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

Stephen Cranefield, Martin Purvis and Mariusz Nowostawski[1]

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 considers the implications of modelling 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, 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 [4]). An example of a simple ontology expressed using UML appears in Figure 1. This depicts an abstract class `Person` (with the attribute `name`) specialised by two subclasses `Man` and `Woman`. Four association relationships between these classes are shown and at the top of the figure an expression in the Object Constraint Language constrains the possible instances of these associations. In addition, UML's *stereotype* mechanism is used to specify that a `Person` object is a resource (discussed in Section 2).

Traditionally, software agent communication languages (ACLs) have been based on the exchange of information represented as sentences in a logic-based *content language*. 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.

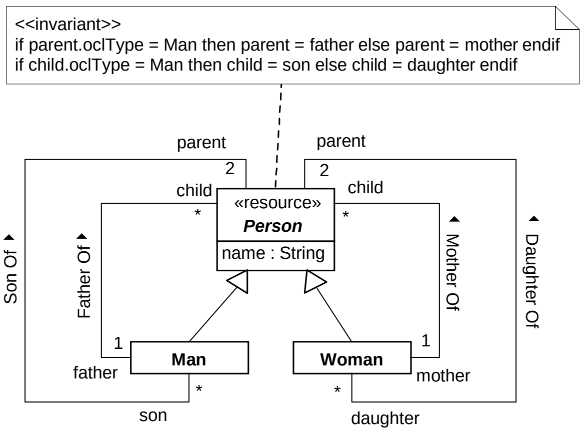

<<invariant>>
if parent.oclType = Man then parent = father else parent = mother endif
if child.oclType = Man then child = son else child = daughter endif
parent
parent
2
2
child
child
*
*
«resource»
Person
name : String
Son Of
Father Of
Mother Of
Daughter Of
1
1
Man
Woman
father
mother
*
*
son
daughter

Figure 1. An ontology describing family relationships

The choice of an object-oriented ontology representation language raises interesting questions about the form in which knowledge should be stored within an agent and encoded within inter-agent messages. Instead of translating between object-oriented internal models and logic-based messages, we believe that a UML object diagram can be considered to be a declarative representation of knowledge and it should be possible to include one directly as the content of a message [5]. Our approach can be summarised as follows. An ontology in UML constitutes 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 can 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'). With this viewpoint, we note that 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.

Figure 2 shows an example of an agent message expressed as a UML 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 depict links: instances of associations between classes.

This diagram can be considered to be an abstraction of a net-

[1] Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand. Email: {scranefield, mpurvis, mnowostawski}@infoscience.otago.ac.nz

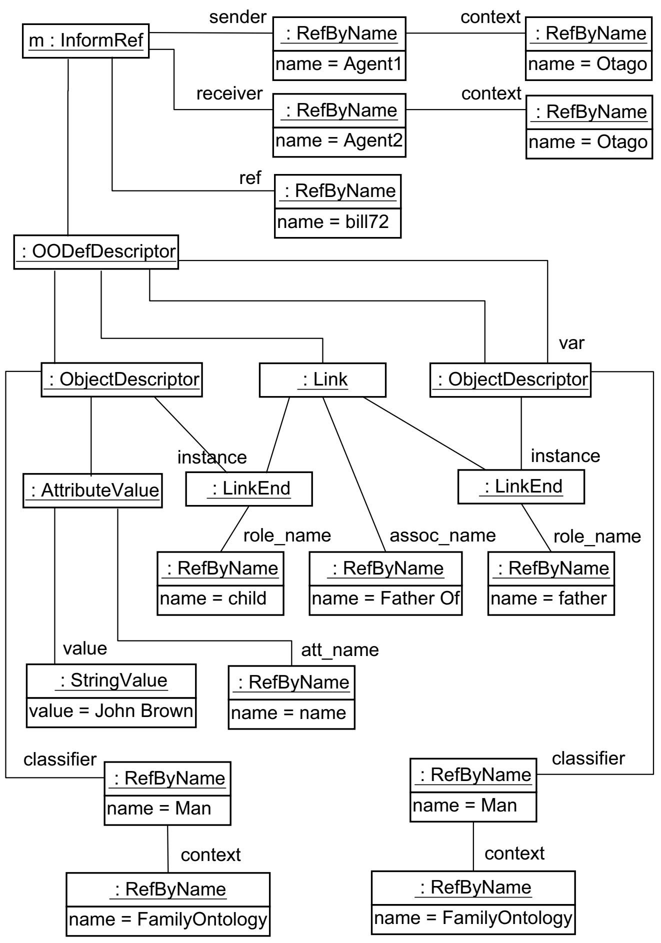
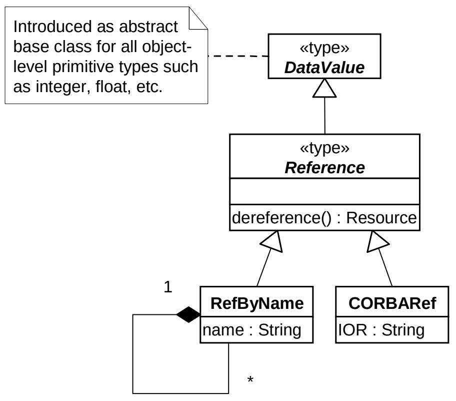


Figure 2. An example message expressed as a UML object diagram

work 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 describing an object for which a reference is required. Its structure encodes an object diagram comprising 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 by a link with a classifier role ("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. The end of Section 4 will discuss a way to provide a more direct encoding of knowledge about an ontology.



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

2 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). 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 [6], based on the notion of a "denoting phrase" from the philosopher Bertrand Russell [7]. 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 ψ that is assumed to hold of some unique but unknown object x. 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

UML meta-model (extended with Resource stereotype)				Level 2 (meta-model)
Data values ontology (Fig. 3)	Domain ontologies (e.g. Fig. 1)	Content language ontologies (e.g. Fig. 6)	ACL ontologies (e.g. Fig. 5)	Level 1 (models)
Domain values (including references)	Domain objects and resources	Knowledge objects	Message objects (e.g. Fig. 2)	Level 0 (objects)

Figure 4. A meta-modelling perspective on agent messaging

also shows examples of two specific 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 'virtual' extension[2] of the UML meta-model. We introduce a specialised type of class, called `Resource` and use an OCL constraint to declare that any class annotated with the stereotype "≪resource≫" will implicitly have a one-to-many association with the class `Reference` from our Data Values ontology (Figure 3). The outcome of this extension is that classes in domain ontologies can now be annotated with the resource stereotype to indicate that objects of these classes have references (see Figure 1 for an example).

3 A META-MODELLING VIEW

The following section presents 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 4 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.

[2] This extension is shown in the long version of this paper [8].

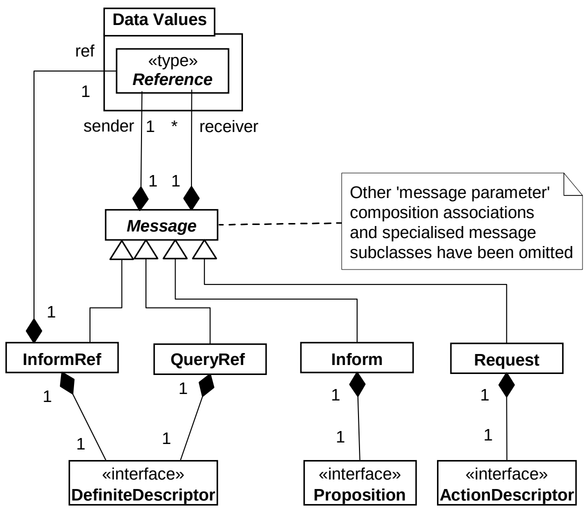


Figure 5. The FIPA Messaging ontology (defining a FIPA-like ACL)

4 MODELLING MESSAGES AND CONTENT

Figure 5 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). 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 (they must include an equality predicate). 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 6 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 5). 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 instance of this class.

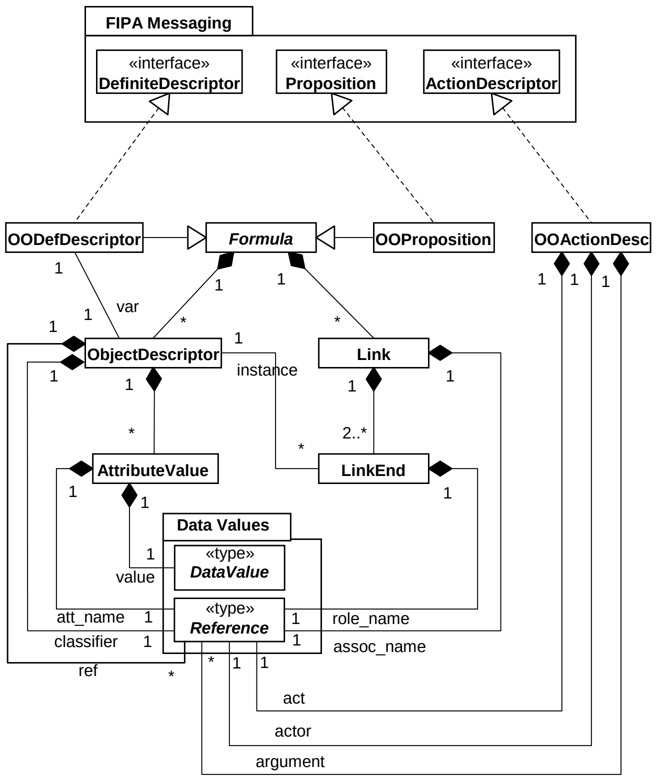


Figure 6. An ontology for an object-oriented content language

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 [9, Annex B].

Note that the class (or, using UML terminology, the "classifier") 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 [10].

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 `OODefDescriptor` 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 [6]. In particular, it may be necessary to add a new "query value" speech act to our communication language.

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 'object 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 important topic for future research.

5 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 [11], 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.

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] Object Management Group Technology Adoptions. http://www.omg.org/techprocess/meetings/schedule/Technology_Adoptions.html, 2000.

[5] S. Cranefield and M. Purvis. Extending agent messaging to enable OO information exchange. In R. Trappl, editor, *Cybernetics and Systems 2000*, pages 573–578, Vienna, 2000. Austrian Society for Cybernetic Studies.

[6] 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.

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

[8] S. Cranefield, M. Purvis, and M. Nowostawski. Is it an ontology or an abstract syntax? Modelling objects, knowledge and agent messages. Discussion Paper 2000/08, Department of Information Science, University of Otago, 2000. http://www.otago.ac.nz/informationscience/publctns/complete/papers/dp2000-08.pdf.gz.

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

[10] Resource description framework (RDF) schema specification 1.0. http://www.w3.org/TR/2000/CR-rdf-schema-20000327/, 2000.

[11] 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.