A Generic Model of Cognitive Agent to Develop Open Systems

E. E. Scalabrin*, L. Vandenberghe, H. de Azevedo**, J-P. A. Barthès

Université de Technologie de Compiègne
CNRS URA 817 HEUDIASYC
BP 529, 60205 Compiègne Cedex, France
E-Mail: {escalab, lvanden, hazevedo, barthes}@hds.utc.fr

Abstract. The paper presents an approach to building multi-agent systems. We are interested in complex agents able to reason about their tasks, and to display a proactive behavior, when installed on a network of heterogeneous computers. We developed the concept of a generic agent (GAg), equipped with the basic communication and "mental" structure, but ignorant, i.e., not containing any application expertise, nor having any knowledge about the external world. When building an application, actual agents are cloned from the generic agent. In addition, a specific environment, OSACA (Open System for Asynchronous Cognitive Agents), simplifies the process of creating agents on a network of heterogeneous machines. The paper discusses mainly the basic structure of the generic agent. Our approach is also illustrated with a small example of an agent which helps writing a technical paper in a research laboratory.

1 Introduction

A new branch of artificial intelligence studies societies of agents working together to solve problems that are naturally distributed, or too large and too complex to be solved by a single program. Our research work takes place in this context. Indeed, we are interested in complex agents able to reason about their tasks, and to display a proactive behavior, when installed on a network of heterogeneous computers. However, when one wants to develop multi-agent systems in practice, a major difficulty is to write the supporting code, to test the prototypes, and to maintain the resulting system. In particular, changes in a multi-machine environment are a real pain. Thus, we focused our work onto open systems, i.e., systems containing a variable number of agents which do not have to be halted when changes occur, i.e., when some agents join the system, or leave the system. In addition, we wanted to simplify the way to build a particular agent, by offering an off-the-shelf typical agent containing all basic mechanisms, and which can be cloned to serve as a starting point for more complex modules.

The resulting environment, OSACA (Open System for Asynchronous Cognitive Agents), gives the user the ability to create, develop, and test multi-agent

* Supported by CAPES/Brazil, grant number 1307/94
** On leave from CEFET-PR, Brazil and supported by CITS/CNPq, grant number 260139/92
systems with minimal effort. Agents are implemented in the MOSS programming environment [1]. Within OSACA [16, 15], each agent is an independent process, communicating with other agents by sending messages asynchronously. OSACA is installed on *UNIX*\textsuperscript{TM}, including *SPARC*\textsuperscript{TM} machines running *SunOS*\textsuperscript{TM} and *Solaris*\textsuperscript{TM}. *Mac*\textsuperscript{TM} OS will be supported in the next release. UDP/IP is used to carry messages between agents in a LAN environment.

The rest of the paper is organized as follows. Section 2 presents a brief overview of Multi-Agent Systems and gives examples of different architectures. Section 3 details the generic agent, GAg, presenting its model and functionalities. Section 4 introduces an application, MEMOLAB, to illustrate how to use GAg to implement an application agent. The paper ends with some concluding remarks about the generic agent approach and future work.

2 Towards Open Systems

Multi-agent systems can be classified according to their architecture (overall organization), to the degree of autonomy of each agent, to the type of protocol they use to communicate, or their complexity. A major distinction concerns reactive vs. autonomous agents. Reactive agents are very simple without any representation of their environment. They interact using stimulus-response type behavior [6]. Thus, intelligent behaviors can emerge from a population of numerous agents [3]. Cognitive agents are very complex. Each agent is specialized and can function by itself. It has a (partial) model of its environment and acts in accordance with the model. Complex agents may have intentions to guide their behavior. The various systems proposed today differ by their overall architecture, communication possibility, and complexity of the basic agent.

*Blackboard systems* allow several specialists (often called knowledge sources) to interact through shared data (posted on the blackboard). Normally, communication occurs only through the shared data, which leads to a form of strong coupling, and possibilities of bottlenecks [10, 8].

In *federated multi-agent systems* [9], complex agents called facilitators, organize the work among simpler agents that notify the facilitator of the tasks they are able to handle. Receiving a request, the facilitator finds a competent agent to execute the task. Some examples of such architectures are: the ABSI [17]; the SHADE matchmaker [12] used in the SHADE project [13]; and the Knowledgeable Community [14]. Facilitator architectures rationalize communication resources. However, because a facilitator operates as a bridge between agents, its failure may prevent communication between the agents.

"Democratic" multi-agent systems gather agents which all have the same status, like in the ARCHON project [4], or in our approach [16].

Agents performing collective actions [2, 19] communication is an important issue and is usually asynchronous, performed by means of various protocols. For example, the Knowledge Query and Manipulation Language (KQML) is a language using performatives that may express their beliefs, needs, and modalities