A Tool Architecture to Verify Properties of Multiagent System at Runtime

Denis Meron[1] and Bruno Mermet[2]

[1] LITIS, University of Le Havre, France
[2] GREYC, University of Caen, France
Denis.Meron@univ-lehavre.fr
Bruno.Mermet@univ-lehavre.fr

Abstract. This paper describes an architecture allowing to verify properties of a multiagent system during its execution. This architecture is the basis of our study whose goal is to check at runtime, if agents and more generally multiagent systems satisfy requirements. Considering that a correct system is a system verifying the properties specified by the designer, we are interested in the "property" notion. That is why we give here a definition of "property" and we present an architecture to validate them. The architecture, a multiagent system itself, is based on a set of agents whose goals are to check at runtime the whole system's properties. So after a brief description of the "property" notion, we describe our architecture and the way to check systems.

1 Introduction

Today, a huge number of scientific areas use computers either to solve problems or to design simulations. Indeed, problems become more and more complex and it is asked to computer science to study and solve such problems. As a consequence, multiagent systems (MAS thereafter) are considered as a solution to take this complexity into account, and are more and more used to design new softwares. However, MAS development is a complex process because many autonomous entities evolve concurrently and asynchronously. So, as it is performed for softwares developed using the object paradigm, multi-agent programming needs to validate and verify MAS developed. In other words, when a MAS is developed, it is required to check whether the system has been correctly developed.

This idea is not new and several research have already been performed on verification and validation of MAS [6,20,4,18,10]. Among them, few ones deal with the way to prove the correctness of MAS either by theorem proving [20] or by model-checking [1]. However, the proof may be difficult to perform especially when the problem is massively distributed with many interactions between agents. Furthermore, model-checking costs time and the problem must be reduced to a finite one, with a limited number of states (even if unbounded model-checking using binary decision diagrams reduces this limitation). A second way to validate softwares is to perform tests, thanks to an architecture making testing feasible. Contrary to the model-checking, tests are not required to be exhaustive. As a consequence, the confidence brought by tests is reduced, but is easier to obtain. But in the case of MAS, which are open systems, the coverage of test is difficult to

R.H. Bordini et al. (Eds.): ProMAS 2006, LNAI 4411, pp. 201–216, 2007.

assess and is often near zero. Moreover, because of the asynchronous execution schema of the agent, a given test scenario is not guaranteed to give the same result twice.

In this context, in order to check that a MAS was correctly developed, it is useful to design an architecture allowing to test the final system on the fly. Indeed, we need to design a set of agents whose goal is to detect and highlight any effective behavior that is not compatible with the expected behavior. Assuming that a well implemented MAS has the properties that the designers implemented as a principle, we think that checking a system must rely on the verification of these properties. Moreover, to validate a large scale of MAS, we think that the architecture needs to be independent of any system or agent oriented framework. The rest of this paper is structured as follows: section 2 presents some existing studies on system validation. Section 3 defines the notion of property. In section 4, the architecture we designed to check properties at runtime is exposed. Finally, Section 5 concludes the paper, with an indication of on-going work.

2 State of the Art

Since the birth of computer science, when designing software for critical systems (where life may depend on it) or just toys problems, the necessity to guaranty both the correctness of the design and the correctness of the system execution appeared as a major problem for developers. But most of agent oriented framework like JADE, AgentBuilder, Jack or MADKIT, don't give really issue for debugging [6]. So it appears that it is necessary to design an architecture allowing to check the integrity of the system. The idea is not new and several researches have already be done by Steven SAPHIRO [20], David FLATER [4], or also David POUTAKIDIS [18,19]. In the three next sections we will see the different ways they use to check systems.

2.1 Specification Language

Steven SHAPIRO *et al.* in [20] describe a language to specify MAS able to prove that a system was correctly designed. The language named CASL for Cognitive Agents Specification Language is a framework which has a mix of declarative and procedural components to facilitate the specification and the verification of multiagent systems. The main idea is to specify the mental states of an agent in a mathematical way, so it may be provable. More particularly the language allow to specify the agent information (such as knowledges and beliefs) and its goals. They have also developed a verification environment (CASLve) for their language based on the Prototype Verification System (PVS) [17] to make it easier to verify properties of CASL specification. The verification environment provides the user a comprehensive library of proof methods, or lemmas.

This method is very interesting but the proof of a complex system is very difficult to obtain. Moreover, they are not interested in problems that may appear at run time or by the effective agent behavior.

2.2 Behavior Study

Some studies rely on collecting information that represent the behavior of the system to control [21,16]. The focus in such debugging tools is on collection of information