are the ultimate choice of users to express their personalized requirements. In our case, for example, the individualized requirements are expressed in $V'_{common} = \{v_1, v_3, v_4, v_5, v_6, v_8, v_9, v_{12}, v_{14}\}$.

In this case, only rules of *Change_Op_To_M* and *Change_A_To_M* will be applied to refine PreRGM. In detail, when Rule *Change_Op_To_M* is implemented, the value of $\langle v_1, v_6 \rangle$ will be changed to M; when Rule *Change_A_To_M* is applied, the value of $\langle v_5, v_{12} \rangle$ will be changed to M, followed by deleting OG "Display Trip Plan by SMS", its relevant goals and the relations between them. In this way, the case in section 4.3 will be transformed into the corresponding RGM in Fig.5.

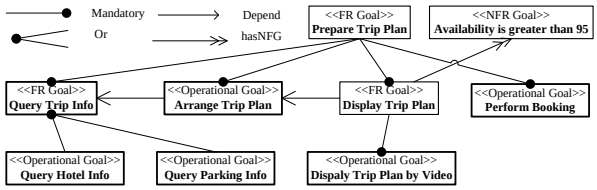
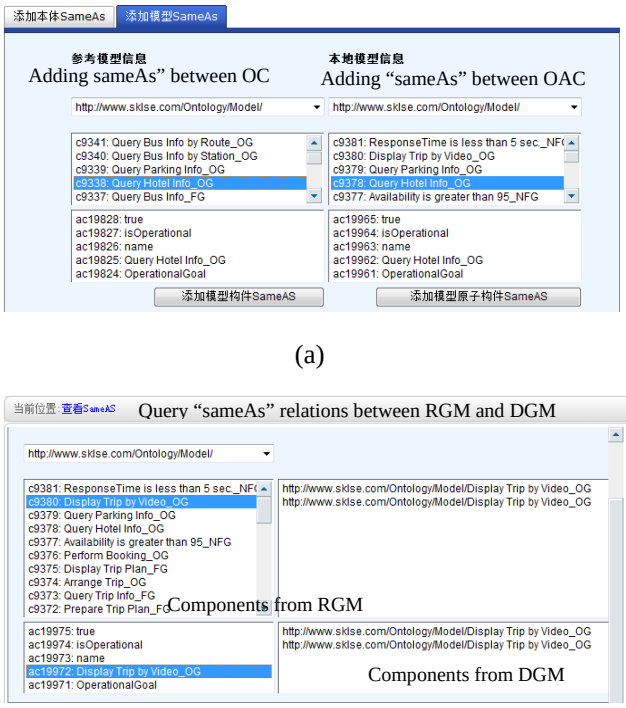


Fig. 5. RGM in urban transportation domain.

4.4 Registering RGM

Till now, only goals matching existing DGM are involved in RGM. In order to completely express individualized requirements of users, goals in $V_{personal}$ will also be added to RGM by means of specifying dependency between goals in $V_{personal}$ and then connecting them to the FGs in RGM. Meanwhile, to keep the consistency of ultimate RGM, it toned to detect potential conflicts between original goals and newly added ones. In this paper, we will focus on conflicts related to nonfunctional requirements, which are further classified into two categories. The first is that Manner of FG and NFR type of NFG are likely to come into conflicts. Take the case in 4.3 as an example, "ResponseTimeisLessThan5sec." is modeled as an instance of NFG beyond the PreRGM. Since information exchange must have a huge transmission volume by means of video, the FG "DisplayTripbyVideo" might hinder realization of "ResponseTimeisLessThan5sec". The second category concentrates on the potential conflicts between NFR types of different NFGs. For instance, "ResponseTime" might be in conflict with "Reliability".

Consequently, it is important to resolve potential conflicts mentioned above to keep a robust goal modeling. As for conflicts between FG and NFG, it is advisable to change the manner of FG or lower expectation of the NFGs. Concerning conflicts between NFGs, we need to lower expectation of some NFGs to leverage the whole quality of experience, which will lead to potential modifications on original goals. Then we should negotiate with users to clarify how to handle conflicts appropriately. In particular, conflicts between FG and NFG should be of top priority.



(a)

(b)

Fig. 6. Screenshot of (a) adding "sameAs" relations to map RGM to DGM and (b) querying added "sameAs" between them.

Similar to the process of registering DGM, ontology registration platform based on MOR supports registration of RGM by importing the corresponding OWL files of RGM. Moreover, "sameAs" relationships defined in MOR to specify how to customize RGM based on DGM should be added manually. For example, Fig.6(a) shows that "QueryHotelInfo" in both DGM and RGM are the same. After that, the newly added "sameAs" relationship becomes visible to query operations. In Fig.6(b), the component "DisplayTripbyVideo" in RGM reuses that of DGM. The operation and query of ontology atomic construct are similar to that of ontology component.

This section just exemplifies how to reuse DGM for customization of RGM and record inherent relationships between RGM and DGM. In this way, the relationships between one DGM and many RGMs can act as the statistics showing how frequently the DGM is reused by which kind of users. That is, it can speed up the process of collecting preference of individuals to merge them into requirements of a certain user group.

5 Related work

Based on software reuse techniques, domain engineering proposes a systematic method for mass reuse of domain knowledge. In general, users' requirements are closely linked with a certain domain where software engineers usually find themselves not familiar with the specified domain knowledge. Thus, the collaboration between knowledge engineering and domain engineering will benefit the process of requirements analysis. Meanwhile, ontologies can be adopted to supply formal description of domain knowledge, which is fundamental for negotiation between users and developers.

Currently, domain-driven requirements engineering is blooming in both academia and industry. In the 1990s, R.Q. Lu and Z. Jin proposed a domain ontology based approach to requirements analysis and modeling for information systems (Ruqian 1995, Ruqian 1996). It performs domain modeling based on ontologies, so that domain ontologis are able to support requirements elicitation, assist the process of requirements modeling and create high-quality requirements model, which is the basis of automatic requirements analysis (Zhi 2000, Ruqian 2000). MADEM (Multi-Agent Domain Engineering Methodology) (Girardi 2007) is designed as a software development methodology for multi-agent requirements modeling. In MADEM, GRAMO (Generic Requirement Analysis Method based on Ontologies) (Girardi 2004) is proposed as an ontology-based approach for requirements modeling and analysis from four aspects, i.e. concept modeling, goal modeling, role modeling and role interaction modeling.

However, most of the approaches to domain-driven requirements modeling mainly concentrate on functional requirements rather than nonfunctional requirements and its correlation with a specified domain. Different domains and scenarios focus on different aspects of nonfunctional requirements. Therefore, domain modeling should include not only functional requirements but its integration with nonfunctional requirements. Our approach not only takes nonfunctional requirements into account, but addresses fusion of FG and NFG by defining two types of NFG-related conflicts and the corresponding solutions for them. In future, those principles used to resolve potential conflicts between FG and NFG as well as between NFGs will be enhanced to ensure customization of consistent and comprehensive requirements for a variety of customers.

6 Summary and future work

In this paper, a goal-oriented approach is proposed to customize personalized RGM based on ontologies by combining RGPS-based domain modeling technique, goal-oriented method for requirements refinement and a common ontology registration mechanism based on MOR as well. Creation of RGM can be implemented by reusing RGPS-based DGM and the registration information based on MOR, while the refinement of initial RGM is processed by the goal model improvement algorithm and checking rules derived from Goal metamodel in RGPS. The ultimate RGM is confirmed by merging FG and NFG requirements. As for MOR, it is responsible for registering RGM and explicitly specifying the relationship between RGM and DGM, so that it can help generate a required RGM based on modeled domain knowledge. And the corresponding registration information will be fundamental for counting the reusing rate of domain knowledge and acting as hints to deduce the requirements of user groups from individual requirements.

In the near future, how to detect and resolve potential conflicts between FG and NFG as well as between NFGs will be the focus of our research. In addition, the platform for ontology registration will be further enhanced by statistically deducing the requirements of a specified user group from the registration information that describes which atomic constructs or components of a DGM are reused by which RGMs.

Acknowledgement

This research project was supported by the National Basic Research Program of China (973) under Grant 2007CB310801, the National High Technology Research and Development Program of China (863) under Grant No.2006AA04Z156, the National Natural Science Foundation of China under Grant No. 60803025, 60873083, 60703018 and 60703009, the Natural Science Foundation of Hubei Province under Grant No.2008ABA379 and 2006ABA228, the Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20070486065, and the Eleventh Five-Year Plan for National Key Technology R&D Program under Grant No. 2006BAK04A20-7.

Reference

Fuqing Yang, Hong Mei. (2005): Internetware: A New Software Modality in the future. China Education Network, Vol.7, (2005)52–54.

D. Fensel, Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce, Springer, 2001.

A. V. Lamsweerde (2001): Goal-oriented Requirements Engineering: a Guided Tour. *Proc. The 5th IEEE International Symposium on Requirements Engineering*, Toronto, Canada, 249-263, IEEE CS.

Dardenne, A. V. Lamsweerde, and S. V. Fickas (1993): Goal-Directed Requirements Acquisition. *Science of Computer Programming* 20:3-50.

E. Yu (1997): Towards Modeling and Reasoning Support for Early Requirements Engineering. *Proc. IEEE International Symposium on Requirement Engineering*. 地点 226 -235.

J. Castro, M. Kolp, and J. Mylopoulos (2002): Towards Requirements-Driven Software Development Methodology: The Tropos Project. *Information Systems*, 27:365-389.

L. Chung, B. A. Nixon, E. Yu, et al. (2000): *Non-Functional Requirements in Software Engineering.* Massachusetts, US, Kluwer Academic Publishers.

J. Mylopoulos, L. Chung, B. Nixon (1992): Representing and Using Non-functional Requirements: A Process-oriented Approach. *IEEE Transactions on Software Engineering*, 18(6): 483-497.

Jian Wang, Keqing He, Bing Li, et al. (2007): Metamodels of Domain Modeling Framework for Networked Software. *Proc. The Sixth International Conderence on Grid and Cooperative Computing. Lecture Notes in Computer Science.* Urumchi. 878-885.

Jian Wang, Keqing He, Ping Gong, et al. (2008): RGPS: A Unified Requirements Meta-Modeling Frame for Networked Software. *Proc. Third International Workshop on Advances and Applications of Problem Frames(IWAAPF'08) at 30th International Conference on Software Engineering(ICSE'08),* Leipaig, Germany, 29-35.

International Organization for Standardization (ISO). ISO/IEC 19763-3 (2007): *Information technology – Framework for metamodel interoperability –Part 3: Metamodel for ontology registration.*

Yangfan He, Keqing He, Chong Wang (2005): Research on Semantic Web Service-Oriented MMFI for Complex Information Registration. *Proc. IEEE International Workshop on Service-Oriented System Engineering (SOSE 2005),* Beijing, 237-243, IEEE Press.

Chong Wang, Keqing He, Yangfan He (2006): MFI4Onto: Towards Ontology Registration on the Semantic Web. *Proc. The sixth International Conference on Computer and Information Technology (CIT 2006).* Seoul, p.4019862, IEEE Press.

Kang, K., Cohen, S., Hess, J., et al. (1990): Feature-Oriented Domain Analysis (FODA): feasibility study. Technical Report: CMU/SEI-90-TR-021, Software Engineering Institute/Carnegie Mellon University.

Ruqian Lu, Zhi Jin, Ronglin Wan (1995): Requirement Specification in Pseudo-Natural Language in PROMIS. *Proc. The 19th International Computer Software and Applications Conference (COMPSAC'95),* Dallas, USA, 96-101, IEEE CS.

Ruqian Lu, Zhi Jin, Ronglin Wan, Youming Xia (1996): An Approach of Acquiring Requirements Information based on Domain Knowledge. *Journal of Software*, 7(3):137-144. (in Chinese)

Zhi Jin (2000): Ontology-based requirements elicitation automatically. *Chinese Journal of Computers*, 23(5):486-492. (in Chinese)

R. Q. Lu, Z. Jin, and G. Chen (2000): Ontology-based Requirements Analysis. *Journal of Software*, 11(8):1009-1017. (in Chinese)

R. Girardi, L. B. Marinho (2007): A Domain Model of Web Recommender Systems based on Usage Mining and Collaborative Filtering. *Requirements Engineering Journal*, 12:23-4

R. Girardi, C. Faria, and L. Marinho (2004): Ontology-based Domain Modeling of Multi-agent Systems. *Proc. The Third International Workshop on Agent-oriented Methodologies at OOPSLA2004*, Canada, 51-62

Using Explicit Semantic Representations for User Programming of Sensor Devices

Kerry Taylor[1] **Patrick Penkala**[2]

CSIRO ICT Centre
GPO Box 664, Canberra, Australia 2601,
[1]Email: Kerry.Taylor@csiro.au
[2]Email: ppenkala@gmail.com

Abstract

As the demand for environmental sensing grows, there is an urgent need to improve access to observation data collected by sensor systems. However, sensing devices and the platforms on which they are deployed are highly heterogeneous in their capabilities as well as in their method for control and for retrieval of observation data. In this paper we propose to employ semantic technologies, in particular descriptions in the OWL-DL ontology language coupled with ontology editors and reasoners to control the heterogeneity.

Through a case study developed for a programmable automatic weather station, we show how an ontological concept description can be translated to an active query or command to a sensor device, coupling interpretation of a declarative ontology with device-specific wrapper code. Our method relies on a single extensible ontological model to describe the capability of sensor devices, and thereby support their discovery, and also to support their programming. This manages the heterogeneity to enable widespread access to sensor programming languages by naive users.

Keywords: Semantic Sensor Networks, Sensor Tasking

1 Introduction

Many of the challenges facing mankind in the 21st Century, such as climate change, biodiversity conservation, food security, water security and sustainable energy require improved data through remote and in-situ environmental sensing services at lower spatial and temporal scales. Currently, sensor devices and their communication networks are both highly heterogeneous and closed: we must find ways to make sensor deployments more widely accessible and re-usable in order to achieve the density of measurement in space and time that is needed.

This issue has been recognised by the Open Geospatial Consortium (OGC), a standards organisation that has been developing a suite of XML-service based standards for Sensor Web Enablement (SWE). However, while the standards specify an interaction protocol and XML markup languages for querying and tasking sensor devices, they do not address the representation of element content, so vital information, for example, the physical property being observed, is generally free text. Recent work in the OGC and the W3C has begun to extend the SWE frameworks by adding ontology-based annotation to the XML content to improve precision and interoperability in this respect (Duchesne et al. 2008).

However, formal ontology languages such as OWL-DL and the emerging OWL 2 DL offer much more than an ability to represent a controlled vocabulary—where the meaning of the terms in the vocabulary is interpreted by human understanding—they offer formal inference available through reasoners for the underlying formal logic. We propose that this mechanism can be used as part of a comprehensive software framework to discover, program and query sensor devices, to retrieve measurements made by sensors; and to integrate the historical and real-time measurement data into broader software systems for analysis and decision support (Li & Taylor 2008).

In this paper, we specifically address the problem of querying and programming sensor devices. Although sensor network architectures vary widely, many sensor platforms permit programmable control over sensor selection, timing and persistent memory management. Querying may be a two-step process of firstly instructing the sensor platform to make the desired measurements, and secondly retrieving the measurements from the sensor platform's memory. In other architectures, sensed data is forwarded to a central Web-connected service from which it may be retrieved through some kind of query. Because of this typical design, unless where specifically stated otherwise in this paper, we do not distinguish between programming, tasking, commanding and querying a sensor network.

Generally, a sensor platform offers a special-purpose programming language for querying and tasking. Although NesC (Gay et al. 2003) is widely used for mote-based Wireless Sensor Networks, there are also many other emerging languages and higher-level programming abstractions are keenly sought, including declarative languages like SNlog (Chu et al. 2007). Typically, commercial sensor platforms employ manufacturer-specific line-based command languages, sometimes intended to be used through a manufacturer-specific client GUI tool. The Environdata Weathermaster Series automatic weather station exemplifies these sensor systems. In our work, unlike the OGC SWE approach for example, we do not attempt to standardise the query interface to sensor systems, but instead to reflect the command and query interface that is offered with a semantic model represented in the OWL-DL language. The semantic model is loaded as configuration data into a client interface tool and thereby a user may work with a common look and feel to discover and interact with multiple sensor networks. Due to the expressive semantic modelling, it is not necessary to constrain the flexibility of the native capability of the sensor systems offered through the common interface, although

it may be desirable to repackage it in order to improve usability. We propose a client interface tool embedded in an architecture for linking multiple heterogeneous sensor platforms. Server-side components support retrieval of shared domain and sensor-specific ontologies, a server-side back end for translation between client-originated queries and sensor network-specific code, and a communication layer for dealing at a low level with heterogeneity in the communication architectures of sensor networks. As a case study, we have implemented our proposal in a prototype that is used for programming or retrieving data from an automatic weather station.

Outline of the paper In the next section we introduce our architecture for ontology-enabled sensor network programming, then we introduce the sensor system that we have used as the basis for the prototype in Section 3. Then we describe the client tool and how it employs description logic classification to assist in programming the sensor network in Section 4. In Section 5 we describe the query translation and communication process, and the retrieval of sensor network data. In Section 6 we discuss related work and we conclude in Section 7 with an analysis of the benefits of our work in a wider context.

2 Architecture Sketch

In Figure 1 we show the structure of the software prototype we have developed. A GUI client tool, the *semantic query client* loads relevant ontologies and permits querying over those ontologies for sensor discovery, analysis and programming. The client tool is built as a plug-in for the well-known editor for OWL DL, Protégé (v3.3.1)[1], and is backed by a description logic reasoner (in our case, Pellet 1.5.1[2]). After processing by the client tool, a query developed by the user is directed to the *ontology transformer* responsible for the intended sensor network. The ontology transformer translates the query to a sensor-device specific form and hands it over to a *communication controller* configured for the sensor device. The device returns a response which is directed back to the ontology transformer for further processing before being returned to the requesting client. In Figure 2 we show the message flow amongst the architectural components during query processing.

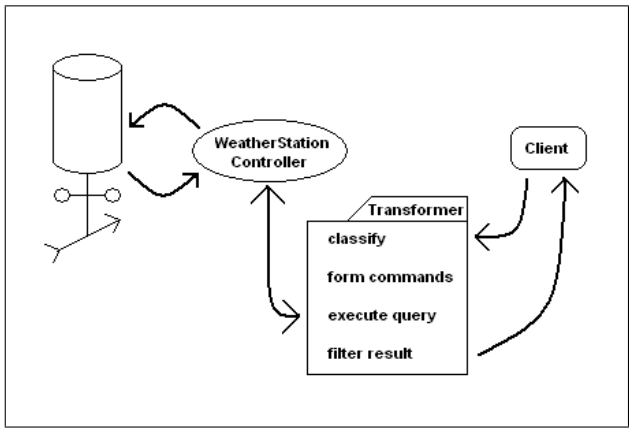


Figure 1: Transformer interacting with the Weather Station and providing functionality to a client

[1]The Protég'é Ontology editor and knowledge acquisition system, http://Protege.stanford.edu/

[2]Pellet OWL DL reasoner, http://clarkparsia.com/pellet/

3 Automatic Weather Station Sensor Platform

In this section we outline the design of the weather station we have used for the case study, concentrating on the command language structure, which will be used to exemplify the presentation of the query processing method later.

Environdata's Weather Station *WeatherMaster 1600*[3] is an industrial self-contained instrument with a built-in battery and attached solar panel. Communication with a host computer uses a proprietary command-line language of about 50 commands in a request-response interaction style over a serial port. It has four sensors which measure air temperature, relative humidity, wind speed, and wind direction, and three other simulated sensors to measure the voltages of the battery and the solar panel and the activity of the serial port.

Measured data can be saved for a limited period in one of four memories, in a FIFO log scheme. The stored data consists of a time stamp and a data value for *each* of the four sensors at the time. The total memory of 104 kilobytes can be split amongst the four memories according to need. The intention behind this splitting support is to avoid logging measurements with the value 0, corresponding to a sensor whose measurement is not recorded. For example, if the Weather Station is programmed to measure the temperature every 30 seconds as well as the relative humidity at 6pm every day, then it will log a data set every 30 seconds, containing the time stamp, the measured value of the air temperature and a 0 for the humidity, since it hasn't been measured yet. An efficient memory management scheme would allocate all commands recording at the same frequency to the same memory, thereby using the space allocated to each sensor efficiently.

To retrieve logged data a command gives the number of the memory to be queried. There is no support to access all measured data from a particular sensor directly, but all memories need to be inspected and unwanted data discarded. This may be data for other sensors, or data summarised undesirably, or zero-data indicating that the desired sensor measurement was not recorded.

3.1 Programming the weather station

The weather station takes measurements according to a program comprising a sequence of STORAGE commands, each of which takes nine or ten arguments as follows:

No	command line number (1–64)
Command	summary function: either AVERAGE, MAXIMUM, MINIMUM or CURRENT
Mem	memory for storage (1–4)
Sensor	sensor number (1–6)
Format	1 unless Command is CURRENT, in which case it is 0
LimitValue	always 0
Param	always 1
Timetype	one of HOUR, EHOUR, EMIN or ESEC
Time1	counting number
Time2	number: only used when Timetype is HOUR

The Command values mean respectively to compute the average, maximum or minimum over the

[3]Environdata Weather Master 1600 - http://www.environdata.com.au/Product/Weather_Stations/WeatherMaster_1600.html

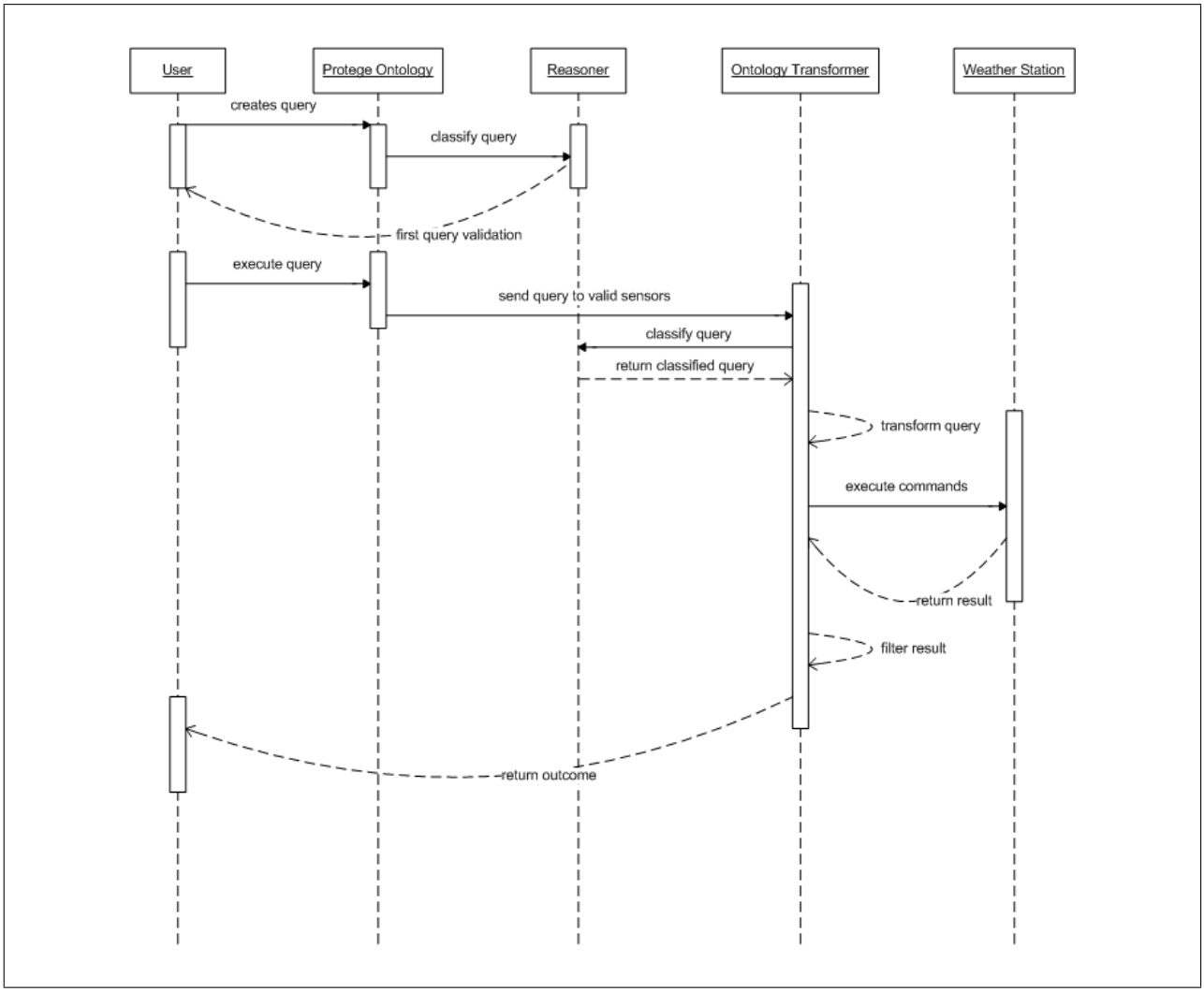


Figure 2: Message flow amongst components in the architecture for ontology-driven querying and programming of the weather station

time period given by Timetype, or to log an instantaneous measurement at the time given by Timetype. The Timetype values mean respectively to log every day at times given by Time1 and (optionally) Time2; every Time1 hours; every Time1 minutes, and every Time1 seconds.

For example, "STORAGE 1 AVERAGE 1 1 1 0 1 HOUR 9" means that command 1 logs the average air temperature (sensor 1) in memory 1 at 9am daily. "STORAGE 13 CURRENT 2 3 0 0 1 EHOUR 1 0" means that command 13 logs the current wind direction in memory 2 every hour. In general, every storage command should be bracketed by MEMOFF and MEMON commands; otherwise it also causes the memories of logged data to be cleared.

3.2 Querying the weather station

There are two weather station commands to retrieve measurements from the sensors: the simplest is the parameter-free R command which simply returns the current value for each sensor and its respective unit of measurement.

Measurements are retrieved from memory by a MEM command with a parameter corresponding to the memory number to inspect (1–4) and optionally a parameter to select a time period from the memory. This parameter may take the values ALL, for everything, UPDATE for data not previously retrieved, or SPECIFIC *from-time to-time* for data time-stamped within the given range (the times here are written in the form YYYY MM DD HH MM SS).

4 Ontology-Driven Client Tool for Programming

The major part of the device-independent query processing is performed within the client tool. The goal of the ontology-driven user interface is threefold: to support a common user interface to widely ranging sensor devices; to support the discovery and interaction with a sensor service by modelling the service capability in an accessible manner; and to offer the full capability of the sensor platform's native interface. We have chosen to offer an explicitly ontology-driven interface for this—as an experiment in stretching the applications of ontologies into new ground, and also in preparation for integration with other sensor systems enabled through the ontology development work of the W3C's SSN-XG (see Section 6). We have developed our interface as a plug-in to Protégé v3.3.1, although we plan to change to the new version 4.0 which has more features and is more convenient for working with OWL ontologies. Our ontology is sufficient for our case study purposes and its simplicity makes it possible to explain here, but we recognise that a much larger and differently-structured ontology would be needed for wider deployment.

4.1 Ontology modelling

We started our ontology by importing NASA's SWEET domain ontology[4] and extending it with classes and properties that model the command language of the Environdata weather station. The modelling is not complete, so we do not offer the full capability of the sensor platform for this prototype. However, our use of the ontology capability is extensive: it may be used for sensor discovery through the browsing and classification ability; it is used for phrasing queries—as definitions of ontology concepts (and thereby also supporting multiple syntaxes for a query function); it is used to detect redundancy in queries and so to improve efficiency, and it is used for validating queries and directing them to the appropriate sensors.

The sensors of the weather stations are described by disjoint subclasses of SWEET's *material_thing:Sensor*. We introduce the object property *measures* with domain *material_thing:Sensor*, and define each sensor through both a universal and an existential restriction on *measures* (for example, the temperature sensor *measures* SWEET's *property:Temperature*). This modelling enables the discovery of sensor platforms within the ontology by reference to the physical property measured. In addition, for each weather station sensor class we use datatype properties to represent parameters required for the weather station, such as *hasSensorNo* with domain *material_thing:Sensor* and an integer value. We create an instance of each sensor and initialise each *hasSensorNo* property value with the corresponding weather station-defined sensor number.

To model time commands, we create subclasses for each of the four SWEET time units: so we have a new *Day*, *Hour*, *Minute* and *Second* which are intended to refer to periods of time for measurement. Each of these classes we existentially restrict with the *hasSubPeriod* property over each of the smaller periods, and add a closure axiom for *hasSubPeriod*—the purpose of this is given later in the paper. Now we allocate a weather station memory to each period—in order to manage the memory efficiently according to the memory management scheme described earlier. To do this, we create the datatype property *usesMemory*, create an instance of each time period, and instantiate it's *usesMemory* value with each of the four memory numbers (1–4). We also create a *timeTypeCmd* datatype property of the time period classes, and instantiate each with the corresponding weather station timetype string: HOUR, EHOUR, EMIN, or ESEC. Further, to support retrieval queries over a time range, we create a *Date* as a subclass of SWEET's *time:Instant*, with integer datatype properties *atYear*, *atMonth*, *atDay*, *atHour*, *atMinute* and *atSecond* to describe a time instant.

We model the summary function part of the weather station capability as subclasses of a new class *Statistic*: *Average*, *Maximum*, *Minimum* and *Current*, with a datatype property *statCmd* instantiated with the value of the corresponding command string, as before.

With these basic concepts in place, now we turn to modelling general schemas for the commands of the weather station. Grouped under a *WM1600Capabilities* class, we provide (complete) descriptions of three capability classes as described in figures 3 and 4 in the form of Protégé screen copies. These capability classes correspond respectively to the weather station's commands R for current sensor readings, MEM for logged data, and STORAGE for reprogramming, as described in Section 3.

[4]Semantic Web for Earth and Environmental Terminology, (SWEET) v1.0 at `http://sweet.jpl.nasa.gov/index.html`

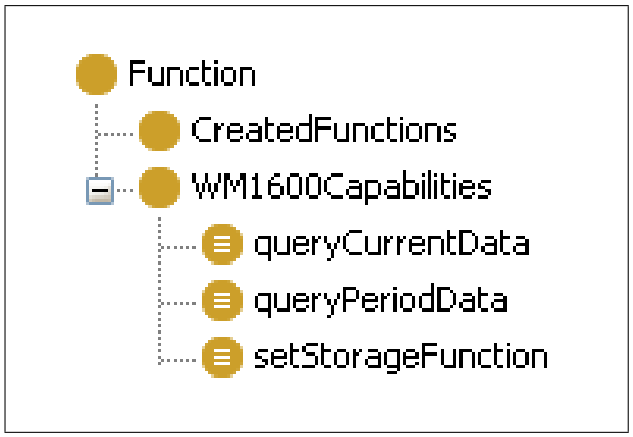


Figure 3: Hierarchy of Functions, Queries and Capabilities in the capability ontology

Finally, we also create a class to describe the functions of our sensor platform. This serves as documentation for the device, by grouping its functions, can be used to discover which devices can perform certain functions. We define the property *hasFunction* (which may be used for all such sensor definitions) and the class *WM1600* as in Figure 5. Note that other sensors, especially other models from the same manufacturer, may have a similar definition, reusing some of the capability classes we have defined for the WM1600.

Now we show how these descriptions of weather station capabilities can be used to compose queries for the weather station.

4.2 Querying the weather station

Our Protégé plug-in provides a query interface for the weather station, loading up the sensor ontology, supporting browsing of the ontology and query formulation, and communicating with the ontology transformer. A user defines a query as a new class definition in the context of the sensor ontology described above. The query is classified, using the services of the connected reasoner. The query must be subsumed by a weather station capability class in order to be a valid weather station command. We use the power of the DL classification in this way to admit alternative syntaxes for the same query by, for example, allowing the definition and use of classes as subqueries in more complex queries. The same method can also be used to assign queries to multiple devices capable of handling the query in which case such a query would have multiple capability class parents (although in our case study we use only one device).

Figure 6 shows the plug-in window for composing queries as class definitions: in this case the query screen has been generated at run-time specifically for the *WM1600* by inspecting its definition in the ontology. Next, the plug-in prompts the user to instantiate the query class just created, and this gathers the predefined datatype property values described earlier, and permits further refinement such as the entry of dates for range queries.

Figure 7 shows the form of two simple queries (displayed in Protégé) after entry through the query screen. The first, *curFunc*, requests the current data for the temperature sensor only. The second, *perFunc*, is a more complex query for raw temperature data from storage for a given period. The figure also shows the instance data attached to the second query (displayed under the "Individual" tab of

Figure 4: Definitions of the *queryCurrentData*, *queryPeriodData*, and *setStorageFunction* capabilities, respectively

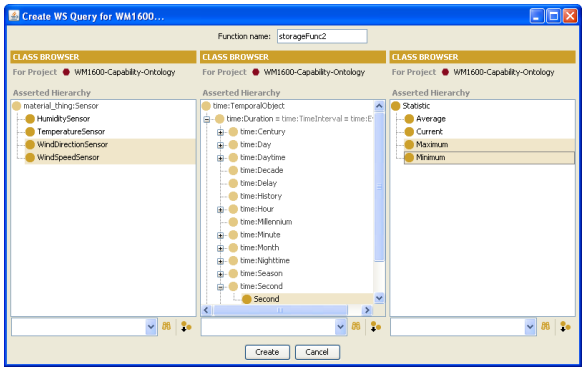


Figure 6: Screenshot of the Plug-In window to create a query

Protégé) connecting the query instance to properties identifying the sensor and the selected time range parameters. This individual view is no longer available in Protégé v4.0.

Finally, the plug-in invokes the ontology transformer that is configured for the selected device, passing on both the name of the new query class and the query instance.

5 Query Translation and Execution

Each different sensor platform is configured with a dedicated "ontology transformer" which, in our case is written in Java code. It communicates with the Protégé plug-in via sockets and with the weather station, via a communication handler, over a serial port on its host. The same transformer may be used for multiple deployments of the same device, but a different one is needed for devices that have different modelled capabilities in the ontology. A detailed description of the ontology transformer is out of scope for this paper: although we note that it is responsible for dealing with device-specific issues such as correlating the device commands with responses, applying error checking and correction as required and adding or filtering unwanted device-specific characters in the command and response dialogue.

The main role of the ontology transformer is to manage the state of the sensor device (through configuration and status commands), to translate the class and instance form of the query to the required command line form, and to filter response data to match the query where necessary. The transformer classifies the query in the context of the ontology (as before in the client) and traverses the ancestor paths to match the name of a capability class known to the transformer. In Figure 8 we illustrate the classification of the example queries given previously (plus several others), with respect to the capability classes of the weather station.

The ontology transformer for the weather station

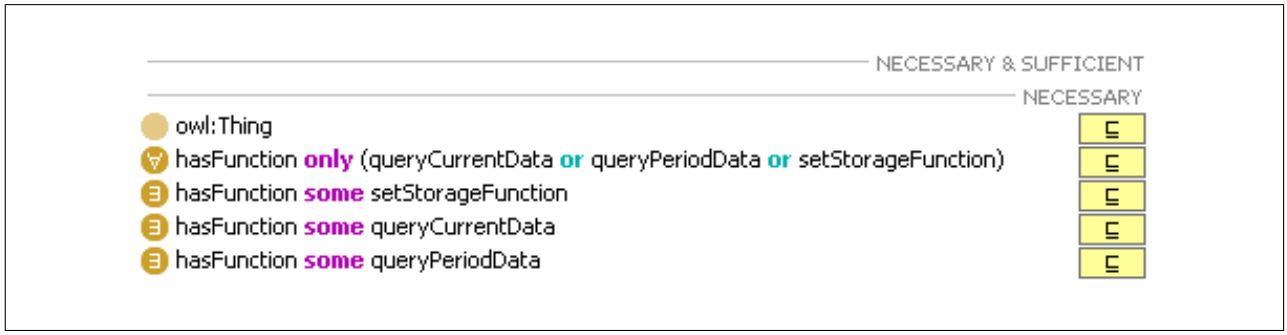


Figure 5: Definition of the *WM1600* class defining its capabilities

then passes the query instance to a handler specialised for that capability class that can retrieve the parameters from the instance and generate the appropriate command strings for the weather station. For some query classes it generates multiple commands: for example the STORAGE command is bracketed by MEMOFF and MEMON commands.

When the corresponding response is received from the weather station, the capability class handler may also filter the response data to match the query. For example, although our *queryPeriodData* capability class supports selection of sensors, this capability is not matched in the corresponding weather station "R" command, so the capability class handler uses the *hasSensorNo* property value in the individual to select only those columns of data corresponding to the desired sensors from the full response, and only those rows with non-zero values for that sensor reading, since a zero value indicates a non-measurement. Similarly, the logged rows with a different summary function to that requested in the query must be removed.

For the reprogramming queries, we also use the classification of the query to perform a query optimisation step to reduce the load on the weather station. Taking advantage of our semantic model for time corresponding to the weather station's EHOUR, EMIN and ESEC parameters, we detect when a reprogramming query is subsumed by an earlier (already executing) reprogramming query. In such a case, the ontology transformer does not submit the query to the weather station, because the corresponding data can already be retrieved from the log due to the pre-existing query.

Finally, the filtered data is returned to the client for display to the user, as in Figure 9.

6 Related Work

This is the first time ontological descriptions and formal description logic reasoning has been used to assist in sensor network programming and querying, although other"semantic" representations have been used for similar purposes.

For example, in (Liu & Zhao 2005) and (Whitehouse et al. 2006) an ontology of sensors, comprising just a simple, explicit, taxonomy of sensor types and represented in restricted prolog is used in conjunction with expressive prolog rules and associated Horn clause reasoning to derive and represent higher-order sensor network services, including composition of multiple sensors to deliver targeted sensor services. However, there is no mechanism for sensor tasking in this framework and the query model corresponds to simply accessing named attributes from a named sensor data stream.

More commonly, formal OWL ontologies have been used for describing sensors and their capabilities. In early work, (Eid et al. 2007) propose a simple taxonomy of sensors and their measurements which is used with an RDF query language for discovery of sensors via their descriptions. DL reasoning is proposed for validation during ontology development (as is customary for OWL ontologies), but there is no support for sensor querying or tasking in this work.

In (de Mel et al. 2009) an OWL-DL ontology and a reasoner is used as part of an algorithm for discovery of sensors suitable for tasks. The ontology of sensors and platforms includes information on sensor capabilities. Platforms (comprising multiple sensors) with any sensor capability that subsumes any of those required for the task are suggested as candidates. A covering algorithm is then used to select platforms for the task from the candidates. In our work, the same kind of subsumption reasoning for discovery of sensors (e.g. discovery of sensors that can measure temperature) is also a valuable feature of the ontology modelling we employ and is quite straightforward. However, in our work we extend the ontology model to support querying and programming of the sensors too.

In (Ha et al. 2007), the services ontology OWL-S is used to represent compositions of sensor instructions and an hierarchical task network planning algorithm is used to discover new compositions. Although this work focuses on tasking sensors, and can generate instruction sequences for sensor devices, it relies on a procedural programming model and procedural reasoning; the ontology is otherwise only used for informal partial matching of input and output datatypes.

In recent work, (Compton et al. 2009) propose rich ontological descriptions of sensors coupled with a number of reasoning services, including sensor classification for discovery and also composition of sensors. This work does not address a mechanism for querying or programming sensors in their native device languages, although it might be possible to encode a procedural sensor program to perform a task within its OWL-S-like process model.

One of the OGC's Sensor Web Enablement standards, the Sensor Planning Service (Simonis 2005), aims to provide a web service interface to program sensor devices, although another service is used to retrieve the data. However, in its current form the service does not use descriptive semantics. Although it provides an operation ("DescribetaskingRequest") to retrieve a description of the commands from the sensor service, the response encodes a list of (unconstrained) parameter names and their types with natural language descriptions for each. Although this description may be sufficient for a user to compose a query to the service, relying on the predefined syntax pattern, there is no support for contextual modelling of the meaning of the parameter names, nor of the commands and functions overall as is offered in our approach. Further, it is aimed at human in-

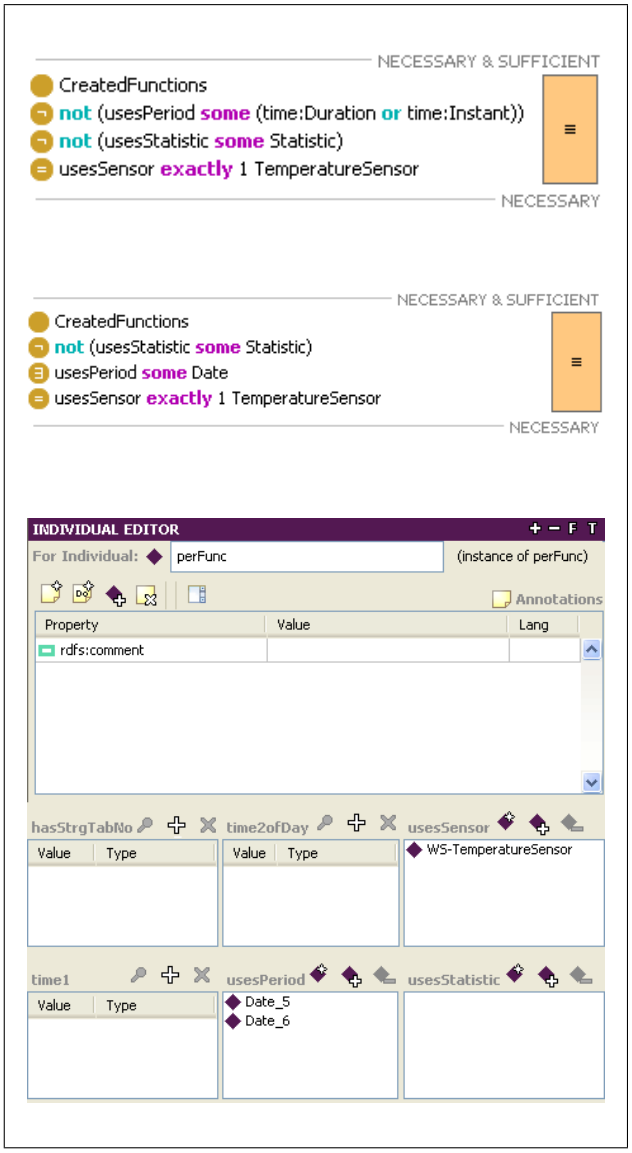


Figure 7: Definitions of queries: *curFunc* to retrieve current temperature data, *perFunc* for temperature data over a period of time, and the instance data for the *perfunc* query, respectively

terpretation and does not facilitate machine inference that could otherwise automate high-level tasks, such as sensor substitution, composition and integration. It does address communication heterogeneity through its standard service protocol, but does not address heterogeneity at the device level.

Finally, under the auspices of the W3C, the SSN-XG[5] is currently developing an expressive device ontology aimed at supporting a number of use cases. This includes acting as a source of terminology for markup of OGC Sensor Web Enablement Web services, but may also include more direct sensor network tasking. Although the ontology will certainly not be developed to the extent of capturing individual sensor network programming languages, as is done in the case study for this paper, it will certainly provide a richer context ontology for embedding such programming concepts than is offered by SWEET that is used in this paper. Employing such a well-developed context ontology will improve the user experience and value of the semantic representation and processing that is demonstrated in this paper.

[5]W3C Semantic Sensor Network Incubator Group, `http://www.w3.org/2005/Incubator/ssn/`

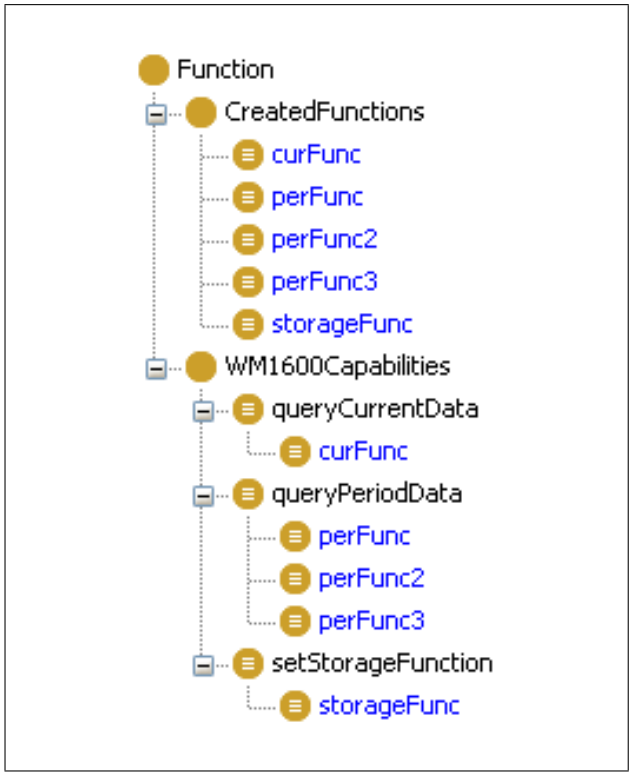


Figure 8: Subsumption of valid queries by capability classes of the weather station

7 Discussion and Conclusion

We have described a case study application of formal ontologies to the problem of programming and querying sensor networks. Like many other approaches to managing heterogeneity, our approach wraps the native interface in a customised device-specific component to manage heterogeneity in the native communication protocol and, to some extent, in the interaction paradigm, command language, and response encoding. However, atypically, our approach also uses a high-level semantic description of the domain and the command language to offer a common user interface for access. This description amounts to configuration data in the client tool—enabling user or application customisation and evolution without reprogramming.

Through our model employing DL descriptions of sensor devices we have shown how DL classification can offer more than just sensor description and discovery. It can also be used to assist in programming and querying heterogeneous sensor networks. We have concentrated on a declarative model for sensor network commands, rather than an alternative process model. This design may make our approach easier to use, but it may be unwieldy and insufficiently expressive for heavily state-based devices, such as those that are programmed to move in space prior to taking a measurement.

Although our case study has been applied to only one sensor device, it is easy to see that the approach can extend to other sensor devices, and to incorporate them all within the same client environment. One such extension might, for example, provide an ontology transformer for the OGC Sensor Planning Service (together with a chosen data retrieval service) to enable modelling of the non-standardised parameter names and sensor capabilities within a semantic framework. Another extension might develop an ontology transfomer for sensors mounted on motes in sensor networks: the transformer would need to

translate the semantic commands to code fragments, embed the fragments within a complete sensor network program in NesC, for example, and distribute the program to the nodes using an over-the-air programming tool.

In this case of a widely distributed and heterogeneous network, it would also be necessary to manage the ontology more carefully at run time: an ontology server could be used to retrieve a desired domain ontology and a set of relevant device ontologies (where relevance may be defined by user context, such as location or access rights). The relevant device and domain ontologies would be merged and delivered to both the client tool and the required ontology transformers on demand.

Our pure-semantics approach to managing the heterogeneity offers a number of other benefits over more conventional architectures for similar problems of heterogeneity in distributed systems. Our ontology offers a data-driven (i.e not pre-programmed) context model to assist in the discovery and querying of sensors suitable for a user task. This enables modelling of sensors by location, type, observable properties, accuracy, availability, platform, mobility, or any other aspects relevant to discovery and application. In our prototype, we have used a simple extension to the well-known ontology-independent Protégé tool to achieve this, although a deployable implementation would use a specialised client tool for better ease-of-use targeted to user expectations.

The ontology modelling offers other benefits in querying. For example, the same device capability may be offered through multiple syntaxes, or via multiple concepts, through careful ontology design. Provided that a query can be mapped to a supported capability class by subsumption computed by the reasoner, verifying that the query is semantically correct, no programming is needed to support such alternative query syntaxes. The ontology transformer, already designed to handle the capability class, will therefore also handle the subsumed query. Further, the verification function can assist a developer to present the device capabilities in a convenient (but correct) way and can also assist a user to understand the device capabilities. As noted by (Compton et al. 2009) integration with inference mechanisms other than DL alone could also help here, for example, reasoners for spatial modelling could help when access to multiple sensors is required for the desired result. Similarly, spatial reasoning may assist with modelling of mobile devices in a declarative manner, so that sequential locator commands may be unnecessary.

The ontology modelling also enables, at least in our case study of the weather station command language, the use of subsumption reasoning to infer when a command to the weather station is redundant because the answers may be retrieved from data produced by an already-executing command, after filtering. In a multi-user sensor network environment, this may offer significant gains in efficiency.

An OWL 2 DL implementation (Motik et al. 2009) of this work would improve the benefit of the verification by subsumption test further: the *data ranges* capability, for example, would offer tighter modelling of sensor device specifications and identification of redundant commands.

In the longer term, we envisage our work contributing to a highly distributed, multi-organisational sensor networked, computation environment. This is needed for the emerging transdisciplinary science that is required to address the big questions in our shared future.

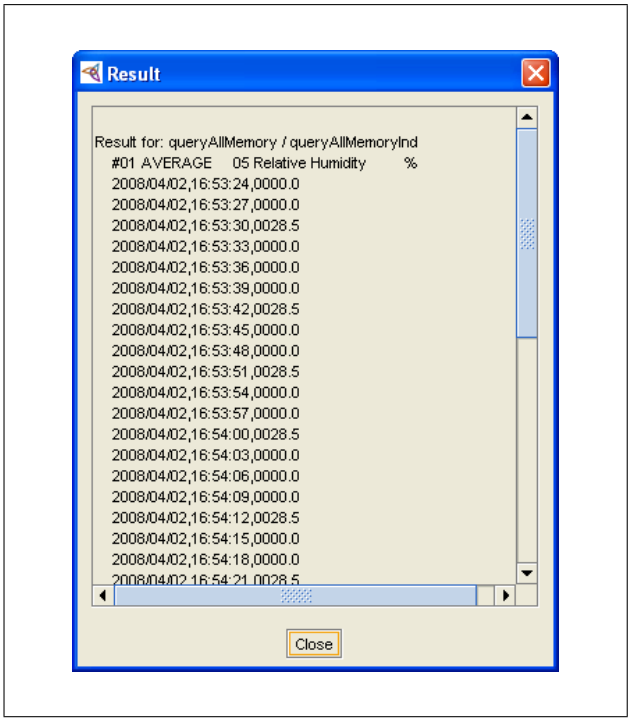


Figure 9: Result window with the answer from the Weather Station

References

Chu, D., Popa, L., Tavakoli, A., Hellerstein, J. M., Levis, P., Shenker, S. & Stoica, I. (2007), The design and implementation of a declarative sensor netowrk system, *in* '5th ACM Conference on Embedded Network Sensor Systems (Sensys 2007)', Sydney, Australia, pp. 175–188.

Compton, M., Neuhaus, H., Taylor, K. & Tran, K.-N. (2009), Reasoning about sensors and compositions, *in* K. Taylor, A. Ayyagari & D. De Roure, eds, 'Proceedings of the 2nd International Workshop on Semantic Sensor Networks, SSN09', Vol. 522, CEUR workshop Proceedings, Washington DC, USA.
URL: *http://ceur-ws.org/Vol-522*

de Mel, G., Sensoy, M., Vasconcelos, W. & Preece, A. (2009), Flexible resource assignment in sensor networks: A hybrid reasoning approach, *in* Corcho, Hauswirth & Koubarakis, eds, '1st International Workshop on the Semantic Sensor Web (SemsSensWeb 2009)', Vol. 468, CEUR workshop Proceedings, Heraklion, Crete, Greece.
URL: *http://ceur-ws.org/Vol-468*

Duchesne, P., Maué, P. & Schade, S. (2008), Semantic annotations in OGC standards, Discussion Paper OGC 08-167, Open Geospatial Consortium.
URL: *http://portal.opengeospatial.org*

Eid, M., Liscano, R. & Saddik, A. E. (2007), A universal ontology for sensor networks data, *in* 'IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 2007)', Ostuni, Italy, pp. 59–62.

Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E. & Culler, D. (2003), The nesC language: A holistic approach to networked embedded systems, *in* 'Programming Language Design and Implementation (PLDI) 2003'.

Ha, Y.-G., Sohn, J.-C., Cho, Y.-J. & Yoon, H. (2007), 'A robotic service framework supporting automated integration of ubiquitous sensors and devices', *Inf. Sci.* **177**(3), 657–679.

Li, L. & Taylor, K. (2008), A framework for semantic sensor network services, *in* 'ICSOC 2008: 6th International Conference on Service Oriented Computing', Vol. 5364 of *LNCS*, Springer, Sydney, Australia, pp. 347–361.

Liu, J. & Zhao, F. (2005), Towards semantic services for sensor-rich information systems, *in* 'Proceedings the 2nd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (Basenets 2005)', Boston, MA.

Motik, B., Patel-Schneider, P. F. & Parsia, B. (2009), OWL 2 web ontology language structural specification and functional-style syntax, Technical report, W3C. W3C Proposed Recommendation.
URL: *www.w3.org/TR/2009/PR-owl2-syntax-20090922/*

Simonis, I. (2005), OpenGIS sensor planning service, OpenGIS Discussion Paper OGC 05-089r1, Open Geospatial Consortium.
URL: *http://www.opengeospatial.org/standards/sps*

Whitehouse, K., Liu, J. & Zhao, F. (2006), Semantic streams: a framework for composable inference over sensor data, *in* 'Proceedings of the 3rd European Workshop on Wireless Sensor Networks (EWSN)', LNCS, Springer.

Review of semantic enablement techniques used in geospatial and semantic standards for legacy and opportunistic mashups

Laurent Lefort
CSIRO ICT Centre
GPO Box 664 Canberra, ACT 2601, Australia
laurent.lefort@csiro.au

Abstract

Networks of sensors are increasingly used to monitor essential environmental variables for biodiversity, water, and climate change research. Such multidisciplinary scientific projects require more flexible ways to publish and aggregate sensor observations from different networks as mashable web resources. Semantically-enabled and linkable descriptions of sensors and sensors services can simplify the integration of legacy backend sensor web services and make it easier for mashup developers to opportunistically combine these resources.

This paper reviews linking and annotation techniques applicable to the development of geospatial mashups services. It describes how approaches based on RDFa could supersede existing techniques for the semantic annotation of RESTful services. It highlights specific issues linked to the hybrid nature of mashups combining solutions based on XML, RDF and HTML standards and the failure risks attached to such multi-standards knowledge systems. It points out the pending technical issues, especially the ones where more coherent approaches are needed e.g. the upgrade of existing standards like XLink and SAWSDL or the integration of validation tools developed for each family of standards.

Keywords: semantic web, sensor web, geospatial standards, mashup, XLink, RDFa.

1 Introduction

As networks of sensors are increasingly used to monitor essential environmental variables for biodiversity, water, and climate change research, we need innovative approaches to simplify the integration of sensor observations from different networks into mashable web resources. Pairing geospatial standards developed by the Open Geospatial Consortium (OGC) and semantic web standards developed by the World Wide Web Consortium (W3C) can foster new approaches for applications that are not (or not yet) clear candidates as web standards.

Apart from the Keyhole Markup Language (KML), most OGC standards have been developed prior to the appearance of modern mashup techniques. The W3C Semantic Sensor Network Incubator Activity[1] (SSN-XG) develops semantic descriptions of sensors and sensors services to semantically enable services based on Sensor Web Enablement (SWE) standards like the Sensor Markup Language (SensorML) and the Sensor Observation Service (SOS). This review focuses on linking and annotation techniques which can support the discovery and composition of these services and their integration into web mashups (Le Phuoc and Hauswirth 2009).

This paper is structured in four main parts. Section 2 defines legacy and opportunistic mashups and how they can be combined in a multi-layered integration scheme. It also discusses how this scheme may evolve with the introduction of new mashup engines and technologies based on existing and actively developed semantic web standards. Section 3 reviews the XML, HTML and RDF-based linking, and annotation methods and their applicability in this context. Two practical examples are used in Section 4 to compare the available approaches and to identify the innovative features of RDFa which are applicable to the semantic annotation of RESTful services. The discussion in Section 5 identifies failure risks which are specific to knowledge systems including sources of interfaces problems likely to occur in such multi-standard setups. It also points out the pending technical issues, especially the ones where more coherent approaches are needed e.g. the upgrade of existing standards like XLink and SAWSDL or the integration of validation tools developed for each family of standards.

2 Typology of mashups

2.1 Multi-layered mashup framework

The Model for layered integration tools proposed by Gamble and Gamble (2008) groups pre-Web, Web 1.0 and Web 2.0 technologies into three separate integration zones with decreasing level of integration effort and increasing readiness for opportunistic development. In this framework, *legacy mashups* require more work because the integration of pre-Web and Web 1.0 resources generally requires the development of custom-made wrappers. First generation mashup engines such as Damia, Yahoo Pipes, Popfly, or Google Mashup Editor (Di Lorenzo et al. 2009, Koschmider et al. 2009) enable the creation of *opportunistic mashups* based on the most popular Web 2.0 service API (Application Programmable Interfaces). These mashup engines have been very successful even if they are often tied to proprietary APIs or platforms.

[1] http://www.w3.org/2005/Incubator/ssn/

Figure 1 illustrates the layered model defined by Gamble and Gamble (2008) where the two types of integration approaches cohabit. Legacy services are integrated in the first integration layer as legacy mashups. The resulting services are exploited in the second integration layer with more lightweight mashup methods.

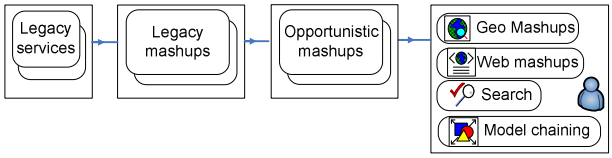


Figure 1: Multi-layered mashups

2.2 Non-semantic mashups

Geospatial and Sensor web service-oriented platforms can combine Web 2.0 technologies like Ajax to global geospatial data resources like Google to enable the online publication of geospatial and sensor datasets and services. Mashable APIs are now available for geospatial and sensor web resources like Google Maps[2] or Pachube[3] and from popular GIS tools like ArcGIS[4].

Figure 2 presents a simple example of multi-layered geospatial mashup. ArcGIS can be used to integrate data from OGC web services and expose it through proprietary Javascript APIs5 which can be further mashed up in Web 2.0 tools like Google maps.

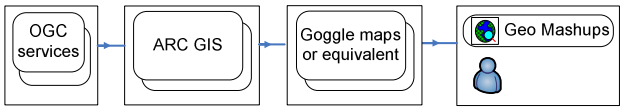


Figure 2: A simple geo-mashup based on Arc GIS

2.3 Semantic mashups

The lack of extensibility of existing APIs is driving the development of the next generation of *semantic mashup* engines based on semantic web standards developed by W3C. SAWSDL (Kopecký et al. 2007) uses semantic descriptions to enable the composition of web services for *legacy semantic mashups*. These rich semantic descriptions help to compose geospatial services (Lemmens et al. 2007, Vaccari et al. 2009). Custom-made operators are often developed to transform the data from XML to RDF (Henson et al. 2009) and to better manage its provenance (Sahoo et al. 2008).

Opportunistic semantic mashups generally use RDF (triple stores) resources applying the Linking Open Data conventions (Bizer et al. 2007) via standard APIs based on SPARQL (Prud'hommeaux et al. 2008, Clark et al. 2008) or via proprietary query languages offered by Web-based development environments such as Metaweb ACRE[6] or Yahoo Pipes[7] designed to offer the possibility for end users to develop and share their mashups.

Opportunistic semantic mashups can also source data from HTML pages, especially from RDFa (Adida et al. 2008) snippets embedded in web pages. RDFa, originally designed as an extension of XHTML2 and now ported[8] to HTML5[9] is a hybrid method devised to sprinkle RDF data or metadata in a web page and make it available for further content aggregation down the track, e.g. at the level of search engines (Benjamins et al. 2008). Search platforms like Google and Yahoo SearchMonkey[10] exploit RDFa content to improve search results and use it in search engine results as richer snippets (Goel et al. 2009).

DERI Pipes (Le Phuoc et al. 2009), MashQL (Jarrar and Dikaiakos 2009) and TopQuadrant's SparqlMotion[11] are three examples of semantic mashup engines which allow end users to chain (or pipe) simple URI-based data integration operators. DERI Pipes users can fetch data from XML using XQuery, from RDF using SPARQL and extract embedded RDFa and microformat data from HTML using purpose-built operators. Figure 3 presents a semantic mashup architecture implemented by Le Phuoc and Hauswirth (2009) which combines a semantic wrapper for Sensor Observation Service similar to SemSOS (Henson et al. 2009) with a SensorMasher application based on DERI pipes. In this implementation, SPARQL is used to query data from the sensor ontologies and from the sensor data streams.

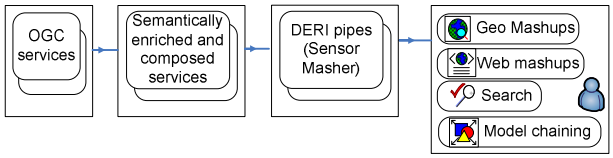


Figure 3: A multi-layered semantic mashup

2.4 Semantic enablement methods

There are four basic *semantic enablement* methods for legacy and opportunistic mashups applicable at different levels of the multi-layered scheme described in Figure 1:

- Inclusion of remote RDF (or SKOS/OWL) resources in XML using XLink,
- Annotation of web services with SAWSDL,
- Annotation of RESTful web services using hRESTs (or SA-REST, MicroWSMO),
- Inclusion of remote RDF (or SKOS/OWL) resources in HTML using RDFa.

[2] http://code.google.com/apis/maps/

[3] http://www.pachube.com/

[4] http://www.esri.com/software/arcgis/

[5] http://www.esri.com/javascript

[6] http://www.freebase.com/apps/

[7] http://pipes.yahoo.com/

[8] http://dev.w3.org/html5/rdfa/rdfa-module.html

[9] http://www.w3.org/TR/html5/

[10] http://developer.yahoo.com/searchmonkey/

[11] http://www.topquadrant.com/products/SPARQLMotion.html

The next section reviews the basic XML, HTML and RDF-based linking and annotation standards and their relevance to the four semantic enablement methods defined above. For this purpose, the following terminology is used. *Mashable content* corresponds to any type of remotely managed resources which can be used in a mashup. *Links* specifies the inclusion of remotely managed resources. *Semantic annotations* define how to map service capabilities to semantic definitions to enable the discovery or composition of web services. The transition from XML-based services to RDF-based services is called a *lifting* operation (Farrell and Lausen 2007) and the inverse one, from RDF to XML is called a *lowering* operation.

3 Linking and annotation methods

3.1 Handling mashable content with javascript

Mashable content can be extracted from XML, RDF (OWL) and HTML resources, and from RDFa snippets included in web pages. Different javascript libraries (see Table 1) can be used to process data sourced from different origins.

Mashed up content	Javascript library
XML resource	JQuery http://jquery.com/
RDF resource	JSON http://www.json.org/ used to serialise SPARQL results http://www.w3.org/TR/rdf-sparql-json-res/
OWL resource	JOWL (JQuery extension) http://jowl.ontologyonline.org/
HTML snippet	JQuery http://jquery.com/
RDFa snippet	rdfQuery (JQuery extension) http://code.google.com/p/rdfquery
Microformat snippets	A custom-made javascript library is needed for each different microformat

Table 1: Types of mashable content

Interest for RDFa is growing fast because the prospect for being able to extend documents without having recourse to standards organisations is enormous and because the addition of RDFa content to already published web pages can be done without forcing the web site designers to change the look of their sites.

Microformats are available for a number of specific applications with various levels of popularity and support. The HTML5 Microdata proposal is an attempt to offer a generic alternative to the existing Microformat coding conventions. It is not reviewed here because this set of requirements (Hickson 2009) can be considered as a subset of the requirements addressed by RDFa.

3.2 Linking methods

Links are defined here as mechanisms used to extend available content from any type of resources with information sourced from remotely managed content (type or instance). Links are possible between two documents of the same type or between documents of different types. Table 2 lists the techniques used to link documents to each other on a range of use cases which can occur in mashups.

Linked resource type	Linking method	Type of link
XML	XLink	XML to XML
XML	XLink	XML to URNs
XML	XLink	XML to RDF
XML	RDFa	XML to RDF
RDF	OWL mapping properties or weaker alternatives like umbel:isLike	RDF to RDF
SKOS	SKOS mapping properties	SKOS to SKOS
OWL	OWL mapping properties	OWL to OWL
HTML	Microformats	HTML to "data"
HTML	RDFa or Common Tag	HTML to RDF

Table 2: Linking methods

The XML Linking language or XLink (DeRose et al. 2001) is a W3C standard which allows the creation of links between XML resources. It is commonly used in OGC standards to include references to external vocabularies managed with URNs.

To link RDF-based vocabularies, ontologies or Linking Open Datasets (LOD) content, the most common approach is to use the basic relationships defined in the Web Ontology Language OWL: owl:sameAs, owl:equivalentClass, owl:equivalentProperty although for plain LOD content, weaker alternatives may be preferable like the one proposed by the UMBEL[12] developers. SKOS[13] offers a richer range of properties (exactMatch, closeMatch, broaderMatch, narrowerMatch) to specify the relationships between concepts.

3.3 Semantic annotation methods

Different semantic annotations methods are needed for WSDL web services and RESTful web services.

Upgrading WSDL web services into semantically enabled services can be done with the help of SAWSDL (Kopecký et al. 2007), now a W3C Recommendation (Farrell and Lausen 2007). The SAWSDL specification has three main features:

- Semantic definitions (in a RDF-based format like OWL) may be included in the WSDL file.
- A small set of elements and attributes can be added in different parts of the WSDL service description to create links from XML schemas elements and attributes to their *model references* which are semantic definitions.
- And finally, additional attributes can be used to associate a schema type or element with a mapping script describing lifting transformation from XML to RDF and lowering transformation from RDF to XML.

Upgrading REST web services into semantically enabled services requires different tools because the service

[12] http://www.umbel.org/

[13] http://www.w3.org/TR/skos-reference/

declaration is generally made within a HTML web page and does not use an XML-based description format. SA-REST (Lathem et al. 2007, Sheth et al. 2007) and MicroWSMO (Kopecký et al. 2009) are two related efforts which use the same semantic annotation microformat, hRESTs (Kopecký 2008). The SA-REST approach is more closely related to the SAWSDL standard while MicroWSMO uses a different ontology: WSMO-Lite.

3.4 Types of lifting operations

GRDDL (Connolly 2007) defines the syntax to embed the reference to a lifting script in any type of well-formed XML format. The file to which the GRDDL annotation has been added is used as the input of the specified lifting operation. The RDF output depends on the location of the GRDDL markup. If the corresponding transformation is available, any HTML files containing microformat-based annotations can use this mechanism to be transformed into RDF.

SAWSDL, SA-REST and MicroWSMO also require the development of custom-made scripts. A major difference is that these scripts specify how to process the XML data manipulated by the service, not the content of the file containing the annotations.

RDFa defines a generic lifting mechanism to transform the annotations included in an HTML file into RDF. In this case, there is no need for user-developed scripts.

Lifting scripts may use languages like XSL transformations[14] (XSLT) or XQuery[15]. Lowering scripts may use hybrid approaches like XSPARQL (Akthar et al. 2008), a W3C Member Submission[16] which mixes XQuery and SPARQL. RDFa users can also use alternative implementations such as the ones available in javascript (Table 1).

4 Comparison of key linking methods

A short summary of the key features of each method is provided below. A more direct comparison is also done on two examples to complete this analysis in relation to two critical issues:

- Choice between the hRESTs microformat and RDFa for the semantic annotations of REST-based services and consistency of these approaches with existing ones (SAWSDL),
- Choice between XLink and RDFa as the linking technique used for XML content.

The first example focuses on semantic annotation requirements to guide the future work on REST services and also bridge the gap between these new methods and what can currently be used for WSDL.

The second example illustrates the differences between the XML-friendly solution based on XLink and the alternative approach based on RDFa.

4.1 Key attributes for each approach

RDFa: for the purpose of this review, we use the W3C Recommendation version of RDFa (Adida et al. 2008).

Attribute	Description	Intended RDF
about	The identification of the resource (to state what the data is about)	rdf:about of domain resource
typeof	RDF type(s) to associate with a resource	rdf:about of class of a resource
href	Partner resource of a relationship ('resource object')	rdf:about of range resource
property	Relationship between a subject and some literal text ('predicate')	rdf:about of datatype property
rel	Relationship between two resources ('predicate')	rdf:about of object property
rev	Reverse relationship between two resources ('predicate')	rdf:about of (inverse) object property
src	Base resource of a relationship when the resource is embedded 'resource object')	rdf:about of domain resource
resource	Partner resource of a relationship that is not intended to be 'clickable' ('object')	rdf:about of range resource
datatype	Datatype of a property	XML type range of datatype property
content	Machine-readable content ('plain literal object')	Value for datatype property

Table 3: RDFa attributes

In RDFa, the about and resource attributes plays the role of rdf:about and rdf:resource attributes in RDF. They can be encoded as compact URIs or CURIES (Birbeck and McCarron 2009), a syntax inspired by the prefix management conventions used in SPARQL. The content of a datatype property can be included as an extra attribute (content) or retrieved from the element content.

hRESTs: hRESTs focuses on the capture of mapping information between the service description and a reference ontology. The additional information is provided through the coding of the lifting script applicable to the service outputs. The hRESTs microformat specification used here is the one published by Kopecký et al. (2009) and the associated examples.

[14] http://www.w3.org/TR/xslt20/

[15] http://www.w3.org/TR/xquery/

[16] http://www.w3.org/Submission/2009/01/

Attribute	Description	Intended RDF
class	Type of XML or WSDL element (service, operation, address, method, input, output, label)	rdf:about of class of domain resource
href next to rel="model"	association between a WSDL or XML schema component and a concept in some semantic model	rdf:about of range class = modelReference
href next to rel="lifting"	Lifting script URL	N/A
href next to rel="lowering"	Lowering script URL	N/A
id	Locally declared id of WSDL element (to be combined with the document URL)	rdf:about of domain resource

Table 4: hRESTs Microformat attributes

The HRESTs microformat mandates the use of blocks with class elements in a rigid parent-child hierarchy (e.g. service contains operation) which will be implicitly transposed in the resulting RDF file.

XLink: For the purpose of this review, we will use the XLink guidelines documented for the Geography Markup Language standard (Portele 2007) rather than the original W3C specification Xlink (DeRose et al. 2001). Table 5 summarises the attributes defined by this specification.

Attribute	Description	Intended RDF
xlink:href	Identifier of the resource which is the target of the association, given as a URI	rdf:about of range resource
xlink:role	Nature of the target resource, given as a URI	rdf:about of class of range resource
xlink:arcrole	Role or purpose of the target resource in relation to the present resource, given as a URI	rdf:about of object property linking domain element to range resource
xlink:title	Text describing the association or the target resource	rdfs:comment

Table 5: XLink attributes

4.2 Feature comparison: hRESTs and RDFa

Kopecký et al. (2009) also specify how hRESTs can be expressed in RDFa. Table 6 is based on this input. The main difference is that hRESTs in RDFa allows the user to specify the target ontology through the definitions of the typeof, rel, property and datatype attributes.

RDF mapping	hRESTs in Microformats	hRESTs in RDFa
Domain instance	id (URL-prefixed)	about
Domain class	class (closed list)	typeof
Object property	ref="model"	rel
Inverse object property		rev
Range instance		href or resource
rdf:about of range class	href	typeof
Datatype property		property
Datatype property type		datatype
Range value		content or element content

Table 6: Comparison of RDFa and hRESTs

4.3 Feature comparison: XLink and RDFa

The direct comparison done in Table 7 can help to locate the major difference between XLink and RDFa which is that the two specifications cover different types of RDF triples:

- XLink: predicate (role) and object (href) for object properties
- RDFa: subject (about), predicate (rel) and object (href) for object properties and subject (about), predicate (property) and object (content or element content) for datatype properties

RDF mapping	Xlink	RDFa
Domain instance		about or src
Domain class		typeof
Object property	arc role	rel
Inverse object property		rev
Range instance	href	href or resource
Range class	role	typeof
Datatype property		property
Datatype property type	role	datatype
Range value		content or element content

Table 7: Comparison of XLink and RDFa

4.4 Examples of semantic annotations

The National Digital Forecast Database is a web service[17] developed by the U.S. National Weather Service to test the Digital Weather Markup Language (DWML). This forecast service (see also Al-Muhammed et al. 2007) is used here because it is simultaneously implemented as a WSDL service and as a REST service. Figure 4 shows an example of SAWSDL annotation in the WSDL file.

[17] http://www.nws.noaa.gov/ndfd/technical.htm

```
<wsdl:part name="endTime"
sawsdl:modelReference="http://sweet.jpl.nasa.gov/2.0/time.owl#End
" type="xsd:dateTime "/>
```

Figure 4: WSDL file with SAWSDL annotations (extract)

Table 8 lists the concepts defined in the SWEET 2.0 ontologies[18] which can be used as model references for the message parts of the NFDGen operation. Model references for service parameters like the product type (Time series or "glance") and the output type are specific to DWML and are not available in SWEET 2.0.

wsdl:part	SWEET 2 ontologies	Class
latitude	spaceCoordinates.owl	Latitude
longitude	spaceCoordinates.owl	Longitude
startTime	time.owl	Start
endTime	time.owl	End

Table 8: Types of mashable content

Many REST services are only documented through a web page. This is why semantic annotation methods like SA-REST or MicroWSMO can use any type of web page describing a service. The two options are to annotate the HTML page (or form) used to run the service (Figure 5) or a "WSDL-inspired" documentation page (Figure 6).

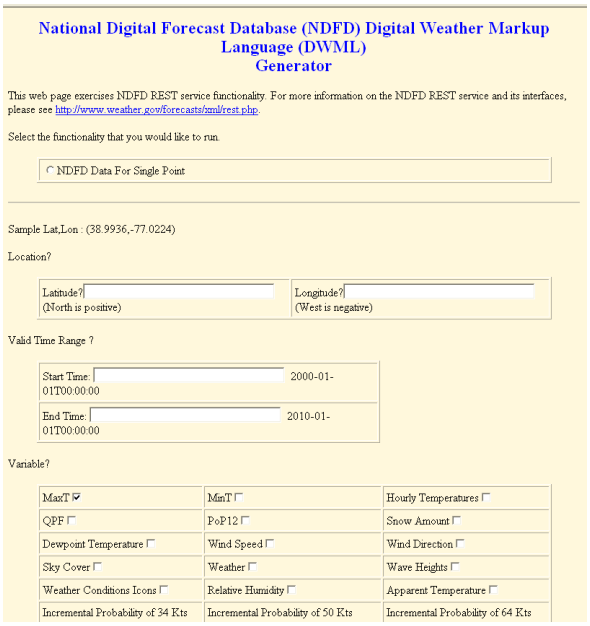


Figure 5: HTML form for a REST service (simplified)

In the first case, the HTML form can host the semantic annotations for the input data and for other elements used to run the service. An advantage of this approach is that the annotated form (Figure 5) can still be used to test that the service works. But additional content is required to annotate the output data (representations and faults).

The alternative is to have a documentation page in HTML which describes hierarchically the service, its operations, and the input and output format for each operation. This style of web page is comparable to what could be generated out of a WSDL file (when such a file is available). Figure 6 illustrates this approach with an HTML file generated out of (an extract of) the WSDL file with an existing XSL transformation[19].

Figure 6: HTML service description derived from WSDL

The two following examples present two types of annotations: hRESTs Microformat (Figure 7), and RDFa (Figure 8) applicable to the HTML form.

The hRESTs example (Figure 7) only includes semantic references for the sawsdl:modelReference attributes in SAWSDL. While the hRESTs solution may seem easier to use, it also requires extra effort for the end user to learn how the mapping between the class annotations used in the microformat (operation, action, input …) and the ontology used for the generated RDF content. This mapping may depend on the hRESTs toolset and on the availability of custom-made lifting and lowering scripts.

```
<FORM method="get" name="NDFDgenForm"
action="http://www.weather.gov/forecasts/xml/sample_products/bro
wser_interface/ndfdXMLclient.php"
class="operation">
 <DIV id="GmlTimePeriod" style="display: block; ">
   <P>Valid Time Range ?</P>
   <OL>
     <TABLE border="1" cellpadding="4" width="60%">
       <TBODY class="input">
         <TR rel="model"
href="http://sweet.jpl.nasa.gov/2.0/time.owl#End">
          <TD><span class="label">End Time</span>: <INPUT
type="text" name="endTime">
size="40" maxlength="80" value=""
onfocus="document.NDFDgenForm.endTime.value = &#39;2010-
01-01T00:00:00&#39;;">2010-01-01T00:00:00</TD>
       </TR></TBODY></TABLE>
   </OL> </DIV>
</FORM>
```

Figure 7: hRESTs example

[18] http://sweet.jpl.nasa.gov/ontology/

[19] http://tomi.vanek.sk/index.php?page=wsdl-viewer

The RDFa example (Figure 8) includes semantic references defining the type of annotations (e.g. sarest:operation). This approach gives more control to the end user for the choice of the service ontology and simplifies the task for the programming of tools which interprets the annotations. The RDFa specification (Adiba et al. 2008) defines processing rules which helps to combine these two types of semantic references seamlessly.

```
<FORM method="get" name="NDFDgenForm"
typeof="[sarest:action]"
action="http://www.weather.gov/forecasts/xml/sample_products/bro
wser_interface/ndfdXMLclient.php"
about="http://www.weather.gov/forecasts/xml/sample_products/bro
wser_interface/ndfdXMLclient.php">>
  <DIV id="GmlTimePeriod" style="display: block; ">
    <P>Valid Time Range ?</P>
    <OL>
      <TABLE border="1" cellpadding="4" width="60%">
        <TBODY rel="[sarest:input]”>
          <TR
about="http://www.weather.gov/forecasts/xml/sample_products/bro
wser_interface/ndfdXML.htm#GmlTimePeriod.endTime”
typeOf="[sweet20:End]” >
          <TD><span property="[rdfs:label]">End Time</span>:
<INPUT type="text" name="endTime">
size="40" maxlength="80" value=""
onfocus="document.NDFDgenForm.endTime.value = &#39;2010-
01-01T00:00:00&#39;;">2010-01-01T00:00:00</TD>
        </TR>
      </TBODY></TABLE>
    </OL>  </DIV>
</FORM>
```

Figure 8: hRESTs in RDFa example

4.5 Examples of semantic links

OGC standards like GML (Portele 2007) define the use of XLink to add annotations in XML files. These annotations can point to extra sources of information (e.g. a file) or to Uniform Resource Name (URN).

The first use case is described in the GML specification as “composition by inclusion of remote resources”: in this case, the XLink annotation use the xlink:href attribute to reference an external file containing additional data (Figure 9).

```
<component name="weatherStation"
xlink:href="http://vast.uah.edu/downloads/sensorML/v1.0/examples/
sensors/DavisWeather/DavisMonitorII-WeatherStation.xml"/>
```

Figure 9: XLink used in SensorML to include extra data

Transposing this example to RDFa requires the inclusion of an annotation which identifies the concept in a repository of sensor descriptions with an about attribute: the URI would then point to an individual or instance (Figure 10) providing access to the data to be included.

```
<component
name="weatherStation"about="http://vast.uah.edu/downloads/sensor
ML/v1.0/examples/sensors/DavisWeather/DavisMonitorII-
WeatherStation"/>
```

Figure 10: RDFa example: additional data

The second use case corresponds to the inclusion of a “model reference to an ontological description”. In this case, the XLink annotation use the xlink:arcrole attribute to define the type of the referenced object (Figure 11). The definition attribute in the SWE schemas and the descriptionReference in the GML schemas are scoped for this particular usage.

```
<member xlink:arcrole="urn:ogc:def:process:OGC:SensorInstance">
```

Figure 11: XLink used in SensorML to define a type

In RDFa, the typeof attribute can be used for the same purpose (Figure 12). A URN pointing to a type definition (or class) is then used

```
<member typeof=="urn:ogc:def:process:OGC:SensorInstance">
```

Figure 12: RDFa example: reference to ontological def.

The example above shows that the current use of Xlink in OGC schemas can be mirrored in RDFa.

In our generalised mashup approach, the semantic annotations should be exploitable by generic or user-defined lifting operators to create the corresponding RDF statements. When this RDF is lowered back into XML, there is a risk of losing some of the information previously available. XLink can be used to maintain some of this lowered content. Table 7 defines the mappings between the two approaches which are possible with the present XLink specification. It also shows that there are other usages which are possible in RDFa but not in the “simple” style of XLink.

5 Directions for future work

5.1 Guidelines for the application of hRESTs

For RESTful services, the format of the HTML content which should be annotated is not specified by the proposed specifications. This is an issue which should be addressed. The form-embedded annotation approach is preferable to the description-based one in general for the part of the description which describes how to run the service, because the annotated form can still be used to test that the service works. For the part of the description which covers the output data (results and error messages), a different approach is required, to be based on an embedded XML schema (this is what WADL does) or on another form of testable content.

5.2 SAWSDL vs. hRESTs in RDFa

The relative complexity and rigidness of the SAWSDL and of the hRESTs Microformat specification contrasts with the flexibility of the approaches based on RDFa (e.g. hRESTs in RDFa), where the choice of the service ontology can be made by the end user without requiring any new developments for the lifting of the semantic annotations into semantic web tools.

This extra flexibility is important not just for RESTful services. Further work is required to upgrade SAWSDL so that it can also let the end user select the service ontology they want if they are not satisfied by the

definitions brought by the SA-REST or WSMO-Lite ontologies.

5.3 Ontologies for other types of services

Other service description languages like WADL (Hadley 2009) and WSDL 2.0[20] may provide a better basis for RESTful services. The hybrid ontology and rule-base framework proposed by Zhao and Doshi (2009) handles three categories of composable RESTful services to add access and transform resources.

SensorML (Botts and Robin 2007) is an OGC-developed markup language for the description of sensors. It includes a process model which is comparable to the other service ontologies discussed above. The challenge for the W3C Semantic Sensor Network Incubator Activity is to develop an ontology describing sensor services based on SensorML and use it for semantic annotations in a context where the boundary between the application-specific ontologies and the service ontologies and between non-semantic and semantic mashups is harder to define.

5.4 Replacement of custom-made lifting scripts

Any solution requiring the development of custom-made lifting mechanisms should be avoided if alternative approaches based on standards which fully specify this critical step like RDFa are available. The dependency on user-developed transformations for the lifting scripts is one of the factors which have slowed down the adoption of semantic annotation standards for services like SAWSDL and hRESTs/SA-REST/MicroWSMO.

As discussed above, the hRESTs in RDFa format provides a generic approach for the transformation of the semantic annotations into a RDF-based format and it should be possible to develop a similar approach for SAWSDL and to also suppress the requirement to develop custom-made scripts for this purpose.

But, it is not yet possible to automatically derive the lifting script for the second type of lifting operation discussed in 3.4, where the script goal is to process the XML data manipulated by the service and not the file containing the annotations. The MyMobileWeb project (Berrueta et al. 2009) has been looking at RDFa for a similar problem, to describe the bindings to data sources and enable multi-device mobile access to semantically enriched information portals.

5.5 Controlled upgrade of legacy standards

Ad hoc semantic upgrade of legacy standards such as XLink should be monitored closely to minimise the risks of failure caused by problematic extensions by end users.

In many cases, techniques bound to one family of standards (XML) have been later adapted to a different context without any assurance that the new usage respects the original intent of the specification. Hybrid ad hoc approaches may also import conflicting or ambiguous definitions from different standard families.

Some parts of SensorML uses XLink annotations to embed "model reference to an ontological description" in the sensor description (e.g. swe:phenomenon). These use cases are a possible source of confusion because they answer to requirements which can potentially be better addressed through new approaches based on semantic web technologies.

For example, to handle all the annotations requirements identified for RDFa in an XML context, a simple approach would be to add a new "style" to XLink for RDFa as an extension to the current XLink specification. For organisations like OGC who already use XLink and maintain a large number of XML schemas, this approach would have two advantages.

- To limit the impact on existing schemas to changes in the XLink schema,
- To provide a mechanism to isolate semantic XLink snippets from normal ones.

This upgrade of XLink should not be done without a careful consideration of the present usage of XLink in OGC standards and also in other standards like SVG[21].

5.6 Failure risk analysis

Combining legacy and opportunistic mashups will require robust and mashable validation tools to prevent and diagnose failures. Opportunistic mashups depends on external resources which may disappear or evolve without notice, especially mashable services and semantic resources, so the risks of failure are greater and more diverse than in other environments.

In a multi-layered mashup environment, it is important to support validation at every possible step of integration and to leverage the validation methods which are specific to each family of standards: XML, HTML and RDF individually. In this context, it is very important to check the availability of validators and their ability to check the content (markup validators) as well as the added annotations or links to remote resources and also the flexibility and robustness of these tools.

The Unicorn[22] (Universal Conformance Observation and Report Notation) project at the W3C is a *validator mashup* combining a HTML validator, a CSS validator and a HTML link checker. Extending this approach to the other families of the W3C[23] and OGC standards used in the type of mashups discussed above would be very useful.

6 Conclusion

There are multiple semantic enablement techniques which can be used in geospatial and semantic standards for legacy and opportunistic mashups. For the insertion of semantics links in XML content formatted according to OGC standards, the less disruptive approach identified in this review may be to add a new style to the existing XLink specification transposing all the RDFa attributes and processing rules defined for the HTML context.

The hRESTs-in-RDFa annotation format is preferred for the annotation of RESTful services. The arguments

[20] http://www.w3.org/TR/wsdl20/

[21] http://www.w3.org/Graphics/SVG/

[22] http://www.w3.org/QA/Tools/Unicorn/

[23] W3C specifications and validators are listed in http://www.w3.org/QA/TheMatrix

formerly raised (Graf 2007) to prefer Microformats to RDFa to add semantic annotations or links to HTML have been invalidated by the W3C decision to make RDFa available in HTML5. The analysis presented above shows that solutions based on Microformats prevent the implementation of generic lifting services with scripting languages such as XSL Transformations, XQuery or XSPARQL or with javascript libraries like rdfQuery which plays an essential role in opportunistic mashups.

The SAWSDL specification should also be upgraded to offer the same possibility for the user to select the service ontology.

Finally, in complex mashups, the risk of failure is greater and the validation methods are different for standards belonging to the XML, HTML and RDF families. There should be a limited number of methods to combine these standards together to lower the cost of development of new markup validators and link checkers. If possible, these new validation services should also be mashable to simplify the creation of more integrated validation services.

7 References

Adida, B., Birbeck, M., McCarron, S. and Pemberton, S. (2008): RDFa in XHTML: Syntax and Processing A collection of attributes and processing rules for extending XHTML to support RDF W3C Recommendation 14 October 2008, Available from http://www.w3.org/TR/rdfa-syntax, Accessed 17 Sep 2009.

Akthar, W., Kopecky, J., Krennwallner, T. and Polleres, A. (2008): XSPARQL: Traveling between the XML and RDF worlds – and avoiding the XSLT pilgrimage. *Proc. of 5th European Semantic Web Conference, ESWC 2008*, Tenerife, Spain, LNCS **5021**:432-447. Springer.

Al-Muhammed, M. J., Embley, D. W.,Liddle, S. W., and Tijerino, Y. A, (2007): Bringing Web Principles to Services: Ontology-BasedWeb Services. *Proc. of IEEE Congress on Services (Services 2007)*, 73-80.

Benjamins, V.R., Davies, J., Baeza-Yates, R., Mika, P. Zaragoza, H., Greaves, M., Gómez-Pérez, J.M., Contreras, J., Domingue, J. and Fensel D. (2008): Near-Term Prospects for Semantic Technologies, *IEEE Intelligent Systems*, **23**(1):76-88, Jan./Feb. 2008.

Berrueta, D., Polo, L., Fernández, S., Cantera, J. M. and Jiménez M. (2009): MyMobileWeb Deliverable D.5.4.1 Semantic extensions for IDEAL, 20 February, 2009, Available from http://forge.morfeo-project.org/wiki_en/index.php/Semantic_annotations_for_IDEAL, Accessed 17 Sep 2009

Birbeck, M. and McCarron, S. (2009): CURIE Syntax 1.0 A syntax for expressing Compact URIs W3C Candidate Recommendation 16 January 2009, Available from http://www.w3.org/TR/curie, Accessed 17 Sep 2009

Bizer, C., Heath, T., Ayers, D. and Raimond, Y. (2007): Interlinking open data on the web. *Proc. 4th European Semantic Web Conference*, Innsbruck, Austria.

Botts, M. and Robin, A. (2007): OpenGIS Sensor Model Language (SensorML), OGC 07-000, Open Geospatial Consortium, July 2007, Available from http://www.opengeospatial.org/standards/sensorml, Accessed 17 Sep 2009

Clark, K. G., Feigenbaum, L. and Torres, E. (2008): SPARQL Protocol for RDF W3C Rec. 15 January 2008, Available from http://www.w3.org/TR/rdf-sparql-protocol/ Accessed 17 Sep 2009

Connolly, D. (2007): Gleaning Resource Descriptions from Dialects of Languages (GRDDL). W3C Rec., W3C, 11 September 2007, Available from http://www.w3.org/TR/grddl/, Accessed 17 Sep 2009.

DeRose, S., Maler, E. and Orchard, D. (2001): *XML Linking Language (XLink) Version 1.0* W3C Recommendation 27 June 2001, Available from http://www.w3.org/TR/xlink/, Accessed 17 Sep 2009.

Di Lorenzo, G., Hacid, H., Paik, H., and Benatallah, B. (2009): Data integration in mashups. *SIGMOD Rec.* **38**(1):59-66, Jun 2009.

Farrell, J. and Lausen, H. (2007): *Semantic Annotations for WSDL and XML Schema* W3C Rec., August 2007, Available from http://www.w3.org/TR/sawsdl/, Accessed 17 Sep 2009.

Gamble M. T. and Gamble R. (2008): Monoliths to Mashups: Increasing Opportunistic Assets, *IEEE Software*, **25**(6):71-79, Nov/Dec 2009.

Goel, K., Guha, R. V. and O. Hansson (2009): Introducing Rich Snippets, Google Webmaster Central Blog, 2009-05-12, Google, Available from http://googlewebmastercentral.blogspot.com/, Accessed 17 Sep 2009

Graf, A. (2007): RDFa v.s. Microformats DERI Technical Report 2007-04-10, Available from http://www.sti-innsbruck.at/results/publications/, Accessed 17 Sep 2009

Hadley, M.J. (2009): Web Application Description Language (WADL) Sun Microsystems Inc. February 2, 2009, Available from https://wadl.dev.java.net/, Accessed 17 Sep 2009.

Henson, C. A., Pschorr, J. K., Sheth, A. P. and Thirunarayan K. (2009): SemSOS: Semantic sensor Observation Service. *Proc. of International Symposium on Collaborative Technologies and Systems 2009*, 44-53.

Hickson, I. (2009): HTML5 Draft Standard, 3 November 2009, Available from http://whatwg.org/html5, Accessed 13 Nov 2009.

Jarrar, M. and Dikaiakos, M. D. (2009): A Data Mashup Language For The Data Web. *Proc. of Linked Data on the Web (LDOW2009) Workshop at WWW2009*, Madrid, Spain, ACM.

Kopecký J. (2007): Web Services Description Language (WSDL) Version 2.0: RDF Mapping W3C Working Group Note 26 June 2007, Available from http://www.w3.org/TR/wsdl20-rdf, Accessed 17 Sep 2009

Kopecký, J., Vitvar, T., Bournez, C. and Farrell, J. (2007): SAWSDL: Semantic Annotations for WSDL and XML Schema, *IEEE Internet Computing*, **11**(6):60-67, Nov./Dec. 2007.

Kopecký, J., Gomadam, K. and Vitvar, T. (2008): hRESTs: An HTML Microformat for Describing RESTful Web Services. *Proc. 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, wi-iat **1**:619-625.

Kopecký, J., Vitvar, T., Fensel, D. and Gomadam, K. (2009): D12v0.1 hRESTs & MicroWSMO CMS WG Working Draft 10 March 2009, Available from http://cms-wg.sti2.org/TR/d12/v0.1, Accessed 17 Sep 2009

Koschmider, A., Torres, V. and Pelechano, V. (2009): Elucidating the Mashup Hype: Definition, Challenges, Methodical Guide and Tools for Mashups. Proc. *of the 2nd Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web* in conjunction with the 18th International World Wide Web Conference, Madrid, Spain.

Lathem, J., Gomadam, K. and Sheth, A. (2007): SA-REST and (S)mashups: Adding Semantics to RESTful Services. Proc. *IEEE Int'l Conf. Semantic Computing*, 469–476, IEEE CS Press.

Lemmens, R., de By, R., Gould, M., Wytzisk, A., Granell, C., and van Oosterom, P. (2007): Enhancing Geo-Service Chaining through Deep Service Descriptions, *Transactions in GIS*, **11**(6):849-871, December 2007

Le Phuoc, D., Polleres, A., Morbidoni, C., Hauswirth, M. and Tummarello, G. (2009): Rapid semantic web mashup development through semantic web pipes. *Proc. of the 18th World Wide Web Conference (WWW2009)*, Madrid, Spain.

Le Phuoc, D. and Hauswirth, M. (2009): Linked open data in sensor data mashups. *Proc. of the 2nd International Workshop on Semantic Sensor Networks (SSN09)*, Chantilly, VA, USA, CEUR-WS Proceedings **522**.

Portele C. (2007): OpenGIS® *Geography Markup Language (GML) Encoding Standard* version 3.2.1 OGC 07-036 Open Geospatial Consortium 2007-08-27

Prud'hommeaux, E and Seaborne, A. (2008): SPARQL Query Language for RDF W3C Rec. 15 January 2008, Available from http://www.w3.org/TR/rdf-sparql-query/, Accessed 17 Sep 2009

Sahoo, S. S., Sheth A. and Henson, C. (2008): Semantic Provenance for eScience: Managing the Deluge of Scientific Data, *IEEE Internet Computing*, **12**(4):46-54, July/Aug. 2008.

Sheth, A. P., Gomadam, K. and Lathem, J. (2007), "SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups," IEEE Internet Computing, **11**(6):91-94, Nov./Dec. 2007.

Vaccari, L. Shvaiko, P. and Marchese, M. (2009): A geo-service semantic integration in Spatial Data Infrastructures. *International Journal of Spatial Data Infrastructures Research (IJSDIR)*, **4**:24-51.

Zhao, H. and Doshi, P. (2009): Toward Automated RESTful Web Service Compositions. *Proc. of the 2009 IEEE International Conference on Web Services (ICWS 2009)*, 189-196, IEEE Computer Society.