Towards a pragmatic use of ontologies in multi-agent platforms

Philippe Mathieu, Jean-Christophe Routier, and Yann Secq

Laboratoire d'Informatique Fondamentale de Lille – CNRS UMR 8022
Université des Sciences et Technologies de Lille
59657 Villeneuve d'Ascq Cedex
{mathieu,routier,secq}@lifl.fr

Abstract The knowledge representation field is gaining momentum with the advent of the *Semantic Web* activity within the W3C. This working group, thanks to previous researches, has proposed the *Ontology Web Language* to enhance the expressivity of web pages and to allow semantic inferences. This paper argues that knowledge representation technologies should be core components of multi-agent platforms.
In the first part of this paper, we introduce our agent model that relies on the notion of skill. Then, we identify several criteria that let us to believe that the Owl language should be used as a content language within *Agent Communication Languages* and also in the design of multi-agent platforms. In the last part, we discuss the conceptual and technological challenges that platform designers coming from the multi-agent field have to deal with when trying to integrate knowledge representation technologies.

1 Introduction

The multi-agent systems field has rapidly been growing during the last decade. They are now used in industrial context, and the standardization process initiated by the Fipa organization[1] is gaining momentum. This growth has led to several proposals of agent methodologies[13,3,19,7] to ease the analysis and design of such complex distributed systems. These methodologies often make reference to knowledge and linguistic theories : ontologies to add a semantic layer to content languages, *Speech Act Theory*[1] to handle illocutary acts and knowledge representation to model agent beliefs[16]. Sadly, even if these intentions are faithful, lots of agents frameworks do not use all these features and rely instead on lower level approaches that have the advantage to be practical. Moreover, the multiplicity of agent models and frameworks makes it difficult to capitalize experiences, that are often not easily usable on another platform that the one used to design them.

The knowledge representation field is also gaining momentum with the so called *Semantic Web*[2] initiative supported by the W3C. More precisely, works that have been done on languages like Rdf[4](Resource Description Framework), Daml+Oil[6] and recently with the Working Draft of Owl, *Ontology Web Language*, 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.

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

This paper is a prospection on the use of knowledge representation languages, and more precisely OWL, as a core component of agents communication and agent platforms. We first introduce our agent model, which tries to define several levels of abstraction in agent design and that relies on generic components called *skills*. Then, we argue that the OWL language should be used to add a semantic layer at different levels within agents. This layer could have an important impact on the engineering of multi-agent systems and more prospectively on their reliability. Finally, we set out the difficulties and challenges that agents platform designer have to deal with, from a conceptual and technological point of view.

2 A generic agent model relying on the notion of skill

The basis of our model is on the one hand the interactive creation of agent[17], and on the other hand a search on the fundamental functionalities of agenthood. We are not interested in the description of the individual behaviour 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 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 behaviour. We have identified four layers which are characterized by the level of abstraction of the functionalities that are proposed :

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

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 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.

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 :

$$\texttt{interface := ((m}_{in}\texttt{)+, (m}_{out}\texttt{)*)* where m}_{x}\texttt{ = message pattern}$$

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 regard to the interface evolution. Thus, if the content of messages are expressed in KIF or OWL, a strong typing will consist of an entire message checking, while a weak typing will only check it partially. An analogy can be done between this approach and the propositions done to type XML messages : XML SCHEMA induce a full verification, while Schematron does partial checking[3]. 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 to realize skills are ranging from *Enterprise Java Beans* from SUN, to the *Corba Component Model* from the OMG, or even *OSGi* bundles[11] for constrained environments.

Skill interfaces are only concerned with incoming and outgoing messages, but this approach brings the problem of the semantic of messages. The exchange of messages within multi-agent systems is one of the main principle. It was early identified with works from Hewitt[12] on *Actor Languages* and has been more recently used as a mean to enable agents interoperability. This last approach has been implemented with the KQML[8] language during the last decade. This language was built using notions from the *Speech Act Theory*, introduced by Austin[1] and developed by Searle[18], which claims that talking is acting. Thus, Searle identified four categories of speech acts : utterances, propositional utterances, illocutionary utterances and perlocutionary utterances. These notions have been interpreted and reduced to the following key elements that constitutes a KQML message : 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) and finally the message content. Nowadays, KQML is being slowly replaced by the FIPA-ACL which retains the same principles.

Although this approach is really appealing, it is not practical : developers always have to agree on content languages and to manually interpret the ontology (ie. the semantic) of messages[4]. We believe that a leap forward could be achieved if the FIPA foundation proposed general purpose ontologies specified in OWL. It could also remove incoherence that emerges

[3] For an interesting comparison of XML schema languages see [14].

[4] We make reference here to heterogeneous agent platforms. Lots of studies have been done and are working with homogeneous environments.

if the all the information on messages were expressed with the same language : the envelope, payload and message[15].

3 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[5] 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 (through Description Logics), it provides rich modelling 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[5], but this initiative does not seem to grow). This enable us to add meta-information to ease the management of skill interfaces or implementations : version number, libraries dependencies, deployment information ... Indeed, the use of DAML+OIL or its successor OWL represents a shift from object processing to document processing (see figure 1), and inference facilities can be seen as powerful information accessors.

Object Oriented
Simple data structures :"hat type", "quantity", "list of colors" and "price". **Tight coupling** : Replies are meaningless without the context of the question. **Synchronous communication** : Because of the tight coupling, I must be able to associate replies with requests. Also, my subsequent requests often depend on a previous reply.
Document Oriented
Complex, structured, self-describing data : I received an entire hat catalog. I knew it was a catalog by reading it, not by knowing that it was a response to my request. **Loose coupling** : I did not need to directly associate the catalog and my previous request because the catalog is sufficiently self-describing to anyone who knows how to read a catalog (understands the schema). **Asynchronous communication**: Enabled by the self-describing data and loose coupling.

Figure 1. Object oriented versus Document oriented paradigm.

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

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

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 ease 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[9]. This approach would induce a real uncoupling were dependencies would only be expressed through the semantic of data, and again inferences would be much more expressive than pattern matching (semantic versus syntax).

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. Who has never been wandering through the thousand of classes available in the standard library of Java ? Indeed, the semantic description of skills could be used in development environment as enhanced technical documentation, a kind of semantic "Skilldoc" (in analogy to the Javadoc, automated project documentation framework). Model-based programming aims at developing sophisticated regulatory and immune systems that accurately and robustly control their internal functions[6]. To accomplish this, these systems exploit a vast nervous system of sensors, to model themselves and their environment. It enables them to reconfigure themselves. A tight coupling between the higher level coordination functions 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.

4 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 (and moreover for OWL which is not yet standardized). Several editors are available : a mode for *emacs*, which do not provide more than syntax highlighting, *OILed*[2], which is defined by their creators as an ontology "notepad", PROTÉGÉ[10], 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*[7] 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 will probably be

[6] Model-based computing at Xerox : `http://www2.parc.com/spl/projects/mbc/`
[7] JTP website : `http://www.ksl.stanford.edu/software/JTP/`

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*[8] project, held under the *Java Community Process*, which implements the FIPA Abstract Architecture[9], and provide 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 "evangelise" the multi-agent community, and more precisely platform designers.

5 Conclusion

In this paper, we have studied the role that knowledge representation technologies could play in the design of open multi-agent platforms. We insist on the fact that using KR languages within agent-based applications is not new, but their use for open systems has not really worked yet. The advent of the *Semantic Web*, and more precisely the transition from DAML+OIL to the Working Draft of OWL, could enable a wider use of KR technologies within the WWW, but will also leverage standardization on ontology languages.

This fact should deeply impact multi-agent technologies : instead of esoteric languages like KIF and SL0 or dedicated low-level languages (ie. Java objects), *Agent Communication Languages* could finally enable real interoperability through an agreement on OWL as a content language. An analogy could be drawn with the advent of XML and the impact it has induced in server-side environments. We believe that with OWL as a W3C standard, the FIPA organization should follow and provide basic ontologies for agent and platform services (instead of informal frame-based ones, which are not usable without being first interpreted by developers).

We have identified several points, where the use of Owl would be useful in agent-based systems : as a content language within ACL, as a knowledge representation language wihtin agent knowledge base, as semantically stronger description for skill interfaces. These are direct applications, but we also raise more prospective ones like a kind of semantical technical documentation, and the facilities that could be used to add model-based notions within

[8] JAS website : `http://www.java-agent.org`.
[9] FIPA Abstract Architecture : `http://www.fipa.org/specs/fipa00001/SC00001L.pdf`

agent-based systems to enhance their reliability. Unfortunately, the use of these technologies is challenging for several reasons : technologies are new so there is not a lot of tools available, and ontology design and management is not an easy task. Nevertheless, we believe that the OWL language will play an important role in the agent software engineering field.

References

1. J. L. Austin. *How To Do Things With Words*. Harvard University Press, second edition edition, 1975.
2. 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.
3. 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.
4. 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.
5. A. Ankolekar et al. Daml-s: Web service description for the semantic web, proc. 1st international semantic web conf. (iswc 02), 2002.
6. D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. Patel-Schneider. Oil: An ontology infrastructure for the semantic web, 2001.
7. J. Ferber and O. Gutknecht. Operational semantics of a role-based agent architecture. In *Proceedings of ATAL'99*, jan 1999.
8. 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.
9. 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.
10. 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.
11. 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.
12. C. Hewitt. Viewing control structures as patterns of passing messages. In *Artificial Intelligence: An MIT Perspective*. MIT Press, Cambridge, Massachusetts, 1979.
13. 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.
14. Dongwon Lee and Wesley W. Chu. Comparative analysis of six XML schema languages. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(3):76–87, 2000.
15. Torsten Illmann Michael Schalk, Thorsten Liebig and Frank Kargl. Combining fipa acl with daml+oil - a case study. In *Proceedings of the Workshop on Ontologies in Agent Systems,1st International Joint Conference on Autonomous Agents and Multi-Agent Systems, Bologna (Italy)*, 2002.
16. A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.
17. JC. Routier, P. Mathieu, 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.
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.