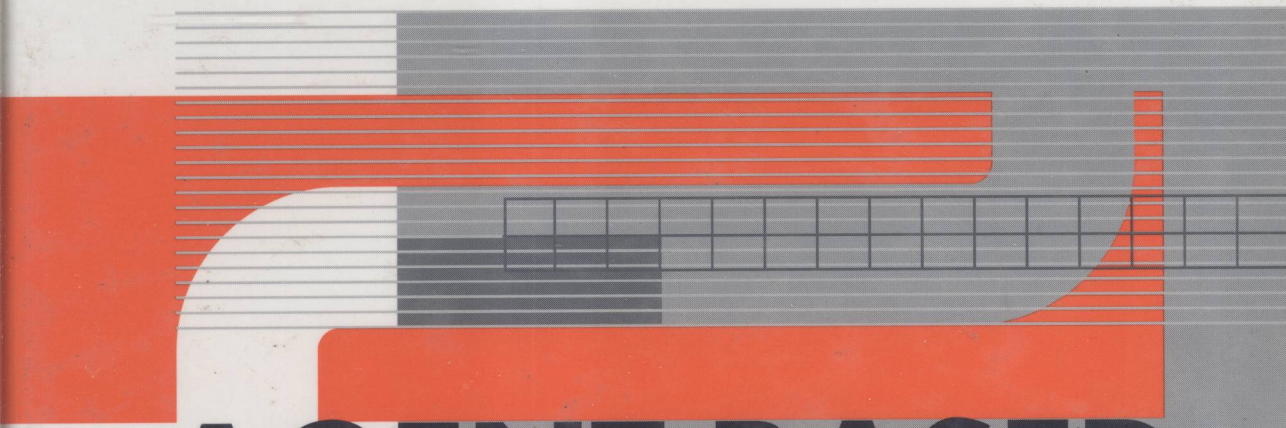
Michael LUCK • Ronald ASHRI • Mark D'INVERNO



AGENT-BASED SOFTWARE DEVELOPMENT

Agent-Based Software Development

Michael Luck
Ronald Ashri
Mark d'Inverno

Artech House
Boston • London
www.artechhouse.com

Library of Congress Cataloging-in-Publication Data

A catalog record for this book is available from the Library of Congress.

British Library Cataloguing in Publication Data

A catalog record of this book is available from the British Library.

Cover design by Yekaterina Ratner

685 Canton Street
Norwood, MA 02062

International Standard Book Number: 1-58053-605-0

10 9 8 7 6 5 4 3 2 1

Agent-Based Software Development

Agent-Oriented Systems
Series Editor: Michael Luck (University of Southampton, United Kingdom)

Overview:
Agent technology has seen a dramatic growth of interest in recent years and is being given added impetus by the integration of new technologies and applications into the mainstream of commercial software. For example, the World Wide Web, e-commerce, Grid computing, and distributed object technologies all suggest that agent technology has an important role in modern computing. This is now also true at the level of commercial and industrial take-up of the technology, with an increasing number of companies investing in agent products and agent development solutions and a number of standards initiatives underway for some time.

The Agent-Oriented Systems Series aims to cover important advances in agent-based computing concerning development methodologies, tools and techniques, standards, and other aspects of agent-oriented systems development. This series will be of interest to all software engineers, IT personnel, computer scientists, and technologists working in the area of agent-based computing and in related application domains as well as related areas of distributed computing, semantic Web, Grid computing, and so forth. It should also provide a valuable set of resources addressing the state of the art in agent-oriented systems development. The series topics include, but are not limited to:

- Agent-oriented software engineering;
- Agent standards;
- Development case studies;
- Infrastructure for agent-based systems;
- Agent platforms and tools;
- Agent-oriented information systems.

For a listing of recent titles in the *Artech House Computing Library,*
turn to the back of this book.

Preface

The field of agent-based computing is large and growing, and set to underpin much of the infrastructure of the next generation of computing that seeks to address issues in ambient intelligence, pervasive and ubiquitous computing, Grid computing, the semantic Web, e-business and many other areas. However, in order to take full advantage of the benefits that agent technology brings, the agent paradigm ought to be adopted at the stage of system design. This suggests the use of agent-oriented methodologies for the development of agent systems, adherence to standards intended to ensure interoperability, and the adoption of relevant tools and techniques to aid the process. This book provides a thorough exploration of all these areas, and brings together the different aspects of agent-oriented development in one coherent text.

OVERVIEW OF THE BOOK

Chapter 1 starts by discussing the key concepts underlying agent systems, and reviewing the motivation for their development through a consideration of the underlying computational context. In Chapter 2, these basic notions of agents are fleshed out through a review of several important agent architectures and key multiagent systems. The aim of this chapter is simply to provide an adequate grounding for the issues that are covered in the subsequent chapters on more specific development-oriented work. It can be taken simply as a necessary preliminary chapter, or as a broad review of the field in its own right.

The main part of the book begins in Chapter 3 with a description of a selection of tools that are available for use in agent systems development. These include commercial systems as well as more academic offerings, with each compared to the others along major axes. Complementing this, Chapter 4 presents a detailed analysis of the various different methodologies and notations that have been developed for the design of agent-based systems. It provides a review of the salient techniques, many of which are now mature, and some in regular use.

Chapter 5 addresses the context for the development of agent systems through the use of *standards*. It describes the work that has been done in the IT community to support interoperability, primarily through the efforts of the Foundation for Intelligent Physical Agents (FIPA), but also

other standards organizations. Arguably, the development of such standards must occur before the widespread adoption of agent technologies. Chapter 6 similarly considers the context, but through the availability of standard technologies that can be used to provide much of the functionality that underlies systems of autonomous interacting agents. Technologies such as Jini and Web services all offer valuable aspects that can be leveraged in developing agent systems.

Finally, Chapter 7 ends the book by reviewing the different resources that are available for consultation for more detail on the various aspects of agent-oriented development, and on agent technologies more generally. In a fast-moving field, Web resources and other sources of information and support can be critically important. Fortunately, there are many possible places to seek further information. One of these is the Web site that accompanies this book, at `www.agentdevelopment.com`, which contains pointers to numerous reference sites, and a range of relevant resources.

ACKNOWLEDGMENTS

This book resulted from discussions with Tim Pitts of Artech House, in the context of the emerging maturity of the field of agent-oriented software development. Tim has provided support and encouragement, and we appreciate his efforts in pushing the project forward. Similarly, the day-to-day work of badgering the authors and dealing with our delays and problems was handled by Louise Skelding and Tiina Ruonamaa, who coped with us admirably during the development of the manuscript. We are grateful to them both for their patience.

Most of the work in producing this book has been undertaken by the three authors, but the book itself would not offer anything like the coverage and depth of analysis had the authors of Chapter 4 and Chapter 5 not agreed to contribute.

A special mention must be made of Serena Raffin, who gave up her own time to help generate many of the diagrams included in the chapters of this book. Without her, we would never have made it.

As is often the case, the environments in which we work provide excellent contexts in which to undertake an endeavor like the one here. We appreciate and recognize that our departments and universities not only enabled us to write this book, they also provided the experience of research and development that we describe throughout. Particular thanks should go to Paul Howells, Nick Jennings, Luc Moreau, Terry Payne, Mark Priestly, and Steve Winter, who provided support from within our own institutions. Thanks also to Florence and Musa for making the days working in London much more pleasant than they would otherwise have been.

Personally and individually, we must thank those around us for bearing with us during the stress and tension of writing late and early, when we should have been spending our time with them. And for those who didn't bear with us, we love you anyway. Special thanks to our parents, Harry and Dalia, Andreas and Photini, and Ray and Pauline.

Thanks also to Katia, Carla and Titta (and Lilly), Nikolas and Panikos, Daniel, Christos "Kozias" Markides, Alex and Olivia, Abha, Jo and Leo, Jeremy and Monica (and Francesca and Martina), Dan and Franky (and Poppy and Joseph), Alex, Rachel, Sharon and Ben (and Daniel, Jonathan, and Francesca), Orly, Dawn, Lucy, Olivia, John and Philly (and Cameron), Eileen, Susan, Nicky, Serena,

Liz, Jane, Clare, Lucia, Mary, Carla, Michelle, Dolly and Candy (and Tiger and Hector), Eli, Phil and Geddy (and Michael and Daisy), Paul Rooney, Gi and Philly (and Beany, Tatti, Oscar, Flo, and Rose), Andy and Mel (and Thomas and Max), Neil and Susi (and Alex and Olly), Tim and Lisa (and Joseph), Dave and Jo (and Saskia), Karen and Jason (and Tali), Val and Bill, Vijay and Lisa, Chris and Catherine (and Eliza), Emma Barrett, Ali and Dave, Chris and Secha, Nicki and Tony, Nicki and Giles, Chris and Sylvia, Tony and Leslie, Polly and Antonia, Eli, Lisa Rigg, Mike Freeman, Mike Bacon, Jenny Goodwin, Kellie Samuda, Mo City, Hutch, Gib, Lawso, Dicks and the Lab Bar in Soho, the Guinea Grill, the weekenders cricket club, and Choral Clench, all of whom have distracted us so easily.

Michael Luck, Ronald Ashri,
and Mark d'Inverno
Southampton and London
January 2004

Contents

Chapter 1

Agent-Based Computing

1.1 OPEN AND DYNAMIC COMPUTING ENVIRONMENTS

Computer systems in the twenty-first century are dramatically different from those that have been the norm over the last 50 years. Unlike previous changes, which have seen vast improvements in computing capacity and power, modern computing is essentially defined by the *interconnection* of computers, and all that it brings. The advent of the World Wide Web 10 years ago offered a radically new take on the *information society*, releasing information and making it available to all. Somewhat more crucially, it also provided the basic infrastructure for the dynamic provision of on-line *services*, which are only now beginning to take root with real substance and significance.

Against this technological background, 2003 was also the year in which Internet penetration in Western Europe and the United States was expected to pass 60%. In Europe, the European Commission has launched its *eEurope* initiative, which aims to bring every citizen, home, school, and business online to create a digitally literate Europe. In short, the maturity of the technology, and its penetration of all aspects of society, have converged to suggest not just new kinds of systems, but also their implementation and deployment on a scale that offers dramatic new opportunities and problems. This book is concerned with one way—and one that is being seen by many as a key underpinning technology for the next generation of computing—of meeting these opportunities and addressing these problems.

As indicated above, the move from a focus on the individual standalone computer system to a situation in which the real power of computers is realized through distributed, open, and dynamic systems has radically changed the nature of software and its development. The key difference now is that there is an environment made up of computers in the infrastructure, in support systems, and embedded in a vast array of devices, as well as on the traditional desktop. More importantly, these computers are typically networked and can interact dynamically to form new configurations of systems to suit current needs. The flexibility that such abilities offer, although increasingly taken for granted, is set to change the way we do business, undertake science, and manage our everyday activities. However, the characteristics of dynamic and open environments in which, for example, heterogeneous systems must interact, span organizational boundaries, and operate effectively within

rapidly changing circumstances and with dramatically increasing quantities of available information, suggest that improvements on the traditional computing models and paradigms are required.

In particular, the need for some degree of autonomy, to enable components to respond dynamically to changing circumstances while trying to achieve overarching objectives without the need for user intervention, is seen by many as fundamental. In practical developments, Web services, for example, now offer fundamentally new ways of operating through a set of standardized tools, and support a service-oriented view of distinct and independent software components interacting to provide valuable functionality. In the context of such developments, *agent technologies* have become some of the most valuable tools that can be used to tackle the emergent problems, and to manage the complexity that arises.

Agents can be viewed as autonomous, problem-solving computational entities capable of effective operation in dynamic and open environments. They are often deployed in environments in which they interact, and possibly cooperate, with other agents (including both people and software) that may have conflicting aims. These are exactly the kinds of characteristics that are needed in the new computational environments. Agent-based systems have emerged over the past 10 to 15 years, from a convergence of technologies in distributed object systems and distributed artificial intelligence, and have seen rapid and dramatic growth both academically and commercially. Indeed, agent technologies are already providing real benefits in a diverse range of business and industry domains, spanning manufacturing, supply chain management, and B2B exchanges, for example.

1.2 OBJECT TECHNOLOGIES

The relation of agents to objects has caused difficulty for some in understanding what it is that makes agents distinct. While object-orientation as a programming paradigm has achieved much success, and offers a valuable abstraction for the development of complex systems, agents provide a different and *higher* level of abstraction. Like objects that provide encapsulation of state and behavior, agents also encapsulate these properties. However, objects are essentially passive in nature—they have no choice as to whether or not they interact, and are simply *invoked* by other objects to perform particular tasks or execute particular functionality. By contrast, agents have the ability to *decide* for themselves whether to participate in computational activity, and whether to perform the desired operation. This is the fundamental distinction that marks out agents as distinct by virtue of their *autonomy*. It is this autonomy that is also responsible for providing the flexibility that is needed for open and dynamic environments. If behavior is predetermined and is guaranteed when invoked, then the ability to provide flexible responses in the light of changing circumstances is severely curtailed, if not ruled out entirely.

In terms of modeling, objects can be regarded as a valuable way to view the world. Yet an agent-based approach offers a much more natural representation of real-world systems in which different individuals interact according to their own agenda and priorities. They then can come together to achieve overarching objectives that might not, or not as easily, be achieved by the individuals alone, but they do so when it is appropriate. When the goals of individual agents are closely aligned, and if they are completely benevolent and honest (or veracious) so that they always respond to requests made of them, and always provide information when queried, then the resulting systems may come

close to resembling an object-oriented system. In these cases, the object-oriented paradigm may be adequate, but this is a particular configuration of agents that is unlikely to provide the flexibility that may be required in modern computing environments.

In short, agents can be distinguished from objects in that they are autonomous entities capable of exercising choice over their actions and interactions. Agents cannot, therefore, be directly invoked like objects. However, they may be constructed using object technology. Moreover, agents typically run in their own thread of control, as opposed to standard object systems, which have one thread.

Object orientation is also relevant when considering how to develop agent systems. The question of where agent development methodologies fit in, and the extent to which they are needed in addition to existing object-oriented methodologies, depends on the approach taken. If agents provide a *programming* paradigm to rival object-oriented programming, then methodology is clearly of central importance, since the program and the programming themselves are informed by the agent paradigm. If, instead, they provide a paradigm or metaphor for design, then methodology is certainly important, but the nature of that methodology is likely to be related to current object-oriented approaches. In this case, however, the agent approach will require significant modifications from a standard object-oriented approach to address the agent abstractions and interactions. In either case, however, methodology is a vital issue that must be considered in some detail in order to support agent-orientation. In particular, the object-oriented paradigm does not address issues of developing software that exhibits flexible autonomous behavior.

There has recently been a good degree of work aimed at addressing these concerns, which are the focus of Chapter 4, but more work remains to be done before the approach is accepted as part of the mainstream. Indeed, as the field matures, the broader acceptance of agent-oriented systems will become increasingly tied to the availability and accessibility of well-founded techniques and methodologies for system development.

1.3 BASIC NOTIONS OF AGENTS

The introduction of the notion of agents is partly due to the difficulties that have arisen when attempting to solve problems without regard to a real external environment or to the entity involved in that problem-solving process. Thus, though the solutions constructed to address these problems are in themselves important, they can be limited and inflexible in not coping well in real-world situations. In response, agents have been proposed as *situated* and *embodied* problem-solvers that are capable of flexible and effective operation in complex environments. This means that the agent receives input from its environment through some sensory device, and acts so as to affect that environment in some way through effectors. Such a simple but powerful concept has been adopted by many branches of computing because of its usefulness and broad applicability.

However, a recurrent theme that is raised in one form or another in many different contexts is the lack of consensus over what it is that actually constitutes an agent. Certainly, the immediately engaging concepts and images that spring to mind when the term is mentioned are a prime reason for the popularization of agent systems in the broader (and even public) community, and for the extremely rapid growth and development of the field. Indeed the elasticity in terminology and definition of agent

concepts has led to the adoption of common terms for a broad range of research activity, providing an inclusive and encompassing set of interacting and cross-fertilizing subfields. This is partly responsible for the richness of the area and for the variety of approaches and applications.

Despite some healthy debate over precise definitions of agenthood, it is clear that there is now a generally accepted understanding of agents as computational entities that are capable of exhibiting flexible behavior in dynamic and unpredictable environments. This understanding is important, and it provides an operational basis for what this book is about, and motivates the effort directed towards the development of agent technologies and agent-based systems. In more specific terms, however, we can drill down and identify two distinct views of agents. The *weak notion* of agents provides a characterization that enumerates four properties regarded as necessary and sufficient for agenthood:

- Autonomy: agents must be self-starting and independent entities that are able to function without direct programmer or user intervention.
- Reactiveness: agents can monitor their environments and respond quickly and effectively to changes in those environments.
- Proactiveness: agents have overarching goals that direct behavior over longer periods of time towards achieving complex tasks.
- Social ability: since agents operate in dynamic and open environments with many other agents, they must have the ability to interact and communicate with these others.

Since this characterization of agents was originally provided in 1994 [1], it has been the subject of much discussion, and several authors have provided alternative characterizations with additional properties. These include, for example, the ability to learn, mobility, the requirement that agents are benevolent or honest (an unlikely requirement in open environments), that they are rational, and many others.

The *strong* or *intentional* notion of agents also requires agents to be based around control architectures comprising mental components such as beliefs, desires, and motivations. While the stance adopted throughout this book is broad and open, and avoids the dogma that can creep into these kinds of discussions, it will be seen very clearly, especially in the initial discussion of agent architectures, that these notions can be important and valuable as a metaphor that leads to the provision of effective and flexible control and behavior.

The broad area of understanding agenthood has merited several efforts that explore the area in some depth, including encompassing agent frameworks [2, 3] and agent taxonomies [4], which go some way to identifying the key features of agent systems and the characteristics of the different branches of the field. In attempting to distinguish agents from programs, Franklin and Graesser constructed an agent taxonomy [4] aimed at identifying the key features of agent systems in relation to different branches of the field. Their aim, amply described by the title of the paper, "Is It an Agent or Just a Program?", highlights the problem of whether there is value in the notion of agents. The definition provided, that an "autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to affect what it senses in the future," serves to distinguish some nonagent programs from agents through the