Engineering Agent Interactions from ACL-based Reusable Connectors [*]

Juan Manuel Serrano, Sascha Ossowski, Sergio Saugar
Artificial Intelligence Group
School of Engineering
University Rey Juan Carlos
JuanManuel.Serrano@urjc.es, Sascha.Ossowski@urjc.es, Sergio.Saugar@urjc.es

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Design, Theory, Standardization

Keywords

Agent Communication Languages and Protocols, Multi-agent programming frameworks, JADE

ABSTRACT

This paper reports on the $\mathcal{RICA}$–$\mathcal{J}$ multiagent programming framework, which provides executable constructs for each of the organizational, ACL-based modelling abstractions of the $\mathcal{RICA}$ theory. Setting out from a component and connector perspective on the elements of the $\mathcal{RICA}$ metamodel, their execution semantics is defined and instrumented on top of the JADE platform. Moreover, a systematic reuse approach to the engineering of interactions is put forward.

1. INTRODUCTION

In the past few years, multi-agent systems (MAS) have been proposed as a suitable software engineering paradigm to face the challenges posed by the development of large-scale, open systems. In particular, organization-oriented abstractions such as roles, social interactions, groups, organizations, institutions, etc., have proved to be an effective means to model the interaction space of complex MAS [4]. Drawing on the $\mathcal{RICA}$ metamodel [3], this paper attempts to reconcile MAS organizational research with two major branches in software engineering: software architectures [1] and component-based development [2]. Section 2 will show how a Component & Connector (C&C) perspective fits the $\mathcal{RICA}$ theory, and analyzes communicative roles and interactions from the point of view of generic software components. Section 3 provides a survey of the $\mathcal{RICA}$–$\mathcal{J}$ framework, emphasizing the mapping of the C&C-based execution semantics to the JADE platform, and the architecture of a $\mathcal{RICA}$–$\mathcal{J}$ application.

2. THE $\mathcal{RICA}$ THEORY

The metamodel of the $\mathcal{RICA}$ theory provides a modelling language of the *organizational* and *communicative* features of MAS [3]. Figure 1 shows a $\mathcal{RICA}$ model of a simple e-commerce application expressed in terms of a UML class diagram. The organizational model comprises the different types of social interactions agents may engage in. For instance, client and clerk agents may jointly perform *purchase* interactions, which instill an e-commerce transaction of a good using some credit card. Other types of interactions include *information exchange* or *advisory* interactions.

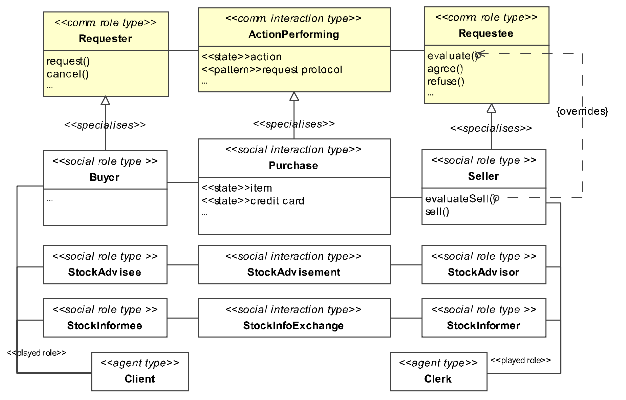

Figure 1: $\mathcal{RICA}$ model of a simple e-commerce application

From a software architecture perspective, agents can be conceived as a particular type of (autonomous, social, situated, etc.) *software component*, since they represent the major computational units of MAS. Furthermore, social interactions can be considered as a kind of *connector*, since they mediate and regulate the different interactions among agents. Thus, the *caller* and *callee* roles in a RPC interaction, or the *reader* and *writer* in a pipe connection, are analogues of the *buyer* and *seller* social roles declared by a purchase interaction. Social interactions mainly differ from pipes, SQL links, and other types of connectors in their characteristic interaction mechanism: the Agent Communication Language (ACL).

[*]Research sponsored by the Spanish Ministry of Education and Science (MEC), project TIC2003-08763-C02-02

In the $\mathcal{RICA}$ theory, so-called communicative interactions provide the pragmatic features of application-dependent social interactions, which basically differ at the semantic level. Thus, the purchase interaction is recursively defined in terms of the *action performing* communicative interaction, which provides the communicative actions (CAs) and protocols required by buyer and seller agents. Besides inheriting these components, social roles and interactions may override and extend some of the super-type features (e.g. the *evaluate* social action is overriden by the *evaluateSell* action; the *sell* action extends the actions that can be performed by seller agents). Thanks to the cross-domain features of CAs and protocols, ACL-based interactions are generic first-class reusable components. Note that these reusable components are actually *connectors*, in the C&C perspective.

As a first step towards defining programming abstractions that correspond to the $\mathcal{RICA}$ modeling entities, the execution semantics of the later needs to be defined. Accordingly, the $\mathcal{RICA}$ theory specifies a generic agent architecture which supports the dynamics of agents within a $\mathcal{RICA}$ organization. Behavioural features of role types, such as *activation* and *participation* conditions, as well as the services provided by protocol instances at run-time are specified as well. Overriding declarations of role actions introduces a *dynamic binding* feature in the execution semantics of $\mathcal{RICA}$ models. For instance, when seller agents are required to perform the *evaluate* action, the action that is actually executed at run-time is *evaluateSell*. This mechanism allows for a direct reuse of interaction protocols, fully specified in the scope of communicative interactions.

3. THE $\mathcal{RICA}$–$\mathcal{J}$ FRAMEWORK

The $\mathcal{RICA}$ theory, given its metamodel and execution semantics, can be conceived as a programming language with close links to Architectural Description Languages (ADLs). As a more pragmatic alternative to the direct instrumentation of the "$\mathcal{RICA}$ programming language", the $\mathcal{RICA}$–$\mathcal{J}$ ($\mathcal{RICA}$-JADE) framework instruments the $\mathcal{RICA}$ theory on top of the FIPA-compliant JADE platform. Basically, the $\mathcal{RICA}$–$\mathcal{J}$ framework adds a layer composed of the `rica.reflect` and `rica.core` Java packages, which instrument the $\mathcal{RICA}$ metamodel and execution semantics, respectively.

The `rica.core` classes map the common behaviour and structure of agents, roles, etc., as defined by the $\mathcal{RICA}$ execution semantics, to the supported abstractions of the JADE framework: basically, agents, behaviours and ACL messages. The resulting architecture of a $\mathcal{RICA}$–$\mathcal{J}$ agent is exemplified in figure 2, where the run-time structure of a possible client agent is shown. Firstly, any `rica.core.Agent` (a kind of JADE agent) schedules a `RoleMonitor` behaviour (a JADE cyclic behaviour), in charge of monitoring the activation conditions of roles. Secondly, each played role is instrumented by a `SocialRoleBehaviour` (a JADE parallel behaviour) which contains a `ParticipationMonitor` behaviour, which acts as an *interaction factory*. Moreover, it also contains one behaviour for each of the interactions in which the agent is participating.

Different types of programmers participate in the development of an application in the $\mathcal{RICA}$–$\mathcal{J}$ framework. Firstly, *component developers* are in charged of implementing the communication library containing differerent communicative roles and interactions (such as the ones underlying FIPA ACL [3]), as well as the required protocol formalisms (e.g.

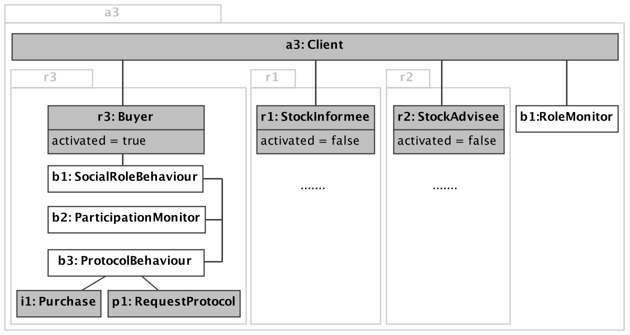

Figure 2: $\mathcal{RICA}$–$\mathcal{J}$ agent architecture

AUML sequence diagrams). Secondly, *organizational developers* implements the social roles and interactions of the MAS organization. Last, *independent agent programmers* rely on the components available in the organizational library, possibly customizing the social roles to be played by overriding their default functionality or implementing their abstract actions.

4. CONCLUSION

The $\mathcal{RICA}$–$\mathcal{J}$ framework relieves agent programmers from the instrumentation of low-level issues concerning the dynamics of agents within the organization. It should be stressed that the proposed approach does not endanger the autonomy of agents, since the social roles available in the organization library may be fully customized to account for the particular requirements of each agent. Moreover, modeling social interactions in terms of software connectors has as a major consequence the identification of the characteristic roles that their participant agents may play within it. This feature differentiates the $\mathcal{RICA}$ organizational metamodel from other proposals, and allows a systematic approach to the reuse of agent interactions. Future work will concentrate on the extension of the underlying metamodel with coarse-grained organizational abstractions, and the instrumentation of the interaction monitoring and compliance capabilities that any open-driven framework must offer.

5. REFERENCES

[1] R. Allen and D. Garlan. A Formal Basis for Architectural Connection. *ACM Transactions on Software Engineering and Methodology*, 6(3):213–249, June 1997.

[2] I. Jacobson, M. Griss, and P. Jonsson. *Software Reuse. Architecture, Process and Organization for Business Success*. Addison-Wesley, 1997.

[3] J. M. Serrano and S. Ossowski. An organizational metamodel for the design of catalogues of communicative actions. *Intelligent Agents and Multiagent Systems (Kuwabara & Lee, ed.), Lecture Notes in Computer Science*, 2413:92–108, 2002.

[4] F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12(3):317–370, July 2003.