ALBA: A Generic Library for Programming Mobile Agents with Prolog

Benjamin Devèze, Caroline Chopinaud, and Patrick Taillibert

Thales Airborne Systems
2 avenue Gay-Lussac, 78851 Elancourt - France
{firstname.lastname}@fr.thalesgroup.com

Abstract. This paper presents ALBA, a generic library dedicated to the commissioning of mobile agents written in Prolog. This library offers a handful of mechanisms for autonomous agent creation, execution, communication and mobility, whose implementation strongly respects the principles of robustness, decentralization of data, flexibility and genericity. In this perspective, the following paper mainly focuses on ALBA architecture and implementation with an emphasis on the technical choices which were made to provide these essential features. It therefore presents an innovative migration protocol, a research algorithm of agents solely identified by their names. It exposes some considerations about communication handling in a fully decentralized environment and some ideas towards a distributed modularity of systems. It also highlights an agent model, called Reasoning Threads, that is being used on top of ALBA to program cognitive agents.

1 Introduction

Since the emergence of multiagent systems (MAS), the corresponding research community has grown considerably and has been hard at work to provide agent communication standards mainly based on the speech act theory [2]. Important efforts have been made to formalize the main characteristics of agents, focusing on agent-oriented languages able to describe the behaviour of an intelligent agent. An abundant literature can be found about MAS related concepts like social attitudes, organization, cooperation or autonomy.

Unfortunately, the design of practical tools that can effectively support MAS programming and deployment appears to miss the necessary maturity to be widely adopted and used for large-scale industrial applications. A remaining gap persists between theories and concrete implementations that prevents us from taking the full benefits of this technology.

In order to demonstrate the added-value of the multiagent paradigm and to convince the remaining sceptic researchers and industrials, it is necessary to provide efficient programming constructs that facilitate the implementation of the essential concepts used in MAS. It is now admitted that MAS deal with flexibility, robustness, decentralization, modularity and scalability [21]. This should be kept in mind when developing new tools in this domain, so as not to alter these valuable qualities.

R.H. Bordini et al. (Eds.): ProMAS 2006, LNAI 4411, pp. 129–148, 2007.

Despite the numerous approaches and platforms architectures that have been proposed for agents commissioning, there is no general agreement on a particular method that would combine all the advantages of the agent paradigm. Platforms are often too centralized and often rely on imperative object-oriented languages, like Java, for agents implementation, which are not so well suited for this task [18]. It is especially the case for the kind of applications we have in mind in our group, that could be characterized as an attempt to apply the multiagent methodology to what is generally called real-time applications built on multitasked operating systems. These applications, which in our case concern mission systems embedded in aircrafts (sea or ground surveillance, coordinated observation missions by UAV -Unmanned Air Vehicles-, etc.) are generally complex since they not only manage a lot of tasks simultaneously but also rely upon complex algorithms whose duration cannot always be predicted (Artificial Intelligence approaches are more and more often necessary to implement the requirements of the new mission systems in preparation). To explore the various possible ways to make these systems evolve from a multitask to a multiagent perspective, a powerful implementation language capable of rapid agent model experimentation and AI algorithms development was needed. The most simple infrastructure was also needed in order to be able to easily merge our agents in an existing system and prove, without a complete redesign, that the multiagent approach was an alternative to present practices. That is the reason why ALBA has been designed as a library rather than a platform as is most often the case. Mobility was also a point since it makes it really easier to commission our agents on changing environments (all agents can be created on one computer -whose access is easier or devoted to our experiments- and then dynamically moved to the available computers at the time of the experiment).

Section 2 exposes the main reasons that led to the development of ALBA. Section 3 gives a general overview of the main aspects of our system that, in a way, put it apart from the majority of other tools. In section 4 we go thoroughly into some practical considerations about communications handling and in section 5 a dynamic agent search algorithm is detailed. Section 6 explains in depth the migration protocol and offers some views about agents mobility. Section 7 introduces a specific agent model, called Reasoning Threads, that is being used on top of ALBA to program cognitive agents illustrating a possible usage of the library. Section 8 describes some industrial applications already implemented using ALBA and the Reasoning Threads. Finally, sections 9 and 10 draw the main lessons of our proposals and discuss related and future works.

2 Why a New Platform?

Recent years have seen a considerable growth in the number of platforms, with a current total of over 100 products. It is then legitimate to ask why it has been necessary for us to develop a new one. The first exigence we had was that the platform had to allow the commissionning of agents written in Prolog.