Agent-Oriented Software Engineering: The State of the Art

Michael Wooldridge[†] and **Paolo Ciancarini**[*]

[†] Department of Computer Science
University of Liverpool
Liverpool L69 7ZF, UK
M.J.Wooldridge@csc.liv.ac.uk

[*] Dipartimento di Scienze dell'Informazione
University of Bologna
Mura Anteo Zamboni 7, 47127 Bologna, Italy
ciancarini@cs.unibo.it

Abstract. Software engineers continually strive to develop tools and techniques to manage the complexity that is inherent in software systems. In this article, we argue that *intelligent agents* and *multi-agent systems* are just such tools. We begin by reviewing what is meant by the term "agent", and contrast agents with objects. We then go on to examine a number of prototype techniques proposed for engineering agent systems, including methodologies for agent-oriented analysis and design, formal specification and verification methods for agent systems, and techniques for implementing agent specifications.

1 Introduction

Over the past three decades, software engineers have derived a progressively better understanding of the characteristics of complexity in software. It is now widely recognised that *interaction* is probably the most important single characteristic of complex software. Software architectures that contain many dynamically interacting components, each with their own thread of control, and engaging in complex coordination protocols, are typically orders of magnitude more complex to correctly and efficiently engineer than those that simply compute a function of some input through a single thread of control.

Unfortunately, it turns out that many (if not most) real-world applications have precisely these characteristics. As a consequence, a major research topic in computer science over at least the past two decades has been the development of tools and techniques to model, understand, and implement systems in which interaction is the norm.

Many researchers now believe that in future, computation itself will be understood as chiefly as a process of interaction. This has in turn led to the search for new computational abstractions, models, and tools with which to conceptualise and implement interacting systems.

Since the 1980s, software agents and multi-agent systems have grown into what is now one of the most active areas of research and development activity in computing

P. Ciancarini and M.J. Wooldridge (Eds.): AOSE 2000, LNCS 1957, pp. 1–28, 2001.

generally. There are many reasons for the current intensity of interest, but certainly one of the most important is that the concept of an agent as an autonomous system, capable of interacting with other agents in order to satisfy its design objectives, is a natural one for software designers. Just as we can understand many systems as being composed of essentially passive objects, which have state, and upon which we can perform operations, so we can understand many others as being made up of interacting, semi-autonomous agents.

Our aim in this article is to survey the state of the art in agent-oriented software engineering. The article is structured as follows:

- in the sub-sections that follows, we provide brief introductions to agents and multi-agent systems, and comment on the relationship between agents and objects (in the sense of object-oriented programming);
- in section 2, we survey some preliminary *methodologies* for engineering multi-agent systems — these methodologies provide structured but non-mathematical approaches to the analysis and design of agent systems, and for the most part take inspiration either from object-oriented analysis and design methodologies or from knowledge-engineering approaches; and finally,
- in section 3, we comment on the use of *formal* methods for engineering multi-agent systems.

We conclude the main text of the article with a brief discussion of open problems, challenges, and issues that must be addressed if agents are to achieve their potential as a software engineering paradigm. In an appendix, we provide pointers to further information about agents.

1.1 What are Agent-Based Systems?

Before proceeding any further, it is important to gain an understanding of exactly what we mean by an agent-based system. By an *agent-based system*, we mean one in which the key abstraction used is that of an *agent*. Agent-based systems may contain a single agent, (as in the case of user interface agents or software secretaries [50]), but arguably the greatest potential lies in the application of *multi*-agent systems [5]. By an *agent*, we mean a system that enjoys the following properties [75, pp.116–118]:

- *autonomy*: agents encapsulate some state (that is not accessible to other agents), and make decisions about what to do based on this state, without the direct intervention of humans or others;
- *reactivity*: agents are *situated* in an environment, (which may be the physical world, a user via a graphical user interface, a collection of other agents, the INTERNET, or perhaps many of these combined), are able to *perceive* this environment (through the use of potentially imperfect sensors), and are able to respond in a timely fashion to changes that occur in it;
- *pro-activeness*: agents do not simply act in response to their environment, they are able to exhibit goal-directed behaviour by *taking the initiative*;