INTERNATIONAL JOURNAL OF COMPUTATIONAL INTELLIGENCE 1(1) 2004 1-12

Modeling Biology Inspired Reactive Agents Using X-machines

George Eleftherakis, Petros Kefalas, Anna Sotiriadou, and Evangelos Kehris

***Abstract*—**Recent advances in both the testing and verification of software based on formal specifications of the system to be built 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 modeling 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; the need to provide a mechanism for translating the formal models into executable software in a simple and transparent way. This paper introduces the use of the X-machine formalism as a tool for modeling biology inspired agents proposing the use of the techniques built around X-machine models for the construction of effective, and reliable agent-based software systems.

Keywords— Biology Inspired Agent, Formal Methods, X-machine

I. Introduction

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 [1]. There are two fundamental concepts associated with any dynamic or reactive system, such as an agent, that is situated in and reacting with some environment [2]. The environment itself must be defined in some precise, mathematical way. The agent will be responding to environmental changes by changing its basic parameters and possibly affecting the environment as well. Thus, there are two ways in which the agent reacts, i.e. it undergoes internal changes and it produces outputs that affect the environment.

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 emerge in intelligent agent engineering is to develop agent models and agent implementations that are "*correct*". According to Holcombe and Ipate [2], the criteria for "correctness" are:

- the initial agent model should match with 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 three stages of agent system development, i.e. *modelling*, *verification* and *testing*.

Although agent-oriented software engineering aims to manage the inherent complexity of software systems [3], there is still no evidence to suggest that any method proposed leads towards "correct" systems. In the last few decades, there has been a strong debate on whether *formal methods* can achieve this goal. Academics and practitioners adopted extreme positions either for or against formal methods [4]. It is, however, apparent that the truth lies somewhere between and that there is a need for use of formal methods in software engineering in general [5], while there are several specific cases proving the applicability of formal methods in agent development, as we shall see in the next section.

Software system specification has centred on the use of models of data types, either functional or relational models such as *Z* or *VDM* or axiomatic ones such as *OBJ*. 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* or *Petri Nets* capture the essential feature, which is "change", but fail to describe the system completely, since there is little or no reference at all to the internal data and how this data is affected by each operation in the state transition diagram. Other methods, like *Statecharts*, capture the requirements of dynamic behaviour and modelling of data but are rather 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

Manuscript received November 5, 2004.

George Eleftherakis is with the Computer Science Department, CITY college Thessaloniki, Greece, Affiliated Institution of the University of Sheffiled, UK (corresponding author; phone: +30-2310-275575; fax: +30-2310-287564; e-mail: eleftherakis@city.academic.gr).

Petros Kefalas is with the Computer Science Department, CITY college Thessaloniki, Greece, Affiliated Institution of the University of Sheffiled, UK (e-mail: kefalas@city.academic.gr).

Anna Sotiriadou is with the Computer Science Department, CITY college Thessaloniki, Greece, Affiliated Institution of the University of Sheffiled, UK (e-mail: sotiriadou@city.academic.gr).

Evangelos Kehris is with the Technological Education Institute (TEI) of Serres, Greece (e-mail: kehris@teiser.gr).