Bridging the gap between semantic and pragmatic

Philippe Mathieu, Jean-Christophe Routier and Yann Secq
LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE, UMR CNRS 8022,
UNIVERSITÉ DES SCIENCES ET TECHNOLOGIES DE LILLE, FRANCE
Email: {Philippe.Mathieu, Jean-Christophe.Routier, Yann.Secq}@lifl.fr

***Abstract*— This paper is a pragmatic study of the use of knowledge representation languages, and more precisely the OWL language proposed by the W3C for the so-called *Semantic Web*, within *Agent Communication Languages*. This study is focused on the integration of OWL has a content language for the FIPA-ACL language, and the benefits it could bring to multi-agent platforms designers.**

The first part of this paper introduces the RDF and OWL languages, then a light introduction to *Agent Communication Languages* is provided. The second part presents the minimal generic agent model that we develop, and describe the possible uses of the OWL language within this model and for multi-agent systems. The last part identifies some conceptual and technological challenges that multi-agent platform designers are faced to when trying to integrate these technologies.

INTRODUCTION

There is a lot of hype around the *Semantic Web* nowadays : on the one hand supporters are describing a radically new experience in the way we and software agents use the Web, while on the other hand skeptics are pointing difficulties and challenges that have to be addressed before something useful could appear (or even if something at all could appear for the more skeptical). Knowledge representation theories and languages are also gaining momentum, and the support the W3C is providing to languages like *Resource Description Framework*[5] (RDF), and more recently with the *Ontology Web Language* [14] (OWL, formerly DAML+OIL) through its Web-Ontology Working Group [1], is a major step towards a widespread use of knowledge related technologies.

On another area, the growth of personal computing and the availability of the Internet have raised new issues on the design of large scale distributed systems. Several approaches have been proposed to handle this new complexity, ranging from parallel computing (with PVM or MPI) to distributed object technologies (like *Remote Method Invocations* or CORBA) and more recently grid computing[10]. Another approach that has rapidly been growing during the last decade are multi-agent system (MAS). A fair amount of publications can be found on theorical aspects, and toolkits or multi-agent platforms are numerous. They are now used in industrial context, and the standardization process initiated by the FIPA organization[2] is gaining momentum.

This growth has led to several proposals of agent methodologies[15], [4], [19], [8] to ease the analysis and design of such complex distributed systems. These methodologies often make reference to knowledge and linguistic theories : ontologies are used to add a semantic layer to content languages, the *Speech Act Theory* introduced by Austin[1] adds illocutary acts and knowledge representation to model agent beliefs. However, there is a gap between theoretical works that emphasized their use and agent toolkits that generally rely on weaker languages to be usable. Moreover, the multiplicity of agent models and frameworks makes it difficult to capitalize experiences, that are often not easily usable on another platform than the one used to design them.

This paper is an attempt to make a bridge between knowledge representation technologies and multi-agent platforms design in order to bring a pragmatic and usable agent platform. The first part of this paper introduces the RDF and OWL languages, then a light introduction to *Agent Communication Languages* is provided. The second part presents our minimal generic agent model that relies on an incremental creation of agents by "teaching" them *skills*, and describes the possible uses of the OWL language within this model and for multi-agent systems. We think that the use of OWL could have an important impact on the engineering of multi-agent systems and more prospectively on their reliability. Finally, the last part identifies some conceptual and technological challenges that multi-agent platform designers are faced to when trying to integrate these technologies.

I. SEMANTIC WEB AND AGENT COMMUNICATION LANGUAGES

A. Knowledge representation and the Semantic Web

The knowledge representation field is gaining momentum with the *Semantic Web* activity initiative that is supported by the W3C. This initiative aims at bringing more semantic in the *World Wide Web*, allowing smart searches by relying on inference engines. More precisely, works that have been done on languages like RDF, *Resource Description Framework*[5], DAML+OIL[7] and recently with the Working Draft of OWL, *Ontology Web Language* (formerly DAML+OIL), show a strong trend towards a broadening of knowledge representation use in everyday Internet technologies. Nevertheless, these languages are still in there infancy, and are seldom available in commercial products.

RDF is meant to describe relation between resources and to add meta-data to web resources in such a way that software agents could handle them. As everything can be accessed

[1]Semantic Web@W3C : `http://www.w3c.org/2001/sw/WebOnt/`
[2]Foundation for Intelligent Physical Agent :`http://www.fipa.org`

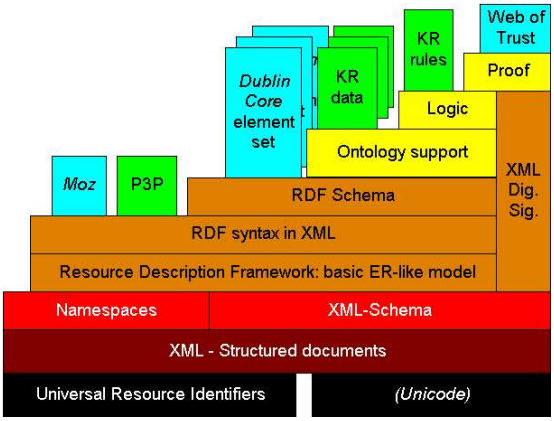

Fig. 1. Stack of XML technologies for Semantic Web

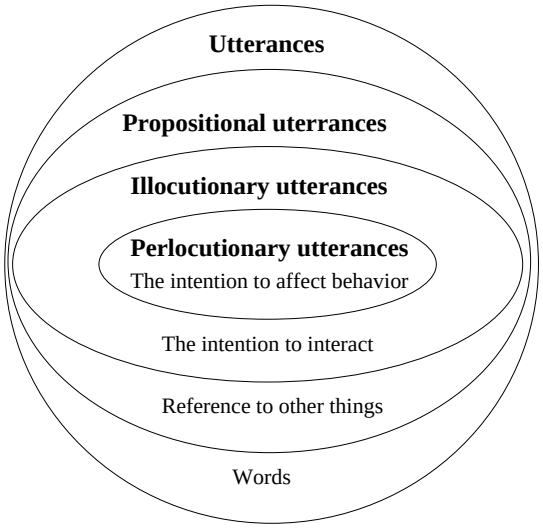

Fig. 2. The four categories of speech act of Searle

through an URI[3], RDF is targeted to augment the Web by describing the semantic of resources. The figure 1 is an excerpt of Tim Berners-Lee talk cited above that illustrates the various layers of the *Semantic Web*. The lowest layer is RDF, above OWL can be found in the *Ontology support* level. OWL provides a sharp semantic that allows first order logic inferences. Particularly, works done on description logics have produced a kind of algorithms[2] that enable complex inferences to be done in reasonable delays. OWL is the successor of DAML+OIL, which was initially composed by DAML, *Darpa Agent Markup Language*, and OIL, *Ontology Inference Layer*. Thus, the OWL language, despite the strong hype that surrounds it and the tendency to associate it only with *World Wide Web* applications, is particularly fitted to multi-agent systems. In the following section, we will describe the approach that the multi-agent field has taken to enable interoperability between heterogeneous systems through the use of *Agent Communication Languages* (ACL). And more particularly, we will explain why OWL can be used as a content language for these ACL.

B. Agent Communication Languages

Despite the lack of definition of *what is an agent*, an interesting way has been taken in the multi-agent community to provide MAS interoperability : *Agent Communication Languages*. The idea behind ACL is to provide interoperability through the exchange of messages that have a defined semantic. This idea is coming from the linguistic domain, and more precisely from the *Speech Act Theory* [1], which argues that talking is acting. More precisely John Searle[18] has identified four basic categories of speech acts : utterances, propositional utterances, illocutionary utterances and perlocutionary utterances. These categories are not independent and should be seen as building blocks. The figure 2 illustrates these categories. Several propositions have been done to implement and standardize *Agent Communication Languages*. The first attempt is the pioneer ACL : KQML[9]. A KQML message consist in :

- the sender agent,
- the intended agent to whom it was addressed,
- the reply to the message that the sending agent needs to receive,
- the performative name (25 are predefined),
- the language used to specify the content,
- the ontology that describes the meaning of the message (i.e. what it is trying to achieve),
- the message content.

The same structure has been used by the FIPA organization, when they defined the FIPA-ACL. Even if these languages are interesting, they are of little use in open environments. This is mainly induced by the fact that the semantic of performatives is hardly warranted when working with heterogeneous platforms and also because of some poor choices for content languages (SL0). We will see in the next section how OWL could be used as a content language for KQML or its "successor", FIPA-ACL.

II. THE ROLE OF OWL WITHIN OUR AGENT MODEL

A. A minimal generic agent model

The basis of our model is on the one hand the interactive creation of agents[16], and on the other hand a search on the fundamental functionalities of agenthood. We are not interested in the description of the individual behavior of agents, but rather in the identification of functions that are sufficient and necessary to an agent. Indeed, the management of interactions, the knowledge management or the management of organizations, are not related to the agent model, but are intrinsic characteristics with the concept of agent. In our model, an agent is a container which can host skills. A skill is a coherent set of functionalities accessible through a neutral interface. This concept of skill is to be brought

[3]Directly or indirectly, see `http://www.w3.org/2000/Talks/0906-xmlweb-tbl/`

4	Applicative skills	Database access, graphical user interface ...
3	Agent model related skills	Inference engine, behavioral engine, ...
2	Agenthood skills	Knowledge base, conversation management, organizations management
1	Minimal system skills	Communication and skill management

Fig. 3. The four layer of our abstract agent model

closer to the concept of software component in object oriented technologies. Thus, an agent consists in a set of skills which carries out various parts of its behavior. We identified four layers which are characterized by the various levels of abstraction of functionalities that are proposed (figure 3). The first level corresponds to *system* skills, i.e. the minimal functionalities allowing to bootstrap an agent: the communication (emission/reception of messages) and the management of skills (dynamic acquisition/withdrawal of skills). The second level identifies *agent* skills: the knowledge base, media of interaction between skills and the place of knowledge representation, the management of interaction protocols and the management of organizations). The third level is related to skills that define the *agent model* skills (reactive, BDI...), while the last level represents purely *applicative* skills. Rather than skills carrying out these various levels, it is the functionalities that they represent which are fundamental: the management of the communications, just like the knowledge base can be implemented in different ways, but it is necessary to have these functions within the agent. Thus, the first and the second level characterize our generic minimal agent model. This model is generic with respect to the agent models that can be used, and minimal in the sense that it is not possible to remove one of the functionalities without losing a fundamental aspect of agenthood[16].

A skill is made of two parts: its *interface* and its *implementation*. The interface specifies the incoming and outgoing messages, while the implementation carries out the processing of these messages. This separation uncouples the specification from its realization, and thus makes it possible to have several implementations for a given interface. The interface of a skill is defined by a set of message patterns which it accepts and produces. These messages must be discriminated, it is thus necessary to type them :

$$\texttt{interface} := ((m_{in})+, (m_{out})*)* \quad \texttt{where } m_x = \texttt{message pattern}$$

The typing of message patterns can take several forms : a strong typing, which has the advantage of totally specifying the interfaces, while a weak typing offers more flexibility with regards to the interface evolution. Thus, if the content of messages are expressed in KIF or OWL, a strong typing will consist in an entire message checking, while a weak typing will only check it partially.

¿From an implementation point of view, our notion of skill is similar to the idea of Web Services : a neutral interface that can be implemented in several languages and component models. The component models that could be used are ranging from EJB, to CORBA components, or even OSGI bundles for constrained environments.

B. Integrating knowledge representation technologies at the core of agent platforms

In our first framework, MAGIQUE[17], agents exchange semantically weak messages. These messages can be viewed as a kind of remote method invocation. Skill interfaces are basically Java interfaces, and their implementations are Java objects or components. So, we wanted to add some XML-based language to describe our skill interfaces to get rid off the Java language dependency. We studied existing approaches, and we found that WSDL[4] was the closer technological solution. But this language is finally just an XML-encoding of our previous approach, and we wanted more expressiveness. So, we took a closer look to RDF, and rapidly to DAML+OIL. The latter unifies works from different communities : it has a formal semantic and efficient reasoning (thanks to Description Logics), it provides rich modeling primitives (frame-like concepts), and a standard syntactical exchange format (RDF triples).

Our first use of OWL is thus to define skill interfaces (we were also attentive with DAML-S[6], but this initiative does not seem to grow). This enables us to add meta-information to ease the management of skills interfaces or implementations : version number, libraries dependencies, deployment information ... Indeed, the use of DAML+OIL, or its succesor OWL, represents a shift from object processing to document processing, and inference facilities can be seen as powerful information accessors.

The second use that we consider is related to the agent knowledge base. As agents exchange semantically strong messages, being able to use the same tool to represent knowledge could ease agent developer task. Implementing agents requires message matching, knowledge base querying and updating and message creation to reply. If the same language is used through all these stages, some translations can be avoided. Moreover, OWL has all the features needed to describe rich knowledge structures (it was designed for this aim), but can also eases the transition from object technologies thanks to the inclusion of datatypes. The other aspect, we consider is to use the knowledge base as a media for local (intra-agent) skill interactions. For that purpose, we propose to use information in the knowledge base as a kind of semantic linda-space[11].

Going further, having OWL as a core component of agent platforms could yield to more prospective aspects like advanced integrated development environment that could leverage the semantic layer, or even facilitate the use of agents platforms for model-based experimentations. Indeed, the semantic description of skills could be used in development environment

[4]Web Service Description Language : `http://www.w3.org/TR/wsdl12/`

as enhanced technical documentation, a kind of semantic "Skilldoc" (in analogy to the Javadoc, an automated project documentation framework). Model-based programming aims at developing sophisticated regulatory and immune systems that accurately and robustly control their internal functions[5]. To accomplish this, these systems exploit a vast nervous system of sensors, to model themselves and their environment, that enables them to reconfigure themselves. A tight coupling between the higher level coordination function provided by symbolic reasoning, and the lower level processes of adaptive estimation and control is thus necessary. Working with agents that are built on semantically strong descriptions, and relying on inference engine, could ease the design of such systems : one of the agent skill could monitor others and react if one fails.

III. Conceptual and technological challenges

Nevertheless, even if the integration of knowledge representation would be really useful for multi-agent software engineers, this task is really challenging. The first challenge is the novelty of these technologies and the lack of tools to ease their integration within existent systems. While RDF is now widely supported, DAML+OIL support is just beginning. Several editors are available : a mode for EMACS, which do not provide more than syntax highlighting, OILED[3], which is defined by their creators as an ontology "notepad", PROTÉGÉ [12], an ontology and knowledge-base editor that should integrate a DAML+OIL plugin soon, and some commercial tools. The main problems are the lack of coherence checking or querying facilities in these editors (even if OILED can rely on the FACT reasoner), and the lack of embeddable components of such tools. An exception should be noted, the Java Theorem Prover[6] is a nice API that is portable and easily embeddable. It is likely that the situation will be better when OWL will become a W3C Recommendation. Another possibility is the OWL-LITE language, which is a subset of OWL : it would be easier to create tools that support it, and this availability of tools could ease the widespread use of OWL.

To leverage the use of these technologies, the FIPA organization could provide OWL ontologies within some of its specifications. Technological choices are fundamental for industry adoption. For example, the use of SL as a content language and IIOP as a transport layer have been wrong choices : relying on an XML-based language like DAML+OIL and HTTP for transport would have ease the development of libraries, tools and applications around FIPA specifications. A nice initiative towards this aim is the Java Agent Services[7] project, held under the Java Community Process, which implements the FIPA Abstract Architecture[8] and provides a nice object-oriented API that could leverage works done on agents infrastructures. Sadly, since this project has gone under Public Review, it seems to be stalled.

A last challenge, more cultural, is that knowledge representation technologies and particularly ontology design and development is not an easy task. And because of the novelty of DAML+OIL and OWL, resources like "how-to" or tutorials are quite scarce. This last point is very important, and the knowledge representation community can play an important pedagogic role to "evangelize" the multi-agent community, and more precisely platform designers.

Conclusion

This paper is an attempt to make a bridge between knowledge representation technologies and multi-agent platforms design in order to bring a pragmatic and usable agent platform. We believe that knowledge representation technologies should be core components of multi-agent platforms. After introducing our agent model that relies on the notion of skill, we have identified some aspects that could benefit from the use of OWL : skill interfaces definition, knowledge bases implementation, ACL content language.

However, there are several showstoppers that have to be addressed before knowledge representation technologies could be seamlessly integrated as core components of agent platforms. Some of these problems are related to the novelty of the language, and should disappear with the development of tools, but some others are deeper because they induce the learning of a new way to think. Indeed, creating and managing ontologies is not an easy task, and is not the same as decomposing a problem in objects. Moreover, the introduction of knowledge representation within object oriented systems means that a clear distinction should be established between data/knowledge (represented with OWL) and tasks/processes (defined in an object oriented language).

Nevertheless, we believe that the OWL language could play an important role in the agent software engineering field. We are working on an implementation of an agent platform relying on our agent model, using OWL for skill interface definition and OSGI[13] components for their implementation.

[5]Model-based computing at Xerox : `http://www2.parc.com/spl/projects/mbc/`

[6]JTP site : `http://www.ksl.stanford.edu/software/JTP/`

[7]JAS homepage : `http://www.java-agent.org`

[8]FIPA Abstract Architecture specification : `http://www.fipa.org/specs/fipa00001/SC00001L.pdf`

References

[1] J. L. Austin. *How To Do Things With Words*. Harvard University Press, second edition edition, 1975.

[2] F. Baader and U. Sattler. Tableau algorithms for description logics. In R. Dyckhoff, editor, *Proceedings of the International Conference on Automated Reasoning with Tableaux and Related Methods (Tableaux 2000)*, volume 1847, pages 1–18, St Andrews, Scotland, UK, 2000. Springer-Verlag.

[3] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: A reason-able ontology editor for the semantic Web. *Lecture Notes in Computer Science*, 2174:396–??, 2001.

[4] F. M. T. Brazier, B. M. Dunin-Keplicz, N. R. Jennings, and J. Treur. DESIRE: Modelling multi-agent systems in a compositional formal framework. *Int Journal of Cooperative Information Systems*, 6(1):67–94, 1997.

[5] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel C. A. Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.

[6] A. Ankolekar et al. Daml-s: Web service description for the semantic web, proc. 1st international semantic web conf. (iswc 02), 2002.

[7] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. Oil: An ontology infrastructure for the semantic web, 2001.
[8] J. Ferber and O. Gutknecht. Operational semantics of a role-based agent architecture. In *Proceedings of ATAL'99*, jan 1999.
[9] T. Finin, R. Fritzson, D. McKay, and R. McEntire. Kqml as an agent communication language. In *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463, Gaithersburg, Maryland, 1994. ACM Press.
[10] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.
[11] D. Gelernter. Multiple tuple spaces in linda. In E. Odijk, M. Rem, and J.-C. Syre, editors, *PARLE '89: Parallel Architectures and Languages Europe*, volume 366 of *Lecture Notes in Computer Science*, pages 20–27, 1989.
[12] W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu, and M. Musen. Knowledge modeling at the millennium – the design and evolution of protege, 2000.
[13] R.S Hall H. Cervantes. Beanome : A component model for the osgi framework. In *Workshop on Software Infrastructures for Component-Based Applications on Consumer Devices*, September 2002.
[14] Ian Horrocks, Dieter Fensel, Jeen Broekstra, Stefan Decker, Michael Erdmann, Carole Goble, Frank van Harmelen, Michel Klein, Steffen Staab, Rudi Studer, and Enrico Motta. OIL: The Ontology Inference Layer. Technical Report IR-479, Vrije Universiteit Amsterdam, Faculty of Sciences, September 2000. See http://www.ontoknowledge.org/oil/.
[15] E. A. Kendall, M. T. Malkoun, and C. H. Jiang. A methodology for developing agent based systems. In Chengqi Zhang and Dickson Lukose, editors, *First Australian Workshop on Distributed Artificial Intelligence*, Canberra, Australia, 1995.
[16] P. Mathieu, J.C. Routier, and Y. Secq. Dynamic skill learning: A support to agent evolution. In *Proceedings of the AISB'01 Symposium on Adaptive Agents and Multi-Agent Systems*, pages 25–32, 2001.
[17] JC. Routier and P. Mathieu. A multi-agent approach to co-operative work. In *Proceedings of the CADUI'02 Conference*, to appear in April 2002.
[18] J Searle. *Speech Acts: An Essay in the Philosophy of Language*. Harvard University Press, second edition edition, 1969.
[19] M. Wooldridge, NR. Jennings, and D. Kinny. The GAIA methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 2000.