A proposal for a strongly-typed ACR framework

Stephen Cranefield, Mariusz Nowostawski and Martin Purvis
Department of Information Science
University of Otago
Dunedin, New Zealand
{scranefield, mnowostawski, mpurvis}@infoscience.otago.ac.nz

December 21, 2001

1 Introduction

This proposal briefly presents a design for an abstract content representation scheme that makes no more commitments on the style of content language allowed than the FIPA abstract architecture and allows for automated type-checking of content expressions.

The ACR presented here is based on UML. Class diagrams are used to define an ACL and a set of content language concepts used by the ACL. In addition, particular content languages such as SL or specialised domain-specific ones can also have their abstract syntax defined using class diagrams. The interface between content languages and the ACL is explained using the notion of type substitution: classes in a content language can be declared to 'implement' a predefined interface representing a specific type of content expression (e.g. proposition, term, action or definite description).

In this framework, a message is viewed in the abstract as a network of objects that are instances of the classes in the ACL and content languages models, and it can be represented graphically as a UML object diagram (see Figure 5).

To provide concrete instantiations of messages, ACR mappings need to be defined from UML to implementation languages and serialisation formats. This has been demonstrated to be possible for Java together with a serialisation format using the XML-based Resource Description Framework [1, 2]; however, the current version of this ACR framework includes some UML features that are not supported directly by Java and are not yet simulated in the Java mapping (multiple inheritance, discriminators on generalization relationships, and multiple instantiation).

2 Content language concepts

Figure 1 presents a UML diagram defining a set of interfaces representing types of content language expressions that the FIPA ACL and communicative act library refer to (we will refer to this as "the CL package"). The 'marker interface' pattern [3] is used[1]: the interfaces have no operations but represent distinct (externally-defined) semantic notions.

Some dependency relationships (dashed arrows) are shown, for example, to indicate that an action description is generally composed from a reference to an agent and a description of an act to be performed [2]. However, at this abstract level it is not appropriate to make any decisions about how the implementation structure of these three concepts should be related.

The key content language concepts identified by the FIPA abstract architecture are `Proposition`, `ActionDescription` and `Term`. `DefDescription` (definite description) is also required for use

[1] Alternatively, UML's "type" stereotype could have been used, but the notation for types and implementation classes is more cumbersome than that for interfaces and classes that realise them.

[2] The terminology used here is that an "act" is something that can be peformed by an agent and an "action" is the performance of an act by an agent.

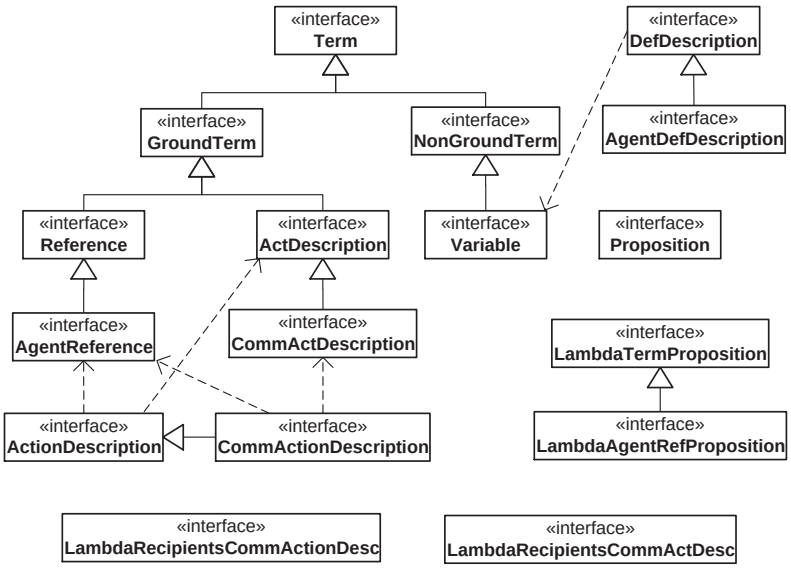

Figure 1: The CL package: generic content language concepts

with the FIPA `query-ref` communicative act. The interface `Variable` is included because the notion of a variable is fundamental to the semantics of definite descriptions [4].

The diagram contains some additional interfaces representing concepts that are not explicitly mentioned by the ACL or communicative act definitions—we believe there are advantages to explicitly modelling these concepts.

- The notion of a reference is reified to allow a better understanding of the role that distributed reference schemes such as CORBA IORs and Web URIs can play within messages. This is further specialised to the notion of `AgentReference`—a reference to an agent.
- Terms describing communicative acts (CAs) and actions are represented by the interfaces `CommActDescription` and `CommActionDescription`.
- A number of `Lambda...` interfaces are introduced to represent the needs of communicative acts having arguments representing propositions with missing subjects (useful in 'call for proposal'-style CAs) and communicative acts or actions with missing recipients (useful in proxy and propagate CAs).

Note that no commitment is made concerning the structure of propositions. All that ACL and communicative act definitions require is that content that plays a particular semantic role (e.g. a proposition) can be identified as playing that role. In particular, note that the concept of a predicate is not included in the CL package. Although some FIPA communicative acts require propositions as parameters, the ACL is neutral about how propositions are represented—a content language is free to represent propositions in non-standard ways, for example as networks of objects asserting the values of the objects' attribute values and some relationships that hold between them.

3 Abstract models of specific content languages

Given the content language concepts model, a particular content language can be defined in an abstract way as a class diagam, with UML realisation relationships used to indicate how particular content language classes correspond to generic content language concepts. Provided that a class contained in a specific content language model is declared to implement one of these interfaces, an instance of that class can be

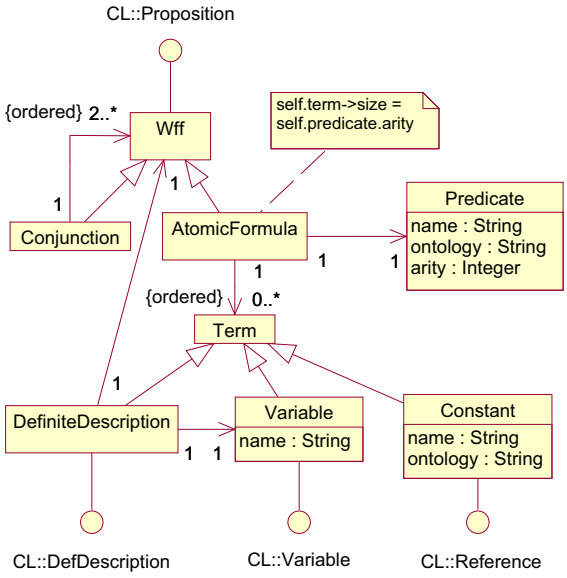

Figure 2: A partial model of a FIPA SL style content language

included within an ACL message wherever that type of element makes sense for the particular communicative act being used. This allows a strongly-typed definition of an ACL (see Section 4) while still supporting the use of alternative domain- or community-specific content languages.

For a content language defined in this way there will be a need for an agent platform to have codecs for the different serialisation formats required for that language. However, in principle these codecs can be generated automatically from the UML class diagram, and in practice this has been shown to be possible using RDF [1, 2]. Serialisations using XML directly are also possible [5], but further work is needed to determine how best to define mappings to string-based syntaxes (extra information would need to be provided, such as the syntax to be used for lists of items).

This section briefly presents two examples of particular content languages.

Figure 2 shows a fragment of a FIPA SL-style language. The UML 'lollipop' symbol is used to indicate realisation relationships, i.e. the declaration that a class implements the interface at the round end of the lollipop. In this case, the interfaces implemented are the general content language concepts modelled in the CL package from Figure 1, and the notion of 'implementation' is that of the marker interface pattern: the class plays the semantic role represented by the interface. A conventional content language such as this could be recommended as a default to be included in any FIPA agent platform.

Figure 3 shows part of a ontology-specific object-oriented content language generated from an ontology defined in UML [6]).

4 An abstract ACL syntax

Figure 4 shows a partial model of a FIPA-style agent communication language. The lower part of the diagram shows how particular CA types are represented by specialisations of the notion of a message. Note that the specialised CA message classes declare their required content types by referring to the interfaces defined in the content language concepts model presented in Figure 1. This provides a strongly typed account of the various numbers and types of content expression required in the body of different CAs. Expressions in any content language can be included within an ACL expression provided that they implement the appropriate interface from the CL package. For example the Object Query Language could be used as a form of definite description simply by defining a class `OQLDescription` with a string-valued attribute `query` and declaring it to implement `CL::DefDescription`.

The diagram does not show all FIPA CAs, and an explicit `InformRef` message type is added to

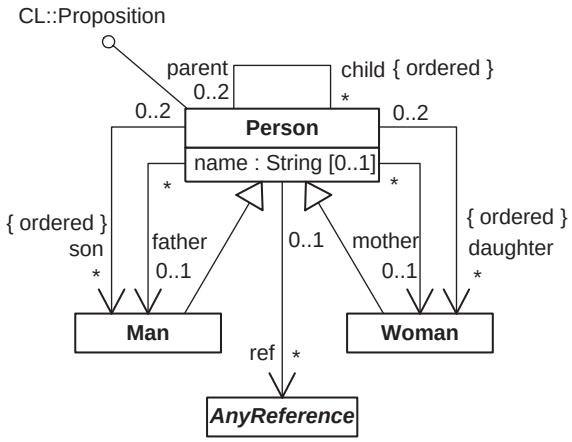

Figure 3: Proposition classes generated from an ontology in UML

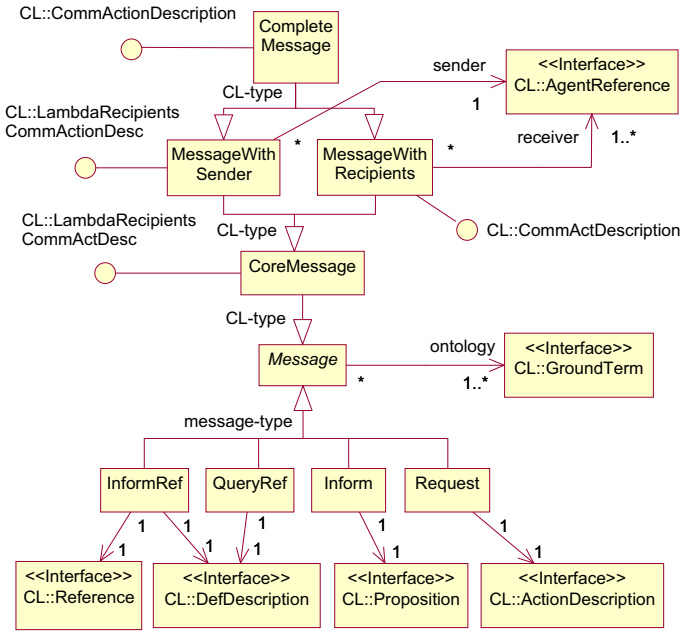

Figure 4: A partial model of a FIPA-style ACL

provide a more convenient way of responding to a `QueryRef` message (currently in FIPA it is necessary to send an `inform` message with a content proposition that equates a term with the definition description that was sent in the original query). The model also currently omits the various additional parameters such as `conversation-id` that a FIPA ACL message may optionally have.

The modelling of an ACL is complicated by the FIPA practice of reusing the ACL message syntax to represent completely or partially specified communicative acts that can appear within the content of messages (e.g. as one of the arguments of a proxy CA). To account for this in a strongly typed modelling language requires the identification of the roles that variously complete forms of message can play as content language expressions. The top half of the figure shows how different specialisations of the abstract message class are declared to 'implement' different types of content language concept from the content language concepts model.

To separate the two distinct type of message specialisation discussed above (different types of communicative act represented and different types of content language expression represented), the UML notion of separate dimensions of specialisation is used. Discriminators (`message-type` and `CL-type` in the figure) can be used to indicate a particular dimension of specialisation. An instance of a concept that is specialised in multiple dimensions must either belong to a class that multiply inherits from specialised classes from all dimensions of specialisation or it must be 'multiply instantiated' (i.e. belong to more than one class) to cover all specialisation dimensions. It would be impractical to use multiple inheritance to define classes covering all possible combinations of different CAs and different combinations of omitted sender and recipients links, so instances of messages in this model are required to be multiply instantiated (see Figure 5). The use of multiple instantiation complicates the mapping to conventional OO programming languages, but this could be simulated using delegation.

5 An example message

Figure 5 shows an object diagram representing an example message using the ontology-specific content language presented in Figure 3. Note that the top-level message object (in the centre top of the figure) is "multiply instantiated", i.e. it belongs to two classes. As discussed in Section 4, this feature of UML is used to allow a strongly-typed account of the multiple roles that message objects can play: as messages and also as content expressions denoting communicative actions.

The figure assumes that a number of other UML models have been defined: an AMF model defining agent descriptor structures and a BasicCL model defining primitive forms of expression such as identifiers.

The message represents a FIPA `inform` message making the assertion that there is a man named "Joe Brown" identified (in Semantic Web fashion) by a particular URI, and his mother (identified by another URI) is named "Mary Brown".

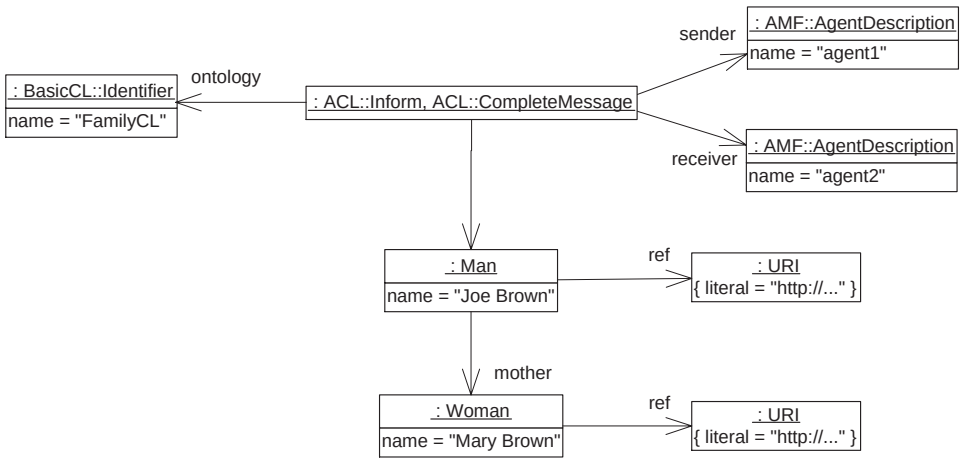

Figure 5: An example message

References

[1] S. Cranefield. Networked knowledge representation and exchange using UML and RDF. *Journal of Digital Information*, 1(8), 2001. http://jodi.ecs.soton.ac.uk/Articles/v01/i08/Cranefield/.

[2] S. Cranefield. UML and the Semantic Web. In *Proceedings of the International Semantic Web Working Symposium (SWWS)*, 2001. http://www.semanticweb.org/SWWS/program/full/paper1.pdf.

[3] M. Grand. *Patterns in Java*, volume 1. Wiley, 1998.

[4] 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/.

[5] M. Jeckle. Practical usage of W3C's XML-Schema and a process for generating schema structures from UML models. In *Proceedings of the International Conference on Advances in Infrastructure for E-Business, Science, and Education on the Internet (SSGRR 2001)*, 2001.

[6] S. Cranefield and M. Purvis. Generating ontology-specific content languages. In *Proceedings of the Workshop on Ontologies in Agent Systems, 5th International Conference on Autonomous Agents*, 2001. http://CEUR-WS.org/Vol-52/oas01-cranefield-2.pdf.