# Is it an Ontology or an Abstract Syntax? Modelling Objects, Knowledge and Agent Messages

Stephen Cranefield, Martin Purvis and Mariusz Nowostawski
Department of Information Science
University of Otago
PO Box 56, Dunedin, New Zealand
Phone: +64 3 479 8142, Fax: +64 3 479 8311
E-mail: {scranefield, mpurvis, mnowostawski}@infoscience.otago.ac.nz

**Abstract**

This paper describes a system of interlinked ontologies to describe the concepts underlying FIPA agent communication. A meta-modelling approach is used to relate object-oriented domain ontologies and abstract models of agent communication and content languages and to describe them in a single framework. The modelling language used is the Unified Modeling Language, which is extended by adding the concepts of resource and reference. The resulting framework provides an elegant basis for the development of agent systems that combine object-oriented information representation with agent messaging protocols.

## 1 Introduction

This paper discusses issues relating to the use of ontologies in communication between software agents. In particular, we consider the implications of modelling the agents' domains of discourse using object-oriented ontologies and how we can integrate the use of object-oriented information with the Foundation for Intelligent Physical Agents (FIPA) [1] messaging framework. This leads us to consider the relationship between the different types of objects that may exist in an agent system at run-time (domain objects, knowledge objects and message objects) and how their respective models (ontologies, content languages and agent communication languages) are related and can be expressed in the same modelling framework.

Previously [2] we have argued for the use of the industry-standard object-oriented modelling language, the Unified Modeling Language (UML) [3], as a representation language for ontologies. The advantages of using UML include the now widely accepted belief that object-oriented modelling fits well with people's intuitive models of the world [4], the fact that UML has a very large and rapidly growing user community, and the standard graphical representation for models that it provides (there is also a standard linear representation defined by the XMI specification [5]). An example of a simple ontology expressed using UML appears in Figure 1.

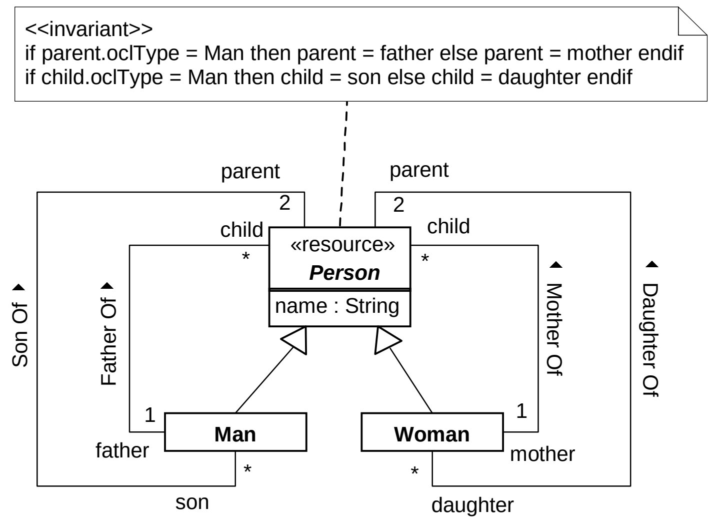
<<invariant>>
if parent.oclType = Man then parent = father else parent = mother endif
if child.oclType = Man then child = son else child = daughter endif
parent
parent
2
2
child
child
*
*
«resource»
Person
name : String
Son Of
Father Of
Mother Of
Daughter Of
1
1
Man
Woman
father
mother
*
*
son
daughter

Figure 1: An ontology describing family relationships

In order to understand the UML class diagrams in this paper, it is sufficient to know the following:

> Rectangles depict classes with the class name in the top section of the box (in italics in the case of an abstract class) and the attributes declared in a separate compartment (if applicable). A class name can be preceded by a 'stereotype': a word in French quotation marks (guillemets) indicating that the class has some special properties. UML defines a number of standard stereotypes; the one used in Figure 1 ("≪resource≫") is non-standard and will be explained in Section 3.
>
> Lines between classes represent association relationships, which may optionally be named, with a solid triangle indicating the direction in which the name should be read. A solid diamond shape may appear at one end of an association; this means that objects of the class beside the diamond symbol will own or contain objects at the other end of the association. Numbers at the end of an association indicate how many objects may have this association with instances of the class at the other end ('`*`' means "zero or more" and '`1..*`' means "one or more"). A solid line with a triangular arrowhead indicates a generalisation/specialisation relationship, with the arrow pointing to the more general class. A similar arrow with a dashed shaft represents the relationship of implementing an interface. Finally, the 'dog-eared' rectangle in Figure 1 represents a constraint on the class `Person`. This is expressed using the Object Constraint Language (OCL) [6] that can be used in conjunction with UML in order to constrain the possible models of a specification in ways that cannot be achieved using the UML structural elements alone. A similar rectangle can be used for notes attached to model objects. These give informal documentation about model elements.

Traditionally, software agent communication languages (ACLs) have been based on the exchange of information represented as sentences in a logic-based *content language* such as the Knowledge Interchange Language (KIF) [7]. The agent communication language has an outer layer that specifies information needed for routing the message, understanding the context and parsing the content of the message, as well as the type of the communicative act (e.g. ‘inform’ or ‘request’) represented by the message. The message’s *content* field is then used to store the details of the act as a string, which must be a well-formed formula in the content language used and must be parsed by the receiving agent.

The use of an object-oriented ontology representation language such as UML raises interesting questions about the form in which knowledge should be stored within an agent and encoded within inter-agent messages. In previous work [8], we have discussed several options for this, including the translation of a UML object diagram (or, more precisely, the implementation of one as a network of objects and inter-object references in a specific object-oriented programming language or database system) into a conjunctive expression in a logic that includes object-oriented concepts, such as those on which deductive object-oriented databases are based [9].

Another, more appealing, option involves regarding a UML object diagram itself as a declarative representation of knowledge, and finding ways of including it directly as the content of a message. Our supporting arguments for this view have been discussed previously [8]. Here, we simply state that this option is in line with a general trend away from defining languages in terms of linear string-based syntax, in favour of “abstract syntax” representations which can then be mapped onto various alternative concrete syntaxes. In particular, this allows the use of the Extensible Markup Language (XML) [10] as a transport format. Forthcoming XML-related technology (such as the XML Data Binding facility for Java [11]) will provide generic mechanisms for marshalling and unmarshalling information between XML documents and in-memory networks of objects that are instances of classes defined in the XML document’s schema[1]. This will free agent system developers from having to implement language-specific parsers and should make it easier to achieve interoperability between different agent deployment platforms.

The FIPA ACL is adopting an abstract syntax [15], although the new ACL specification approved in October 1999 has not been publicly released on the FIPA’99 specification Web site [16] at the time of writing. Also, the FIPA-RDF content language [17] is based on an abstract syntax defined as an RDF schema [18]. This suggests to us the following interesting scenario. We can regard our ontologies in UML as an abstract syntax for the domain of discourse. If the content language and ACL used are also defined by an abstract syntax using UML, then an entire agent message could be expressed as an object diagram containing nodes corresponding to two different types of objects: knowledge objects (the ‘content’ of the message) and message objects (the ‘wrapper’). In addition, the knowledge objects must refer to concepts from the ontologies. Furthermore, the existence of a UML-based abstract syntax for the ACL and content language will allow an agent message transport system to provide a direct binding to OO languages such as Java for the creation and receipt of messages.

[1]Unfortunately there is not yet a published technology for generating an XML schema specialised for a particular model in UML, although there has been work on generating XML schemas from ontologies expressed using languages less expressive than full UML [12, 13]. However, such a mapping is of great commercial and academic interest [14] and will undoubtedly be available soon.

This paper explores this and other consequences of the philosophical and architectural choice to use UML-based abstract syntax for an ACL and content language. In particular, the distinction between an ontology and an abstract syntax (which defines the concepts that can be expressed in a language) becomes blurred when three different types of run-time object (domain, knowledge and message) are modelled using the same formalism.

## 2 Messages as Object-Oriented Structures

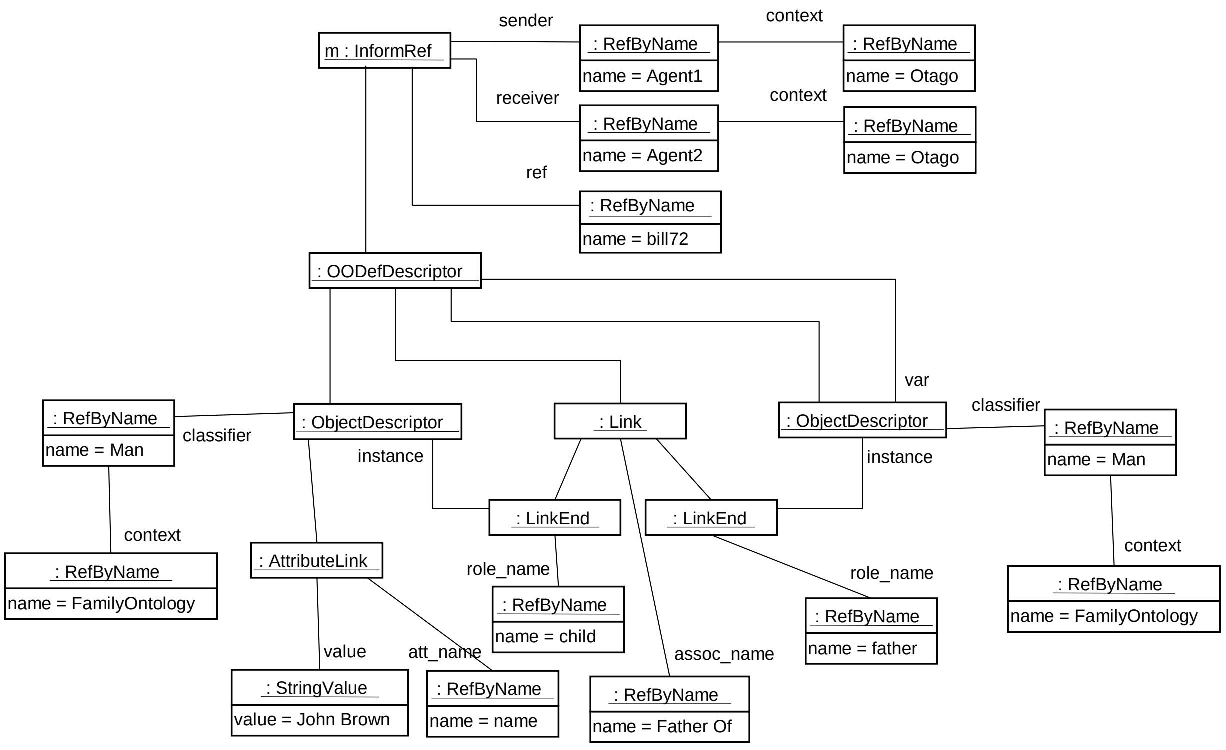

Figure 2: An example message expressed as a UML object diagram

Figure 2 shows an example of an agent message expressed as a UML object diagram. Ontologies describing the classes named in this diagram will appear later in this paper. Here we present this as an illustration of what we see as the natural arrival point when object-oriented modelling is introduced to the FIPA framework.

In an object diagram, rectangles denote objects, specifying their 'name' (corresponding to the name of a variable referencing the object, if there is one) and then (after a colon) the class of the object. The object's attribute values are also shown, and the lines between objects show 'links': instances of associations between classes.

This diagram can be considered to be an abstraction of an actual network of interlinked objects in some programming language. It shows a message object `m`, of a type that is specialised for informing an agent that the object corresponding to a given description (the object of type

`OODefDescriptor` below `m`) has a given reference (the object linked to `m` with the role `ref`). The classes appearing in this diagram will be defined in later diagrams. In particular, the class `OODefDescriptor` is proposed as a way of representing the properties of an object for which a reference is required. Its structure encodes an object diagram of object descriptors and links, and in addition it labels the object being asked about (the `var` role).

It may seem odd that concepts in the Family ontology only appear indirectly in this message using a 'reference by name' attached to an object descriptor at the 'classifier' end of a link ("classifier" is the UML name for the type of an object). However, this is no different from the traditional mode of passing knowledge as string-encoded propositions—in that case the receiving agent (or the knowledge representation system inside it) must parse the string to extract the names of the predicates used, which are considered to be standard references to the concepts they represent. Section 6 will discuss a way to provide a more direct encoding of knowledge about an ontology.

This diagram corresponds roughly to the following message in the standard FIPA s-expression syntax:

```
(inform
  :sender Agent1.otago
  :receiver Agent2.otago
  :ontology FamilyOntology
  :language FIPA-SL2
  :content (= (iota ?y (exists ?x (and (man ?x)
                                        (and (name ?x "John Brown")
                                             (father ?x ?y)))))
               bill72))
```

## 3 Objects and References

If a UML object diagram is chosen as the basis for knowledge representation and/or communication, it is important to avoid confusion between the objects being represented, and their descriptors: the objects comprising the object diagram. The objects that are the subject of inter-agent communication may be real world objects, Internet resources, or system objects that the agent wishes to access via a non-agent based protocol (e.g. CORBA's IIOP). However, after receiving a message describing a domain object, an agent may wish to directly access or contact that object, e.g. by downloading it if it is a Web document, sending it a CORBA message if it is a CORBA object, or engaging it in conversation via a speech interface if it is a person. This requires the agent to have some sort of reference to the object, and thus agents must have some way of requesting and communicating information about object references.

The FIPA ACL has adopted a mechanism for asking and informing agents about 'references' from the prior work of Sadek [19], based on the notion of a "denoting phrase" from the philosopher Bertrand Russell [20]. In brief, an agent can ask another agent to inform it of the reference corresponding to a *definite description*: a syntactic expression $\iota x \psi(x)$ that specifies a property $\psi$

that is assumed to hold of some unique but unknown object $x$ (this is the `iota` expression that appears in the example FIPA ACL message in the previous section). The reply to this request is supposed to be an `inform` message with the content $\iota x \psi(x) = n$ where $n$ is the 'standard name' for some object. It is not clear that this notion of a standard name represented by a constant in the content language will be adequate for dealing with object references in the full object-oriented systems sense. We leave that question for future research and in this paper take a meta-modelling approach to define the types of entities that ACLs and content languages need to denote.

First, we need to define what a reference is. We do this by extending the UML notion of a data type—a descriptor for a set of values that have no identity and cannot be altered. We define a Data Values ontology (Figure 3) that introduces an abstract class `DataValue` and assert that this is the abstract base class for all primitive types (UML has nothing to say about the primitive types available in a system being modelled, except that numbers and strings exist, so asserting that these share a common superclass is a harmless restriction on our model of the run-time environment of an agent). The class reference is then defined as a subclass of `DataValue`. Figure 3 also shows two examples of possible specific reference types.

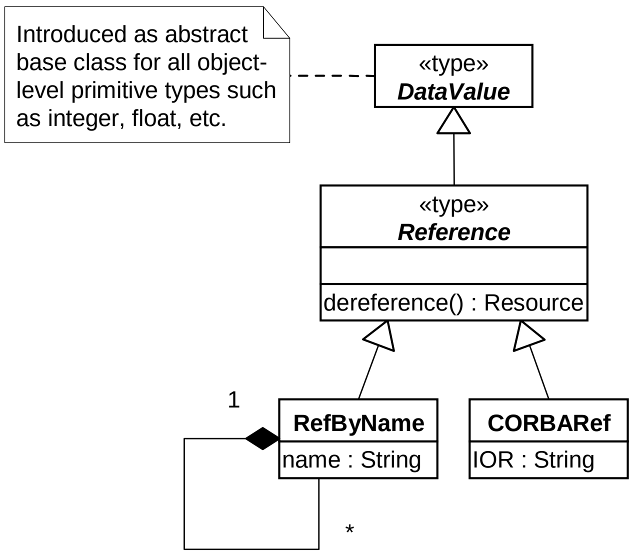
Introduced as abstract base class for all object-level primitive types such as integer, float, etc.
«type»
DataValue
«type»
Reference
dereference() : Resource
1
RefByName
name : String
CORBARef
IOR : String
*

Figure 3: The Data Values ontology with some possible reference types

Now that the notion of a reference has been introduced in a 'standard' ontology, our domain ontologies can include associations between domain classes and the `Reference` class to indicate that objects of those domain classes can have references. This is likely to be extremely common in ontologies for use with FIPA agent systems (where asking for information is expressed using the `query-ref` communicative act). Rather than require explicit modelling of these 'reference of' associations, we use UML's *stereotype* mechanism to define a specialised type of class, called `Resource` and declare that this has additional semantics as shown in Figure 4. This figure defines an extension to UML, and so it is a class diagram at the meta-modelling level rather than the ontology modelling level, i.e. it is an addition to the definition of UML (which happens to be expressed using UML itself).

The outcome of this extension is that classes in domain ontologies can now be annotated with the stereotype “≪resource≫” to indicate that objects of these classes have references (see Figure 1 for an example).

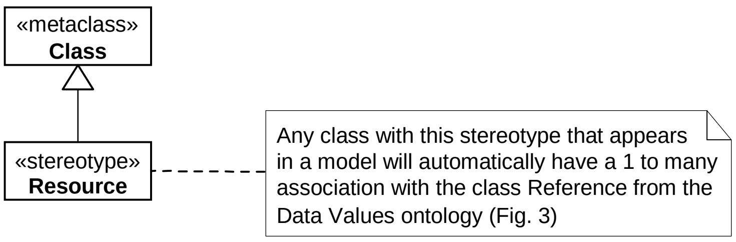
«metaclass»
Class
«stereotype»
Resource
Any class with this stereotype that appears in a model will automatically have a 1 to many association with the class Reference from the Data Values ontology (Fig. 3)

Figure 4: Incorporating resources into the UML meta-model

## 4 A Meta-Modelling View of Agent Systems

| | | | | |
|---|---|---|---|---|
| UML meta-model (extended with Resource stereotype - Fig. 4) | | | | Level 2 (meta-model) |
| Data values ontology (Fig. 3) | Domain ontologies (e.g. Fig. 1) | Content language ontologies (e.g. Fig. 7) | ACL ontologies (e.g. Fig. 6) | Level 1 (models) |
| Domain values (including references) | Domain objects and resources | Knowledge objects | Message objects (e.g. Fig. 2) | Level 0 (objects) |

Figure 5: A meta-modelling perspective on agent messaging

The following sections will present a UML-based abstract syntax for FIPA-like messages, and a related abstract syntax for a content language suitable for expressing information about a domain with an object-oriented ontology. Figure 5 shows how these models can be put in the same context as the standard type of ontology that models an application domain. The figure shows a ‘meta-modelling’ view of an agent system. The bottom layer (Level 0) depicts the concrete entities that exist in and around an executing agent. Above this, Level 1 shows the models that define the properties of the Level 0 instances: there is an “instance of” relationship between each object at Level 0 and some concept defined in Level 1. In particular, a domain object is an instance of a concept in an ontology, a knowledge object is an instance of (i.e. expression in) a content language and similarly a message object is an instance of an ACL model. Finally, Level 2 shows the meta-model used in this paper to describe the Level 1 models. This is the UML meta-model (the abstract definition of the modelling constructs in UML): each concept in a model at Level 1 is an instance of one of the UML modelling elements such as “Class”.

With this viewpoint, it can be seen that the notion of an ontology is strongly related to the notions of content and agent communication language abstract syntaxes—all appearing at the same level of the meta-modelling hierarchy—so we will use the term ontology to refer to all three.

It is interesting to note there are no agent objects appearing in Level 0. It seems that an agent is not a first class entity in current FIPA language models. Instead, the existence of agents is built into the semantics of the ACL (in which agents are modelled as having beliefs and intentions relating to the contents of messages they send and receive). Perhaps this problem could be solved by introducing conversation models into Level 1 of Figure 5, in which case the corresponding objects at Level 0 would include agents.

## 5 Modelling Messages and Message Content

Figure 6 shows an object-oriented ontology for agent messages, based on the FIPA ACL. The tabbed rectangle around the class `Reference` indicates that this is defined in another ontology: `Data Values` (Figure 3).

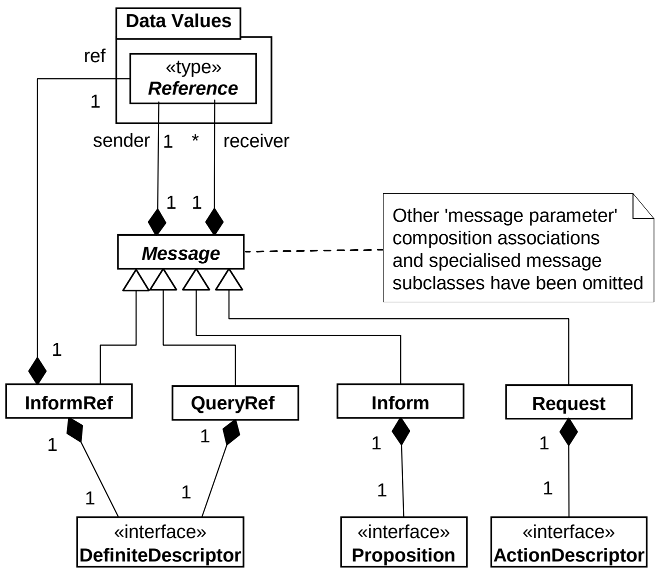

Figure 6: The FIPA Messaging ontology (defining a FIPA-like ACL)

The notions of definite descriptor, proposition and action descriptor are modelled as 'tag' interfaces (i.e. interfaces with no operations). This ontology places no structural requirements on the representations that a content language might use to express these concepts, it just declares that the concepts exist.

Note that this ontology includes a message class `InformRef`. This is not the same as the FIPA ACL `inform-ref` "macro action", which is a technical device that allows an agent to plan an informing action before it knows the actual reference that corresponds to the object to be identified (e.g. “the father of John Brown”). When the plan is executed, the actual message sent will be an `inform` action containing a proposition stating that the definite descriptor supplied is equal to a given object. However, requiring the use of the generic `inform` message type for this communication needlessly constrains the form of content languages that can be used within a FIPA ACL message. The messaging ontology above therefore adds a concrete `InformRef` message class that separately identifies the definite descriptor and the reference instead of requiring the content language to relate these within a proposition.

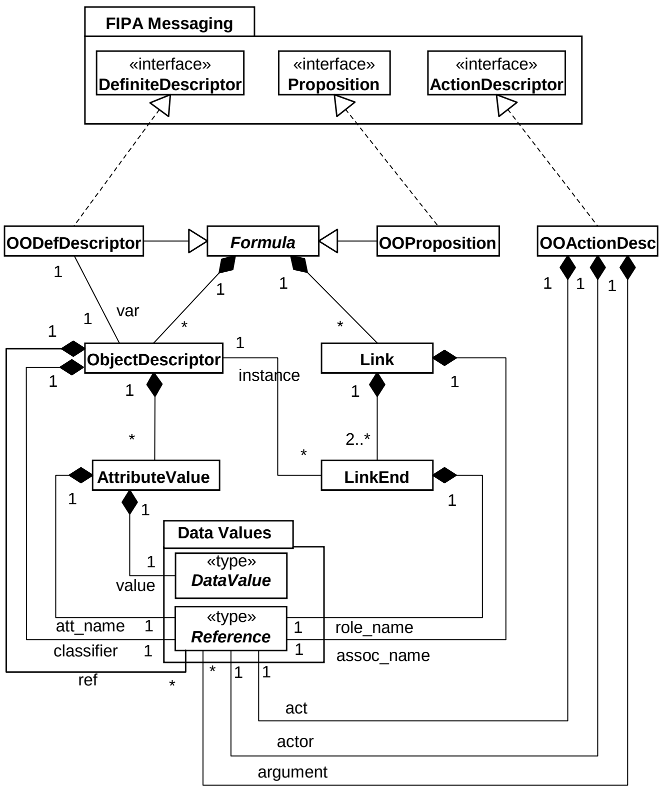

Figure 7: An ontology for an object-oriented content language

Figure 7 shows an object-oriented content language that can be used to directly encode information about objects that are instances of classes in an object-oriented ontology. Instead of representing information about objects as a conjunction of facts in a logical sentence, a formula takes the form of an object diagram, i.e. a network of objects and links. The class `OOProposition` is a concrete version of the `Formula` class that is declared to implement the `Proposition` interface (thus allowing objects of this type to appear in messages encoded according to the FIPA Messaging Ontology shown in Figure 6). The class `OODefDescriptor` extends the object diagram by indicating a particular node in the diagram that denotes the object of interest in a query. Figure 2 includes an example instance of this class.

The representation of action descriptors in this ontology is a simplified (for brevity) version of the RDF Schema model behind the FIPA-RDF content language [21, Annex B].

Note that despite the introduction above of the explicit `InformRef` message type, our proposed content language *can* express the relationship between resources and references within an `OOProposition` object and so basic `Inform` messages can be used to send information about objects' references.

Also, the class (or more generally, the "classifier" to use UML terminology) of the object described by an object descriptor is represented by a reference. We assume that each model element in an ontology has been assigned a permanent reference, as in RDF schemas [18].

One current limitation of this content language is that the notion of definite descriptor in this language only allows *objects* to be the subject of a query (the object is identified by the `var` component of an `OODefiniteDescriptor` object). In general, when an object-oriented ontology is used, an agent may wish to ask another agent about the value of some attribute of an object. Adding this capability to our content language requires further investigation of the semantics of definite and indefinite descriptors, references and values, which seem to be more complex in an object-oriented setting than in the first order logic setting on which the semantics of FIPA agent communication is based [19]. In particular, it may be necessary to add a new "query value" speech act to our communication language.

## 6 Ontology-Specific Content Languages

The content language presented above is designed to encode propositions as object diagrams where the nodes can be descriptors for any domain objects that are defined using an object-oriented ontology. Nothing can be assumed about the actual ontology of the objects to be described, and therefore the types of object and the possible relationships that might need to be encoded. Therefore, the basic concepts expressed in the content language ontology are 'action descriptor', 'attribute value', 'link' and 'link end'. If this content is actually describing information about men, women and 'father of' relationships, a receiving agent that wishes to use a more direct and specialised encoding of the information must piece together an interlinked structure of man and woman objects by 'parsing' the object descriptor and link objects that comprise the message content. However, if two agents have established that they share a common ontology, their messages to each other could use a specialised content language that contains direct descriptors of ontology objects and ontology relationships (e.g. man descriptors, and 'father of' links). This is an interesting subject for future research.

## 7 Conclusion

This paper has shown how abstract models of agent communication and content languages are strongly related to the notion of domain ontologies, and has presented a common framework for these types of model. A content language was proposed for communicating information encoded according to an object-oriented ontology, as well as an ontology for FIPA-style agent messages.

Future work includes incorporating these ideas into the NZDIS FIPA agent platform [22], clarifying the semantics of references, refining the object-oriented content language presented, developing techniques to automatically generate ontology-specific content languages, and investigating ways of modelling agent actions within ontologies.

## Acknowledgements

This work is part of the New Zealand Distributed Information Systems project [22, 23] which is funded by the New Zealand government's Public Good Science Fund.

## References

[1] Foundation for Intelligent Physical Agents Web site. http://www.fipa.org/, 2000.

[2] 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://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-23/cranefield-ijcai99-iii.pdf.

[3] J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

[4] G. Booch. *Object-Oriented Analysis and Design with Applications*. Addison-Wesley, 2nd edition, 1994.

[5] Object Management Group Recently Adopted Specifications Web page. http://www.omg.org/techprocess/meetings/schedule/tech2a.html, 2000.

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

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

[8] S. Cranefield and M. Purvis. Extending agent messaging to enable OO information exchange. In R. Trappl, editor, *Proceedings of the 2nd International Symposium "From Agent Theory to Agent Implementation" (AT2AI-2) at the 5th European Meeting on Cybernetics and Systems Research (EMCSR 2000)*, Vienna, April 2000. Austrian Society for Cybernetic Studies. Published under the title "Cybernetics and Systems 2000" (to appear).

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

[10] The XML Cover pages. Organization for the Advancement of Structured Information Standards (OASIS) Web page at http://www.oasis-open.org/cover/xml.html, 2000.

[11] 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.

[12] 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.

[13] M. Erdmann and R. Studer. Ontologies as conceptual models for XML documents. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*. Knowledge Science Institute, University of Calgary, 1999. http://sern.ucalgary.ca/KSI/KAW/KAW99/papers.html.

[14] V. Jagannathan and M. Fuchs. Workshop report on integrating XML and distributed object technologies. In *8th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET-ICE'99)*. IEEE Computer Society Press, 1999. http://www.jeffsutherland.org/xml/IEEE_XML_Report_Draft.htm.

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

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

[17] FIPA-RDF specification. Annex B in FIPA Draft 18-1999: Content languages, Foundation for Intelligent Physical Agents, 1999. http://www.fipa.org/spec/fipa99/fipa9a28.pdf.

[18] Resource description framework (RDF) schema specification 1.0. World Wide Web Consortium Web page at http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, 2000.

[19] M.D. Sadek. Logical task modelling for man-machine dialogue. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 970–975. AAAI Press, 1990.

[20] B. Russell. On denoting. In R. C. Marsh, editor, *Logic and Knowledge: Essays, 1901-1950*. Allen and Unwin, 1956. http://www.santafe.edu/˜shalizi/Russell/denoting/.

[21] FIPA Draft 18-1999: FIPA content language library. http://www.fipa.org/spec/fipa99/fipa99Kawasaki.htm, 1999.

[22] 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.

[23] New Zealand Distributed Information Systems project Web page. http://nzdis.otago.ac.nz/, 2000.