Agent-Oriented Software Engineering

Stefan Bussmann[*] Paolo Ciancarini[†] Keith Decker[‡]
Michael Huhns[#] Michael Wooldridge[×]

[*] Daimler-Chrysler AG, Forschung und Technik
Alt Moabit 96a, Berlin 10559, Germany

[†] Dipartimento di Scienze dell'Informazione
University of Bologna, Mura Anteo Zamboni 7, 47127 Bologna, Italy

[‡] Department of Computer and Information Sciences
University of Delaware, Newark, DE 19716-2586, USA

[#] Electrical and Computer Engineering Department
University of South Carolina, Columbia, SC 29208, USA

[×] Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, United Kingdom

1 Introduction

The ATAL workshops focus on the links between the theory and practice of intelligent agents. One aspect of this, which is steadily growing in importance, is the idea of agent technology as a software engineering paradigm. Previous ATAL workshops have had special tracks on programming languages for agent-oriented development, and methodologies for agent system development. ATAL-99 aims to build on this experience by focussing on the wider issues of agents as a software engineering paradigm.

Software engineering has been a major concern in computer science since the late 1960s and early 1970s. Despite major advances in our understanding of software since then, it remains a major concern in computing today. Progress in software engineering has been made primarily through the development of increasingly powerful abstractions with which to understand and implement complex systems. Consider the progression from procedural abstraction, through abstract data types, to object-oriented programming. Each step represents a move away from a machine-oriented view of software towards one that more closely reflects the way that we ourselves understand the world. The state of the art – object-oriented development – allows us to model and develop complex systems from essentially passive entities that interact with one another in simple ways.

In tandem with work on developing more powerful abstractions, software engineers have also 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. Systems that contain many dynamically interacting components, each with their own thread of control, 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. Indeed, many researchers now believe that in future, computation itself will be understood as chiefly as a process of interaction.

Since the 1980s, intelligent agents and multi-agent systems have grown into what is now one of the most active areas of research and development activity in computing 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. Just as we can understand many systems as being comprised 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.

This recognition has led to the growth of interest in agents a new paradigm for software engineering. In this special track of ATAL-99, we seek to examine the credentials of agent-based approaches as a such a software engineering paradigm, and to gain an insight into what agent-oriented software engineering will look like. To this end, the panelists were asked to respond to the following questions:

1. The "OO mindset" contains about half a dozen key concepts – class, instance, encapsulation, inheritance, polymorphism, and so on. In your view, what are the key concepts in the "agent-oriented" mindset? If you had to identify just one, then what would it be and why?
2. Agent-based solutions are not appropriate to all applications. For example, it doesn't make sense to have agents performing extremely fine-grained computations (such as addition and subtraction) – the overheads make such an approach unworkable. One of the keys to the success of agent-oriented software engineering is therefore to identify the application characteristics that indicate an agent-based solution. So, how do we identify what should and should not be modelled/implemented as an agent? What are the key features you look for in a problem that suggest an agent-based solution?
3. Over the past few years, there has been an increasing trend in the object-oriented community towards the development of "agent-like" features. Examples include distributed objects (CORBA, RMI), applets, and mobile object systems. This trend is likely to continue at least in the short term. Given this, how does an agent-oriented software engineering view sit in relation to other software paradigms, in particular, object-oriented development? What are the key attributes of agent-oriented development that make it *unique* and *distinctive*?
4. If you have a particular approach to agent-oriented development, then describe this approach, and give a *short* case study of a realistic problem that you have tackled using it. Describe the major software engineering principles that you learnt from this experience, and outline the software engineering issues that your experience have raised.
5. How will the state of the art in software engineering in ten (twenty) years time differ from software engineering today? What will current activities in agent R&D have contributed to this state of the art? What are the major issues that the agent R&D community must address in order to realise this vision?