Bridging Agent Theory and Object Orientation: Agent-Like Communication Among Objects

Matteo Baldoni[1], Guido Boella[1], and Leendert van der Torre[2]

[1] Dipartimento di Informatica. Università di Torino - Italy
{baldoni,guido}@di.unito.it
[2] University of Luxembourg
leendert@vandertorre.com

Abstract. This paper begins with the comparison of the message-sending mechanism, for communication among agents, and the method-invocation mechanism, for communication among objects. Then, we describe an extension of the method-invocation mechanism by introducing the notion of "sender" of a message, "state" of the interaction and "protocol" using the notion of "role", as it has been introduced in the `powerJava` extension of Java. The use of roles in communication is shown by means of an example of protocol.

1 Introduction

The major differences of the notion of agent w.r.t. the notion of object are often considered to be "autonomy" and "proactivity" [25]. Less attention has been devoted to the peculiarities of the *communication capabilities* of agents, which exchange messages while playing roles in protocols. For example, in the contract net protocol (CNP) an agent in the role of initiator starts by *asking for* bids, while agents playing the role of participants can *propose* bids which are either *accepted* or *rejected* by the Initiator.

The main features of communication among agents which emerge from the CNP example are the following:

1. The message identifies both its *sender* and its *receiver*. E.g., in FIPA the acceptance of a proposal is:
   ```
   (accept-proposal :sender i :receiver j :in-reply-to
       bid089 :content X :language FIPA-SL).
   ```
2. The interaction with each agent is associated to a *state* which evolves according to the messages that are exchanged. The meaning of the messages is influenced by the state. E.g., in the FIPA iterated contract net protocol, a "call for proposal" is a function of the previous calls for proposals, i.e., from the session.
3. Messages are produced according to some *protocol* (e.g., a call for proposal must be followed by a proposal or a reject).
4. The sender and the receiver play one of the *roles* specified in the protocol (e.g., initiator and participant in the contract net protocol).

R.H. Bordini et al. (Eds.): ProMAS 2006, LNAI 4411, pp. 149–164, 2007.

5. Communication is *asynchronous*: the response to a message does not necessarily follow it immediately. E.g., in the contract net protocol, a proposal must follow a call for proposal and it must arrive, no matter when, before a given deadline.
6. The receiver autonomously decides to comply with the message (e.g., making a proposal after a call for proposal).

The message metaphor has been originally used also for describing method calls among objects, but it is not fully exploited. In particular, message-exchange in the object oriented paradigm has the following features:

1. The message is sent to the receiver without any information concerning the sender.
2. There is no state of the interaction between sender and receiver.
3. The message is independent from the previous messages sent and received.
4. The sender and the receiver do not need to play any role in the message exchange.
5. The interaction is synchronous: an object waits for the result of a method invocation.
6. The receiver always executes the method invoked if it exists.

These two scenarios are rather different but we believe that the object-oriented (OO) paradigm can learn something from the agent-oriented world. The research question of this paper is thus: is it profitable to introduce in the OO paradigm concepts taken from agent communication? how can we introduce in the OO paradigm the way agents communicate? And as subquestions: which of the above properties can be imported and which cannot? How to translate the properties which can be imported in the OO paradigm? What do we learn in the agent-oriented world from this translation?

The methodology that we use in this paper is to map the properties of agent communication to an extension of Java, `powerJava` [3,4,5], which adds roles to objects. Roles are used to represent the sender of a message (also known as the "player of the role"), to represent the state of the interaction via role instances, allowing the definition of protocols and asynchronous communication as well as the representation of the different relations between objects.

The choice of the Java language is due to the fact that it is one of the prototypical OO programming languages; moreover, MAS systems are often implemented in Java and some agent programming languages are extensions of Java, e.g., see the Jade framework [8] or the JACK software tool [24]. In this way we can directly use complex interaction and roles offered by our extension of Java when building MAS systems or extending agent programming languages.

Furthermore, we believe that in order to contribute to the success of the Autonomous Agents and Multiagent Systems research, the theories and concepts developed in this area should be applicable also to more traditional views. It is a challenge for the agent community to apply its concepts outside strictly agent-based applications. The OO paradigm is central in Computer Science and, as observed and suggested also by Juan and Sterling [18], before AO can be widely