DUNEDIN NEW ZEALAND

Extending Agent Messaging to Enable OO Information Exchange

Stephen Cranefield
Martin Purvis

The Information Science Discussion Paper Series

Number 2000/07
April 2000
ISSN 1177-455X

University of Otago

Department of Information Science

The Department of Information Science is one of six departments that make up the Division of Commerce at the University of Otago. The department offers courses of study leading to a major in Information Science within the BCom, BA and BSc degrees. In addition to undergraduate teaching, the department is also strongly involved in postgraduate research programmes leading to MCom, MA, MSc and PhD degrees. Research projects in spatial information processing, connectionist-based information systems, software engineering and software development, information engineering and database, software metrics, distributed information systems, multimedia information systems and information systems security are particularly well supported.

The views expressed in this paper are not necessarily those of the department as a whole. The accuracy of the information presented in this paper is the sole responsibility of the authors.

Copyright

Correspondence

This paper represents work to date and may not necessarily form the basis for the authors' final conclusions relating to this topic. It is likely, however, that the paper will appear in some form in a journal or in conference proceedings in the near future. The authors would be pleased to receive correspondence in connection with any of the issues raised in this paper, or for subsequent publication details. Please write directly to the authors at the address provided below. (Details of final journal/conference publication venues for these papers are also provided on the Department's publications web pages: http://www.otago.ac.nz/informationscience/pubs/publications.html). Any other correspondence concerning the Series should be sent to the DPS Coordinator.

Department of Information Science
University of Otago
P O Box 56
Dunedin
NEW ZEALAND

Fax: +64 3 479 8311
email: dps@infoscience.otago.ac.nz
www: http://www.otago.ac.nz/informationscience/

Extending Agent Messaging to Enable OO Information Exchange

Stephen Cranefield and Martin Purvis
Department of Information Science
University of Otago
PO Box 56, Dunedin, New Zealand
E-mail: {scranefield, mpurvis}@infoscience.otago.ac.nz

Abstract

It is canonical practice in agent-based systems to use a declarative format for the exchange of information. The increasing usage and facility of object-oriented tools and techniques, however, suggests there may be benefits in combining the use of object-oriented modelling approaches with agent-based messaging. In this paper we outline our efforts in connection with the New Zealand Distributed Information Systems project to use object-oriented knowledge representation in an agent-based architecture. Issues and tradeoffs are discussed, as well as the possible extensions to current agent-based message protocols that may be necessary in order to support object-oriented information exchange.

1 Introduction

For the representation of knowledge within agent-based systems, the general tendency has been to favour declarative representations, such as the Knowledge Interchange Format (KIF), the Foundation for Intelligent Physical Agents (FIPA) Semantic Language (SL), or other logic-based representations [NCITS, 1998; FIPA, 1999b]. In fact the agent-based community has adopted *de facto* standard protocols that are based on the declarative approach. It is our contention, however, that existing declarative approaches are not the only possibility and that with some suitable adjustments, object-oriented knowledge representation can be incorporated into agent-based information systems.

In order to examine how object-oriented notions might relate to the existing, preferred declarative representation of knowledge, we first consider the perceived benefits of the declarative approach. Declarative knowledge is represented as a static collection of assertions or facts, such as might be collected in a database, and it does not contain details about the use to which the knowledge might be put [Shapiro, 1987]. More generally, Genesereth and Nilsson have defined declarative knowledge representations to be any "explicit representations of knowledge ... that can be interpreted as making declarative statements. We call knowledge represented in this way declarative knowledge, because it is contained in declarations about the world." [Genesereth and Nilsson, 1987]. This has been contrasted with the procedural representation of knowledge, in which "facts are represented in terms of programs and data structures" [Shapiro, 1987, p. 884]. Without going into the details of the "declarative vs. procedural controversy" [Hayes, 1977; Moore, 1982; Winograd, 1980], we may summarise the relative advantages of the declarative perspective [Genesereth and Nilsson, 1987]:

1. It is straightforward to access declarative knowledge by means of introspective programs or by means of direct visual inspection.
2. Declarative knowledge may be changed more easily, since a change is likely to be local to a specific record of the data store.
3. Declarative knowledge can be used for purposes that were not necessarily anticipated when the knowledge was assembled.
4. Declarative knowledge can be extended by employing reasoning processes that derive additional knowledge.

These generally accepted advantages have led the agent-based software engineering community to set up standards, such as KQML and the FIPA ACL that presume a declarative knowledge representation for agent message communication content. In association with declarative agent communication content has been the adoption of declarative knowledge-based formalisms, such as KIF and the KL-ONE style knowledge representation languages [Brachman and Schmolze, 1985], for representing ontologies. Although this approach has been able to exploit the recognised advantages of declarative knowledge listed above, the representations have entailed the adoption of large and relatively complex systems, e.g. LOOM [ISI, 1999] and description logics [Donini *et al.*, 1996; Owsnicki-Klewe, 1990], for encoding and reasoning about the information. For applications such as distributed information retrieval that do not necessarily require the deductive reasoning capabilities of systems such as LOOM, we believe that there can be significant advantages for adopting an overall object-oriented modelling approach. In this paper we examine some of the object-oriented knowledge representation schemes that we are using in connection with the New Zealand Distributed Information Systems project [Purvis *et al.*, 2000], and we discuss some of the resulting implications in the context of existing agent-based communication protocols that have been based on a declarative approach.

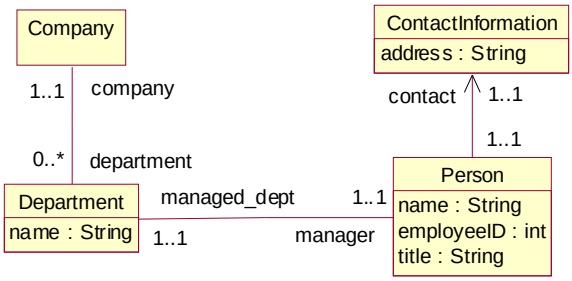

Figure 1: A simple ontology expressed as a UML class diagram

2 Object-Oriented Knowledge

For the purposes of representing various aspects of the world in information systems, the general benefits of object-oriented (OO) modelling have been discussed widely [Booch, 1994]. In particular, object-oriented modelling is perceived to have the following benefits:

- the representation of physically-situated objects that encompass state and behaviour corresponds to the common way in which the physical world is organised by human perception and is expressed in everyday language;
- the notions of class inheritance and object instantiation also correspond to natural cognitive classification schemes that have been traditionally used to manage the representation of complexity in the real world;
- the existence of object-oriented programming language compilers and software development tools facilitates the direct mapping of object-oriented modelling into operational computer representations of these models.

We view these benefits to be sufficiently compelling to justify the adoption of

- the Unified Modelling Language (UML) [Booch *et al.*, 1999] from the Object Management Group (OMG) [OMG, 1999b], together with its associated Object Constraint Language (OCL) [Warmer and Kleppe, 1998] for the representation of ontologies [Cranefield and Purvis, 1999] (see Figure 1 for a simple example),
- the OMG's Meta-Object Facility (MOF) [OMG, 1999a] as a repository for the storage of metainformation, including ontologies, and
- the use of object-oriented knowledge encodings within the content of agent messages.

For the specific case of UML for ontology representation, we can identify the following benefits:

- UML has a large and expanding community of users. Users of distributed information systems will be increasingly familiar with UML notation, as opposed to that of KIF or description logics.
- Unlike traditional declarative formalisms, there is a standard graphical representation for models expressed in UML. The OMG is also in the process of adopting XMI (XML Model Interchange) as a standard for stream-based model interchange [DSTC, 1999].

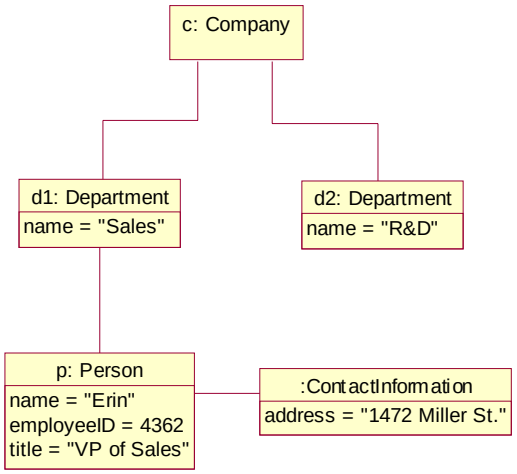

Figure 2: A UML object diagram (adapted from [Booch *et al.*, 1999, p. 196])

Moreover, the adoption of an OO knowledge representation approach does not necessarily involve the loss of the benefits associated with declarative knowledge representation. Consider again the four benefits listed in Section 1:

1. The existence of standard graphical representations for OO models means that this information is also accessible for direct visual inspection. In fact for complex structures, the standard graphical representation may be more illuminating than a declarative one.
2. The modular aspect of object-oriented modelling means that such models are also easily modified. Changes to such models are likely to be local, and the visual representation of the model facilitates changes to the model.
3. OO models can also be used and extended for purposes that were not anticipated at the time of model creation.
4. The emergence of object-oriented deductive databases [Kifer *et al.*, 1995] offers the promise of reasoning mechanisms that can be directly used to extend object-oriented knowledge.

From this perspective we suggest that object-oriented knowledge representation has the fundamental beneficial features of declarative knowledge.

3 Object Diagrams as Declarative Knowledge

Object-oriented data structures can be viewed as encodings of UML object diagrams, which can be considered to be to declarative representations of knowledge. For example, consider the object diagram in Figure 2. This encodes knowledge about the objects c, $d1$, $d2$ and p, such as the information that object p has the value "Erin" for its name attribute and p is the manager of object $d1$.

If two agents represent domain knowledge as object diagrams, it would be most convenient for them to communicate information to each other in this form as well. Note that we do not propose that arbitrary object-oriented data structures should appear within agent messages. On the contrary, an agent coded in Java (for example) should not be able to send an arbitrary Java object to another agent within an ACL message. A fundamental principle of agent communication languages is that a message should be able to be understood without knowledge of the agent that sent it. This is where ontologies are crucial: the content of a message must be expressed according to an ontology that defines the semantics of that content. This remains true for object-oriented message content—only object data encoded according to some advertised ontology should be sent within messages.

How should object-oriented information be sent to another agent, and in particular, how could this be done within a FIPA agent system? In this section we discuss possible avenues towards a solution.

3.1 An Ontology for Object Diagrams

Content languages to be used with the FIPA ACL are required to be able to express propositions, objects and actions, where the notion of an object is not necessarily required to be equivalent to an object in object-oriented programming, rather it can be any type of construct that represents an "identifiable thing" in the domain of discourse. A FIPA `inform` message must have a content field that represents a *proposition*, thus an agent cannot be simply "informed" of information in the form of an object diagram. One solution to this problem is to define a (propositional) ontology for object diagrams that contains a unary predicate `true_object_diagram`. If an agent sends a proposition `true_object_diagram`(o) to another agent (where o is an encoding of an object diagram), this would indicate that the information represented within the diagram is asserted to hold.

3.2 Propositional Encoding of Object Diagrams

Defining an ontology for object diagrams will require a theoretical account of how an object diagram can be interpreted as a conjunction of propositions. Deductive object-oriented languages such as F-logic [Kifer *et al.*, 1995] should provide a sound basis for this endeavour. These logics typically include atomic formulae to express that an object is a member of a class and that a given attribute of an object attribute has a certain value. There are also formulae that express schema information (such as the fact that a class has a given attribute—we will not require these, as it is assumed that agents will have this information available already via ontologies). Liu [1999] includes a discussion of the features of a number of such languages. To represent UML object diagrams adequately, however, it may be necessary to select (or design) a language that reflects UML's explicit distinction between attribute values and links (instances of class associations). In particular, this may be necessary to account for instances of n-ary associations.

3.3 Defining a New Content Language

In the `true_object_diagram` proposition above, an object diagram appeared as an argument. This situation will arise not only when an object diagram represents a set of propositions about object attribute values and the relationships between objects, but also when an object diagram is used to represent an object in the domain. For example, an agent a that represents its plans as object-oriented structures may communicate its current plan p to another agent using a proposition of the form: `current_plan`(a, p). Representing message content such as this will require a new content language that supports both propositions and the full notion of an object from object-oriented programming. The abstract syntax of this content language will need to be able to encode object diagrams (including references to the ontologies for these diagrams). To define a string-based encoding for this language, some structured format such as XML will be required.

An agent platform that supports the use of such a content language may also offer the ability via its Agent Communication Channel (ACC) to translate to and from this language. An agent local to the platform, that only knows this "object-enhanced" content language, could still offer its services to external agents if they were able to express an object diagram using a conjunction of propositions as discussed in Section 3.2. The ACC could then translate the incoming message into a form in which the object diagram is explicitly represented.

3.4 Extending FIPA Semantics for `inform`

The previous section presented an example of an agent communicating its current plan to another agent, with the object-oriented plan appearing as the argument to the `current_plan` predicate. However, this solution is rather unsatisfactory as there is no fundamental need to mix propositional and object-oriented representations here. The solution suggested above requires the `current_plan` predicate to be declared in some (propositional) ontology that must also include the concept of an agent (which appears as the first argument of the predicate). On the other hand, this ontology could be represented in an object-oriented manner with a class Agent having an association with a class Plan. Information about an agent a's current plan can now be fully encoded as an object diagram depicting an agent named a linked to a plan object. It seems a natural extension of the FIPA ACL to allow such an object diagram to be sent to another agent as the content of an `inform` message, particularly if there were a standard account of how the object diagram corresponds to a conjunction of propositions as discussed in Section 3.2.

4 Object-Oriented Encodings of ACL Message Content

The minimal message transport mechanism specified for the FIPA ACL is a sequence of ASCII characters delivered over a byte stream. The syntax of the ACL is defined in terms of this character representation (although the FIPA 99 Seventh Call for Proposals [FIPA, 1999a] called for an abstract syntax for the ACL to be included in the FIPA '99 specification, due to be finalised in October 1999). The FIPA Agent Message Transport Specification [FIPA, 1999b] states that agents within an agent platform are free to use a proprietary transport mechanism to transport messages, provided that the Agent Communication Channel

(ACC) can communicate with external agents using the standard ASCII message encoding and the IIOP message transport protocol. The ACC within a given agent platform provides a Message Transport Service (MTS) that can be used by agents within that platform to communicate with other agents, both internal and external to the platform.

Sending and receiving string-based messages either requires individual agents to contain string-processing code to analyse the structure of a message and extract the required information from the surrounding syntax, or (more generally) the ACC must provide parsers for ACL and all supported content languages, as well as (ideally) support for pattern-matching on the resulting structures to conveniently extract information from incoming messages. In order to interoperate with other agent platforms, there is no escaping the need to support the string-based message encoding. However, the parsing problem can be alleviated by using a more structured textual representation of object-oriented information, in particular the Extensible Markup Language (XML). Also, within an agent platform a higher-level interface to the MTS can be used that corresponds more directly to the abstract syntax of the ACL and content languages than string encodings. This section discusses these issues, and the techniques considered are summarised in Figure 4.

4.1 From UML Ontologies to XML Schemas

The benefits of using XML for encoding structured information have been widely touted. With the inclusion of the XLink and XPointer languages [Cover, 1999], XML documents are capable of representing arbitrary networks of objects connected by references. The use of an XML schema declaration language such as the Document Type Definition (DTD) grammar allows an XML document to be represented and parsed in terms of domain-specific structures. Once an XML document has been received by an application, there are two main approaches to parsing it: event-based application programmer interfaces (APIs) such as SAX (Simple API for XML) and APIs based on complete in-memory parse trees, such as W3C's Document Object Model (DOM). These are relatively low-level APIs and in an attempt to allow programs to "manipulate XML content ... at the same conceptual level as the content itself" [Reinhold, 1999a], Sun have issued a Java Specification Request for an XML data-binding facility for Java [Reinhold, 1999b]. Given the schema for an XML document (the choice of schema language has not yet been made), a command-line tool would produce a set of classes representing the elements of the XML schema. A marshalling framework would support the marshalling and unmarshalling of XML documents into graphs of interrelated objects. The following discussion focuses on the encoding of object-oriented information within the content of a FIPA ACL message as an XML string. However, work is also underway within FIPA to define an XML representation for FIPA ACL itself.

The likely future existence of technologies such as the XML data-binding facility for Java will provide a convenient method for agent systems developed using object-oriented languages such as Java to process XML content within ACL messages. However, we must also find a technique for encoding object-oriented information, modelled

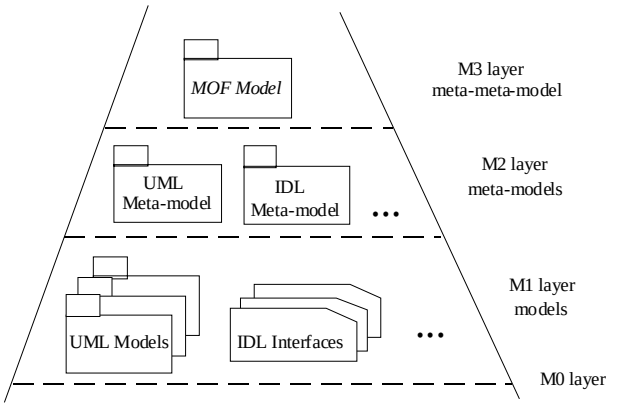

Figure 3: MOF Metadata Architecture (from [OMG, 1999a])

according to some ontology expressed in UML, as an XML document together with an appropriate XML schema. One way to do this is provided by the OMG's XML Model Interchange (XMI) specification. This is associated with the OMG Meta Object Facility (MOF) which defines a "meta-meta-model" for defining modelling languages such as UML. In the MOF terminology (see Figure 3), a modelling language such as UML is a meta-model, allowing models (e.g. ontologies) to be defined. XMI specifies how a model stored in a MOF-based repository can be represented as an XML document that is encoded in terms of a DTD that is generated from the MOF meta-meta-model definition of the modelling language (the meta-model) used. This does not quite solve our problem, however, because the information we wish to exchange between agents is usually not at the ontological level (the "M1 layer" that XMI deals with), but actually represents instances of the concepts defined in the ontology (the "M0 layer"). We can get around this problem by noting that UML includes object diagrams as well as class diagrams, and therefore information about objects can be represented at the M1 layer that XMI is designed to encode. However, the resulting XML document will be very cumbersome to parse as the associated DTD will describe the structure of UML models in general (at the M2 layer) instead of being specialised for a particular ontology (the M1 layer). For example, an object will be represented not by a single DTD element having the name of the object's class, but instead by an Object element containing references to other elements in the XML document: e.g. a Classifier element representing the object's class, and attributeLink elements, each of which in turn will reference Attribute and Instance elements (representing the attribute's identity and value).

One solution to this problem is to encode ontologies directly using the MOF model, rather than the more expressive UML. Another solution has been presented by Skogan [1999] to support the interchange of geographic information, although his work only addresses a limited subset of UML class diagrams (incorporating packages, classes, attributes and associations but not generalisation relationships). With this approach, three models are defined: an abstract schema model (a subset of UML's metamodel), an instance model (a "minimal model for structuring and

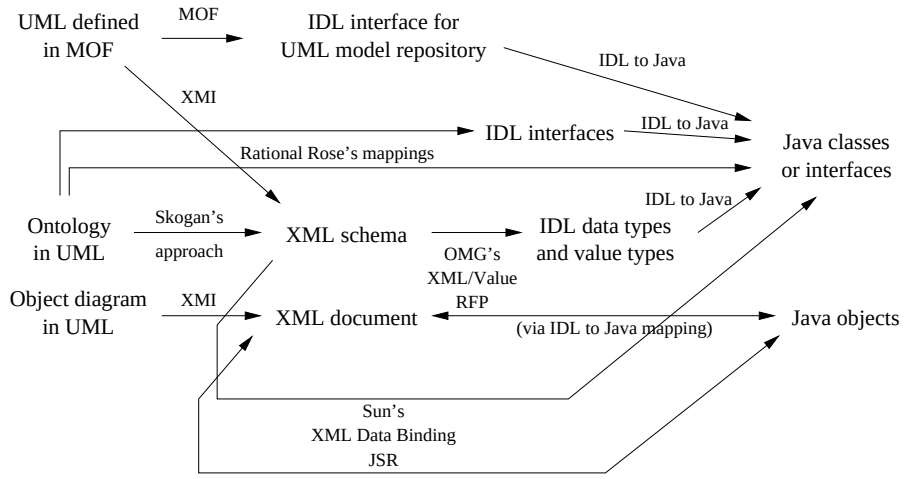

Figure 4: Possible mappings for including object diagrams within message content

representing data") and an ouput data structure—a hierarchical structure composed of XML elements. Two sets of conversion rules are then defined: one to map instances of the schema model to DTDs and another to map instances of the instance model into instances of the output data structure, which can then be encoded as an XML file. This results in an XML file with a structure that directly corresponds to the schema used to encode the data. To apply this technique to ontologies, the schema model, and possibly the instance model and output data structure, would need to be extended to account for a larger subset of UML, with corresponding extensions to the mapping rule templates also required. However, this approach is worthy of further investigation.

4.2 IDL and Java Bindings for UML Ontologies

Although an appropriately structured XML document can directly encode object-oriented data, there is still overhead associated with transporting it (due to its verbose format) and parsing it. It would be more efficient for an agent platform that is built using an object-oriented technology such as CORBA or Java to provide an object-oriented interface to its Message Transport Service. Ideally such an MTS would be extensible so that interfaces corresponding to ontologies in UML could be used to construct IDL or Java representations of object diagrams conforming to those ontologies. It would also be useful if the MTS interface included options to support the transmission of objects either by value or reference. Just as the OMG's Meta Object Facility provides a solution to generating XML files (via XMI), the MOF defines an IDL interface for manipulating models expressed in UML, which includes operations that can be used to construct object diagrams. However, this approach suffers from the same shortcoming as the XMI approach: this interface is not specialised to any particular ontology, and would therefore be awkward to use.

Another approach to generating IDL interfaces corresponding to an ontology in UML is to use a specialised XML schema (e.g. generated via an extension of Skogan's approach) as an intermediate representation. The OMG have issued a Request for Proposals for a standard way of representing XML documents using IDL data types and value types [OMG, 1999c]. The RFP states that "CORBA 2.3 IDL ... can encode arbitrary graphs and has essentially the same expressive power of [sic] XML". When this technology turns from 'proposalware' into reality, this may be a useful approach to follow.

Finally, the object-oriented modelling tool Rational Rose (and possibly other similar tools) provides proprietary mappings from UML models to IDL and Java. Although it would be better to use a standard mapping, for the short term this may be the simplest approach for building an MTS interface that directly supports the construction of object-oriented message content (although IDL valuetypes, introduced in CORBA 2.3 to support the passing of objects by value, are not yet supported by Rose).

5 Conclusions

Both object-oriented and agent-based technologies are recognised to have practical advantages for the implementation of distributed information systems. There is, however, a perceived impedance mismatch between the notions of agents and objects: agents are assumed to represent and exchange information in terms of declarative propositions, which are not conventionally thought to be compatible with the structured nature of object-oriented information. This paper has discussed the practical issues that must be addressed in order to incorporate object-oriented content into agent-based ontologies and message structure. We first highlighted the fact that object-oriented knowledge representation technology now offers many of the positive features that have been associated with the propositional representations of existing agent-based systems. We then examined how some of the emerging object-oriented technical standards, such as XML, XMI, MOF, and mappings to IDL and Java, are beginning to provide the missing pieces that can be used to include object-oriented knowledge in agent message exchange. The New Zealand Distributed Information Systems Project is building an agent-based prototype that uses these techniques.

Acknowledgments

The NZDIS software is under ongoing development by Geoff Bush, Dan Carter, Bryce McKinlay, Mariusz Nowostawski and Roy Ward, and is funded by the New Zealand government's Public Good Science Fund. The ideas expressed in this paper have benefited, in particular, from fruitful discussions with Mariusz Nowostawski.

References

[Booch *et al.*, 1999] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

[Booch, 1994] G. Booch. *Object-oriented Analysis and Design with Applications*. Benjamin/Cummings, 2nd edition, 1994.

[Brachman and Schmolze, 1985] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, April 1985.

[Cover, 1999] R. Cover. XML linking and addressing languages. OASIS Web page at http://www.oasis-open.org/cover/xll.html, December 1999.

[Cranefield and Purvis, 1999] S. Cranefield and M. Purvis. UML as an ontology modelling language. In *Proceedings of the Workshop on Intelligent Information Integration, 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, 1999. http://nzdis.otago.ac.nz/download/papers/UML_Ontology_99.pdf.

[Donini *et al.*, 1996] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation and Reasoning*, Studies in Logic, Language and Information, pages 193–238. CLSI Publications, 1996.

[DSTC, 1999] XMI spec recommended. News item on Distributed Systems Technology Centre Web page at http://www.dstc.edu.au/Research/Projects/MOF/, January 1999.

[FIPA, 1999a] FIPA 99 seventh call for proposals. http://www.fipa.org/cfp/cfp7.html, April 1999.

[FIPA, 1999b] FIPA specification. On Foundation for Intelligent Physical Agents Web site at http://www.fipa.org/spec/index.html, 1999.

[Genesereth and Nilsson, 1987] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.

[Hayes, 1977] P. J. Hayes. In defence of logic. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 559–565. Morgan Kaufmann, 1977.

[ISI, 1999] Loom project home page, Information Sciences Institute. http://www.isi.edu/isd/LOOM/LOOM-HOME.html, 1999.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the Association for Computing Machinery*, 42(4):741–843, July 1995.

[Liu, 1999] M. Liu. Deductive database languages: Problems and solutions. *ACM Computing Surveys*, 31(1):27–62, March 1999.

[Moore, 1982] R. C. Moore. The role of logic in knowledge representation and commonsense reasoning. In *Proceedings of the 2nd National Conference on Artificial Intelligence (AAAI-82)*, pages 428–433, 1982.

[NCITS, 1998] Draft proposed American national standard for Knowledge Interchange Format. National Committee for Information Technology Standards. http://logic.stanford.edu/kif/dpans.html, 1998.

[OMG, 1999a] MOF specification. Object Management Group. http://www.omg.org/cgi-bin/doc?ad/99-09-04, 1999.

[OMG, 1999b] OMG homepage. Object Management Group. http://www.omg.org/, 1999.

[OMG, 1999c] XML/Value request for proposals. Object Management Group document orbos/99-08-20, http://www.omg.org/techprocess/meetings/schedule/XML_Value_RFP.html, 1999.

[Owsnicki-Klewe, 1990] B. Owsnicki-Klewe. A general characterisation of term description languages. In K.-H. Blasius, U. Hedtstuck, and C. Rollinger, editors, *Sorts and Types in Artificial Intelligence*, Lecture Notes in Artificial Intelligence 418, pages 183–189. Springer, 1990.

[Purvis *et al.*, 2000] M. Purvis, S. Cranefield, G. Bush, D. Carter, B. McKinlay, M. Nowostawski, and R. Ward. The NZDIS project: an agent-based distributed information systems architecture. In R.H. Sprague Jr., editor, *Proceedings of the Hawaii International Conference on System Sciences (HICSS-33)*. IEEE Computer Society Press (CDROM), 2000. http://nzdis.otago.ac.nz/download/papers/nzdis-project_1-00.pdf.

[Reinhold, 1999a] M. Reinhold. An XML data-binding facility for the Java platform. http://java.sun.com/xml/docs/bind.pdf, July 1999.

[Reinhold, 1999b] M. Reinhold. XML data binding specification. Java Specification Request JSR-000031, Sun Microsystems. http://java.sun.com/aboutJava/communityprocess/jsr/jsr_031_xmld.html, 1999.

[Shapiro, 1987] S. Shapiro, editor. *Encyclopedia of Artificial Intelligence*, volume 2. Wiley, 1987.

[Skogan, 1999] D. Skogan. UML as a schema language for XML based data interchange. In *Proceedings of the 2nd International Conference on The Unified Modeling Language (UML'99)*, 1999. http://www.ifi.uio.no/~davids/papers/Uml2Xml.pdf.

[Warmer and Kleppe, 1998] J. B. Warmer and A. G. Kleppe. *The Object Constraint Language: Precise Modeling With UML*. Addison-Wesley, 1998.

[Winograd, 1980] T. Winograd. Extended inference modes in reasonong by computer systems. *Artificial Intelligence*, 13(1,2):5–26, 1980.