Formal Development of Reactive Agent-Based Systems

P. Kefalas
CITY College, Greece

M. Holcombe
University of Sheffield, UK

G.Eleftherakis
CITY College, Greece

M.Gheorghe
University of Sheffield, UK

INTRODUCTION

Recent advances in both the testing and verification of software based on formal specifications have reached a point where the ideas can be applied in a powerful way in the design of agent-based systems. The software engineering research has highlighted a number of important issues: the importance of the type of modelling technique used; the careful design of the model to enable powerful testing techniques to be used; the automated verification of the behavioural properties of the system; and the need to provide a mechanism for translating the formal models into executable software in a simple and transparent way.

An agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives (Jennings, 2000). There are two fundamental concepts associated with any dynamic or reactive system (Holcombe & Ipate, 1998): the environment, which could be precisely or ill-specified or even completely unknown and the agent that will be responding to environmental changes by changing its basic parameters and possibly affecting the environment as well. Agents, as highly dynamic systems, are concerned with three essential factors: a set of appropriate environmental stimuli or inputs, a set of internal states of the agent, and a rule that relates the two above and determines what the agent state will change to if a particular input arrives while the agent is in a particular state.

One of the challenges that emerges in intelligent agent engineering is to develop agent models and agent implementations that are "correct." The criteria for "correctness" are (Ipate & Holcombe, 1998) the initial agent model should match the requirements, the agent model should satisfy any necessary properties in order to meet its design objectives, and the implementation should pass all tests constructed using a complete functional test-generation method. All the above criteria are closely related to stages of agent system development, i.e., modelling, validation, verification, and testing.

BACKGROUND: FORMAL METHODS AND AGENT-BASED SYSTEMS

Although agent-oriented software engineering aims to manage the inherent complexity of software systems (Wooldridge & Ciancarini, 2001; Jennings, 2001), there is still no evidence to suggest that any methodology proposed leads toward "correct" systems. In the last few decades, there has been strong debate on whether formal methods can achieve this goal. Software system specification has centred on the use of models of data types, either functional or relational models, such as Z (Spivey, 1989) or VDM (Jones, 1990), or axiomatic ones, such as OBJ (Futatsugi et al., 1985). Although these have led to some considerable advances in software design, they lack the ability to express the dynamics of the system. Also, transforming an implicit formal description into an effective working system is not straightforward. Other formal methods, such as finite state machines (Wulf et al., 1981) or Petri Nets (Reisig, 1985) capture the essential feature, which is "change," but fail to describe the system completely, because there is little or no reference to the internal data and how these data are affected by each operation in the state transition diagram. Other methods, like statecharts (Harel 1987), capture the requirements of dynamic behaviour and modelling of data but are informal with respect to clarity and semantics. So far, little attention has been paid in formal methods that could facilitate all crucial stages of "correct" system development, modelling, verification, and testing.