Preface

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. Indeed, many researchers now believe that in the future, computation itself will be understood chiefly as a process of interaction.

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 in general. 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 satisy 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 a 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 as a new paradigm for software engineering. The aim of the AOSE-2000 workshop, held at the ICSE-2000 conference in Limerick, Ireland, in June 2000, was to investigate the credentials of agent-oriented software engineering, and to gain an understanding of what agent-oriented software engineering might look like.

Some 32 papers were submitted to the workshop, and after refereeing, about half were accepted for presentation. After the workshop, these papers were revised in light of the discussions at the workshop and, together with a selection of invited papers (by Bussmann, Petrie, Rana, and Shehory), these revised papers make up the volume you are now reading.

We are convinced that agents have a significant role to play in the future of software engineering. This book offers insights into the issues that will shape that future.

September 2000

Paolo Ciancarini
Michael Wooldridge

Organising Committee

Paolo Ciancarini	(CHAIR)	University of Bologna, Italy email ciancarini@cs.unibo.it
Michael Wooldridge	(CO-CHAIR)	University of Liverpool, UK email M.J.Wooldridge@csc.liv.ac.uk

Programme Committee

Carlos Angel Iglesias Fernandez	Spain
Dennis Heimbinger	Germany
Michael Huhns	USA
Nicholas Jennings	UK
Liz Kendall	Australia
Yannis Labrou	USA
Jaeho Lee	Korea
James Odell	USA
Andrea Omicini	Italy
Jan Treur	The Netherlands
Jeffrey Tsai	USA
Robert Tolksdorf	Germany
Franco Zambonelli	Italy

Topics of Interest

The workshop invited the submission of all papers covering aspects of agent-oriented software engineering, but particularly the following:

- Methodologies for agent-oriented analysis and design
- Relationship of agent-oriented software to other paradigms (e.g., OO)
- UML and agent systems
- Agent-oriented requirements analysis and specification
- Refinement and synthesis techniques for agent-based specifications
- Verification and validation techniques for agent-based systems
- Software development environments and CASE tools for AOSE
- Standard APIs for agent programming
- Formal methods for agent-oriented systems, including specification and verification logics
- Engineering large-scale agent systems
- Experiences with field-tested agent systems
- Best practice in agent-oriented development
- Market and other economic models in agent systems engineering
- Practical coordination and cooperation frameworks for agent systems

We were particularly interested in papers that addressed 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? 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 suggests an agent-based solution?
2. 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, mobile object systems, and coordination mechanisms and languages. 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?
3. What is the impact of agent-oriented languages and tools on the software development process? How can legacy software architectures be integrated with agent- or multi-agent-oriented applications? Which specification, design, implementation, maintenance, or documentation systems and strategies have to be adopted in order to deal with agent-oriented issues?
4. Agent-based solutions are not appropriate to all applications. One of the keys to the success of agent-oriented software engineering is therefore to identify the application requirements that indicate an agent-based solution. What are the key properties that indicate an agent-based approach is appropriate?