

Implementing agent communication languages directly from UML specifications

Stephen Cranefield, Mariusz Nowostawski and Martin Purvis

Department of Information Science

University of Otago

PO Box 56, Dunedin, New Zealand

{scraneffield, mnowostawski, mpurvis}@infoscience.otago.ac.nz

ABSTRACT

This paper proposes the use of the Unified Modeling Language as a formalism for defining the abstract syntax of agent communication languages and their associated content languages, with implementations for specific programming languages and message serialization formats generated automatically from these models. This eliminates the error-prone manual process of generating compatible language bindings and message encodings for different implementations of a language and, by decoupling these issues from the implementation of other agent code, provides a framework for specifying and experimenting with alternative agent communication languages. The approach is illustrated by an automatically generated application programmer interface in Java for processing strongly-typed messages and serializing them using RDF.

Categories and Subject Descriptors

I.2.11 [Computing Methodologies]: Artificial Intelligence—Multiagent Systems

General Terms

Languages, Design, Experimentation

1. INTRODUCTION

Agent communication languages (ACLs) and the content languages used with them have traditionally been defined as string-based languages specified by a grammar [3]. However, the available software toolkits for building agents have generally not provided a high level of support for the string processing required to parse and access the contents of messages. Creating agents using these toolkits, although preferable to starting from scratch, requires time-consuming and error-prone implementation of parsers and pattern matchers for KIF or other content languages. Furthermore, the use of strings to encode messages reduces the software engineering benefits of using strongly-typed languages such as Java to develop agents, as malformed messages cannot be detected until they are parsed at run time.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'02, July 15-19, 2002, Bologna, Italy.

Copyright 2002 ACM 1-58113-480-0/02/0007 ...\$5.00.

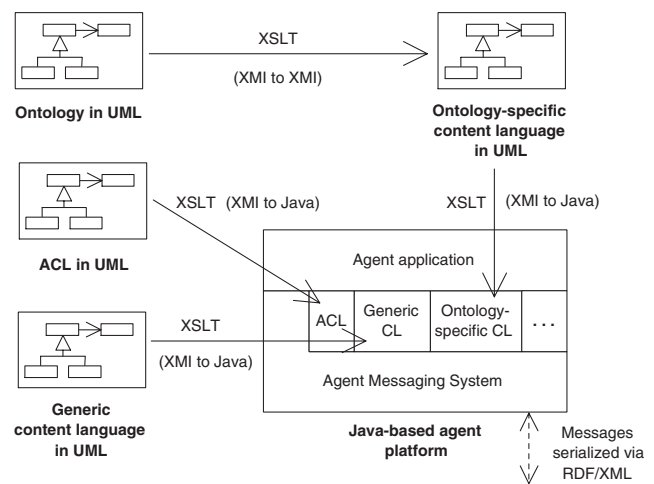


Figure 1: Overview of the approach applied to Java

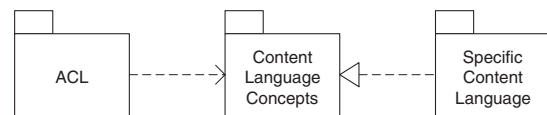


Figure 2: Decoupling the ACL and content language models

This paper proposes a strongly-typed approach to the definition and implementation of agent communication and content languages, based on:

1. the use of the Unified Modeling Language (UML) [4] to define the abstract syntax of an ACL and content languages, and
2. an automated mapping from UML to Java classes, including XML-based object serialisation support using the Resource Description Framework. [7]

This provides a high-level way of defining agent communication languages together with a standard method of generating language implementations that can be “plugged into” an agent platform. As well as offering the benefits of an object-oriented application programmer interface for message-handling, this approach supports research into novel agent communication and content languages by both easing the definition and implementation of those languages and decoupling these aspects from the underlying agent platform.

```

// Construct query structure containing variable x
WomanDescriptor mary = new WomanDescriptor();
ManDescriptor fatherOfMary = new ManDescriptor();
StringPattern nameOfMary = new StringPattern();
nameOfMary.setValue("Mary Smith");
mary.setName(nameOfMary);
mary.setFather(fatherOfMary);
StringPattern unknownName = new StringPattern();
unknownName.setVarName("x");
fatherOfMary.setName(unknownName);
// Construct message object
Message m = new Message(new Identifier("FamilyOnt"),
    new SenderAndReceivers(new AgentRef("agent1"),
        Collections.singleton(new AgentRef("agent2"))),
    new QueryRef(new OODefDescription("x", mary)));

// Marshall message
messageStream = ...; // Get output stream to agent platform's message transport system
MarshalHelper.marshalObjects(Collections.singleton(m), // All objects to be marshalled
    m, // Root object
    "http://some.namespace/for/message#", messageStream);

```

Figure 3: Using the generated Java code to create and serialize a message

2. IMPLEMENTATION OVERVIEW

Figure 1 illustrates our prototype implementation of the approach. The ACL and content languages are defined as UML class diagrams and exported in the standard XMI format. XSLT [6] stylesheets are used to produce sets of Java classes corresponding to those in the UML models. These can then be plugged into the agent messaging system of a Java-based agent platform (we have not yet implemented the plug-in mechanism but intend to add this to our FIPA-compliant agent platform, Opal [5]). In addition, ontologies can also be modelled using UML. The classes in these models specify the structure of domain entities, whereas messages contain propositions and queries about domain entities. Therefore another mapping (designed [2], but not yet implemented) produces UML models for ontology-specific content languages defining object-oriented structures for propositions and queries about instances of an ontology, and Java classes can also be generated from these.

Figure 2 illustrates how we decouple the definition of ACLs and content languages by providing a set of UML ‘marker’ interfaces representing fundamental concepts of content languages such as propositions. ACL models have associations to these interfaces and

particular content language models contain classes that implement the interfaces [2].

Figure 3 shows some example Java code to create a message as an object structure and serialize it. The structure of the message is shown using a UML object diagram in Figure 4 (the UML models of the ACL and ontology-specific content language used are presented elsewhere [2]).

3. SUMMARY

We have proposed the use of abstract syntax models of ACLs and content languages in UML, with implementations generated by automated mappings from those models. This enables a strongly-typed treatment of messages, and facilitates the development of new ACLs and content languages. We have submitted this work for consideration by FIPA [1].

4. REFERENCES

- [1] S. Cranefield, M. Nowostawski, and M. Purvis. A proposal for a strongly-typed ACR framework. Input Document f-in-00048, Foundation for Intelligent Physical Agents. <http://www.fipa.org/docs/input/f-in-00048/>, 2002.
- [2] S. Cranefield and M. Purvis. A UML profile and mapping for the generation of ontology-specific content languages. *Knowledge Engineering Review, Special Issue on Ontologies in Agent Systems*, 2002. Forthcoming.
- [3] Y. Labrou, T. Finin, and Y. Peng. The current landscape of agent communication languages. *IEEE Intelligent Systems*, 14(2):45–52, 1999.
- [4] Object Management Group. Unified Modeling Language Specification, version 1.4. <http://www.omg.org/cgi-bin/doc?formal/01-09-67>, 2001.
- [5] M. Purvis, S. Cranefield, M. Nowostawski, and D. Carter. Opal: A multi-level infrastructure for agent-oriented software development. Discussion Paper 2002/01, Department of Information Science, University of Otago. <http://www.otago.ac.nz/informationscience/publctns/complete/papers/dp2002-01.pdf.gz>, 2002.
- [6] World Wide Web Consortium. XSL Transformations (XSLT) specification version 1.0. <http://www.w3.org/TR/xslt>, 1999.
- [7] World Wide Web Consortium. Resource Description Framework overview Web page, version 1.141. <http://www.w3.org/RDF/Overview.html>, 2002.

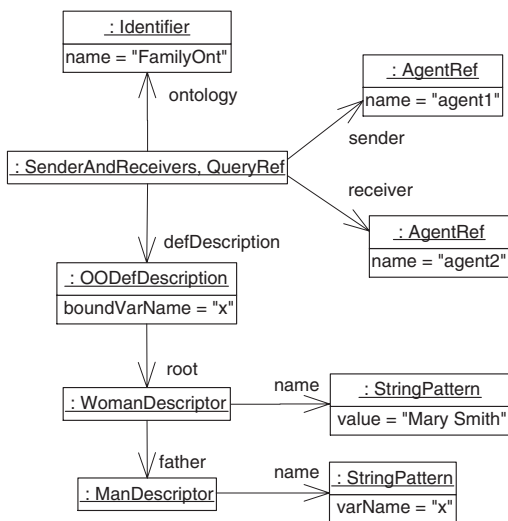


Figure 4: A UML object diagram of the example message