Architectural Design of Component-Based Agents: A Behavior-Based Approach

Jean-Pierre Briot[1,2], Thomas Meurisse[1], and Frédéric Peschanski[1]

[1] Laboratoire d'Informatique de Paris 6 (LIP6)
Université Paris 6 - CNRS
Case 169, 4 place Jussieu
75252 Paris Cedex 05, France
{Jean-Pierre.Briot,Thomas.Meurisse,Frederic.Peschanski}@lip6.fr
[2] Currently visiting CS Dept., PUC-Rio, Rio de Janeiro, Brazil

Abstract. This paper relates an experience in using a component model to design and construct agents. After discussing various rationales and architectural styles for decomposing an agent architecture, we describe a model of component for agents, named MALEVA. In this model, components encapsulate various units of agent behaviors (e.g., follow gradient, flee, reproduce). It provides an explicit notion of control flow between components (reified through specific control ports, connexions and components), for a fine grain control of activation and scheduling. Moreover, a notion of composite component allows complex behaviors to be constructed from simpler ones. Two examples, in the domain of multi-agent based simulation, are presented in this paper. They illustrate the ability of the model to facilitate both bottom-up and top-down approaches for agent design and construction and also to help at different types of potential reuse.

Keywords: component, agent, multi-agent systems, behavior, design, composition, architecture, simulation.

1 Introduction

Components and multi-agent systems are among current popular approaches for designing and constructing software. Both of them propose abstractions to organize software as a combination of software elements, with easier management of evolution (such as changing and adding elements). We consider that multi-agent systems push further the level of abstraction and the flexibility of component coupling, notably through self-organization abilities [5]. Meanwhile, we believe that the component concept and technology may help in the actual construction of multi-agent systems:

– at the *system level*, we may consider each agent as a component, to provide some support for integration, configuration, packaging and distribution of multi-agent systems,

R.H. Bordini et al. (Eds.): ProMAS 2006, LNAI 4411, pp. 71–90, 2007.

– at the *agent level*, by providing some support for structuration, (de)composition and reuse of its internal architecture.

In this paper, we focus on the second category. Indeed, we believe that the design and construction of an individual agent can benefit from the principles of software components (encapsulation, explicit connectors...). Our objective is to help in an incremental design of agents as the composition of simpler agent behaviors and activities (e.g., follow gradient, flee, reproduce...). Our main application field target is multi-agent-based simulation of phenomena (biological, ecological, social, economical...). Their specific requirements definitely influenced our design decisions, as well as our case studies and applications conducted. Meanwhile, we believe that the scope of the component model that we propose goes beyond the domain of multi-agent-based simulations, and that other application areas could benefit from some of its principles, e.g., making control available at the composition level.

After first discussing some rationales for the design of component-based agent architectures, and referring to related work, we describe a component model named MALEVA, which aims at encapsulating and composing units of behaviors to describe complex agent architectures. This component model does not impose a specific architectural style. One of its specificities is that it applies the principles of components and software composition to the specification of control, through the notions of control ports and control components. Two examples will be presented in the paper. They illustrate how MALEVA can support bottom up as well as top down design, and also how it offers some potential for reuse and specialization, through: structural composition of behaviors, abstract behaviors and design patterns.

2 Rationales and Styles for Agent Architectures

We consider an *agent architecture* as the description of the relations between the software (or sometimes hardware) modules that implement the various agent functions. Except for simple reactive agents, the architecture of an agent may be complex. It is thus useful to describe it in terms of simpler lower level components that interact with each other.

Inspired by the seminal work on software architectures by Shaw and Garlan [23], we tentatively propose a classification for agent architectures from the perspective of *architectural styles*.[1] In this paper, we focus on the rationales for decomposition and on their impact on the reuse of the architecture or/and of its components. It is important to note that we do not expect our typology to be exhaustive. Also note that, as for software architectures [23], a complex architecture (e.g., InteRRaP, see Section 2.3) may juxtapose and combine several architectural/decomposition styles.

[1] There is of course no unique typology for agent architectures, and we may find other classifications in the literature (e.g., in [20]), such as horizontal/vertical or reactive/cognitive/hybrid.