The Art of Agent-Oriented Modeling

Leon Sterling and Kuldar Taveter

The MIT Press
Cambridge, Massachusetts
London, England

This book was set in Times New Roman on 3B2 by Asco Typesetters, Hong Kong.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Sterling, Leon.
The art of agent-oriented modeling / Leon S. Sterling and Kuldar Taveter.
p. cm. — (Intelligent robotics and autonomous agents series)
Includes bibliographical references and index.
ISBN 978-0-262-01311-6 (hardcover : alk. paper) 1. Intelligent agents (Computer software) 2. Computer software—Development. I. Taveter, Kuldar. II. Title.
QA76.76.I58S757 2009
006.3—dc22 2008044231

10 9 8 7 6 5 4 3 2 1

Preface

The introduction of the personal computer changed society's view of computer systems. No longer was a computer a complicated machine doing a task, tucked away in a separate room or a far-off corner of an office or work environment. A computer was a machine on your desk that was ready for daily interaction. No longer was a computer only for specialist programmers—it was a tool to be used by virtually all office workers.

The rapid rise of the World Wide Web in the mid-1990s changed society's view still further. No longer was the personal computer restricted to a desk or a small office network. The machine on your desk could be part of a global network. In addition, the use of computers at home skyrocketed. Almost every home in the developed world acquired a computer. Laptops were mandated for school children in many schools. The volume and visibility of email addresses and domain names exploded.

In the mid-2000s, as this book is written, our interaction with computers is ever-expanding. Businesses depend on software for many—if not most—of their core activities. Cars and appliances have embedded software of which we are barely aware. Teenagers and young adults engage in social networking, and share music and videos on Web sites such as LimeWire and YouTube. We look up knowledge on Wikipedia. Skype is a common method for making telephone calls over the Internet.

So how do we develop computer software to interact with the ever-increasing complexity of the technical world and the increased fluidity of social organizations? It is not an easy task. Having been involved in research and teaching of software engineering and software development projects, we find a need for changing conceptual models. We want to develop software that is open, intelligent, and adaptive. We are becoming convinced that such software needs to be conceived, designed, and developed differently from existing applications that assume fixed requirements and an unchanging environment.

One consideration we believe is important is the need to take a systems view. The system of people and technology has a life cycle of inception, design, implementation, testing, and maintenance. Processes need to be understood for each of the stages of the life cycle. A systems view implies being able to take a holistic approach and to include considerations of all stakeholders.

In this book, we advocate an agent-oriented view. An *agent* is a concept and metaphor that is familiar for people with and without technical background or interest. An agent allows the blurring of the boundary between people and technology to allow more people to be engaged with the process of building software. Thinking in terms of agents can change the way that people think of software and the tasks it can perform.

The concept of an agent is not new to computing. Agents have been discussed in regard to artificial intelligence (AI) from its early days. More recently, agents have been discussed in the context of object-oriented technology, with many researchers and practitioners viewing them as an extension of object-oriented technology. Agent models have been added to established object-oriented modeling approaches, most notably based around UML (Unified Modeling Language). Agent programming extensions have been developed for languages such as Java and Prolog.

We prefer viewing an agent-oriented approach as different from an object-oriented approach. Agent-oriented models need to be described independently of object-oriented models, though sensible reuse of notation is advisable. The reason for differentiation is that people can be sometimes trapped into thinking only in ways familiar to them. As the saying goes, if all you have is a hammer, everything looks like a nail. In our experience, thinking in terms of agents requires a different mindset. Students and practitioners who insist in thinking about building from their existing (object-oriented) methods are less successful in developing agent-oriented applications.

Our interest in writing this book is to encourage a wide variety of stakeholders in the software development process to engage with an agent-oriented approach. This is a challenge. Software developers, software acquirers, software users, and software maintainers have different concerns. Models are needed that are understandable for all stakeholders. The variety of concerns suggests a variety of models. If the models are understandable, the stakeholder is more likely to be engaged. We believe that engagement throughout the process will improve the development of systems consisting of people and software that will interoperate and evolve successfully.

So why another book? One factor in the adoption of a technology is having textbook material in support. There has been an increase in agent-oriented theory, language, and methodology books over the past several years. This progress is to be encouraged. Nonetheless we feel that this book fills a niche not currently being filled.

There are three main features of this book that we feel add significantly to the literature. The first is the presentation of substantial new examples that range from requirements through design to deployment. The more examples that potential users of technology see, the more likely a technology is to be adopted. The second is the use of abstract agent-oriented models at various levels of abstraction and geared to different stakeholders. The models relate to a range of agent-oriented software engineering methodologies that are being actively developed. The third is related—namely, the emphasis on models rather than languages—and the philosophy that models can be mapped to a variety of deployment platforms, possibly including conventional languages, makes the concepts accessible to a wider audience. Our recent discussions with colleagues in both academia and industry have reinforced our appreciation for the need for different models for open, distributed domains.

There are often forces working against trying to show how an approach covers a range of implementation languages. Agent language vendors want companies to be locked into their technology. Indeed, one of the initial motivations for the research underlying this book was a request by an industrial partner on what agent models would be "future-proof." The industry partner had been burned by using an agent-based vendor platform that was no longer supported. Research groups also want to encourage others to use their specific methods. Though this motivation is understandable, the agent community will benefit from broadly supportable models.

Why do we use the word "art" in the title of the book? We have experienced modeling as a creative process that does not always follow clear-cut rules. Many decisions are left to the discretion of the modeler and his or her background and intuition. Modeling is also iterative. Usually one cannot—and should not—end up with final models right away. Even some of the models in the book would improve through further rounds of iteration. Given that we cannot give definitive advice on building "perfect" models, we settle for the more modest aim of providing guidance for creating models.

Having explained some of the motivation that prompted the book, we address the issue of timing. Why now? We have each been working with agent applications for the past ten years. Only now, in our opinion, do we perceive that agent concepts are known widely enough to allow agent-oriented modeling. Modeling tools are emerging: an essential development, if agent concepts are to be used by industry. Also, recent developments in the research areas of autonomic systems, event-based systems, and multiagent systems seem to indicate that the paradigm of peer-to-peer computing in the broad sense is gaining momentum.

Our conclusion that agent concepts have matured sufficiently has been developed through our delivering of seminars and courses on agent technology in academia and industry. We have been involved in teaching graduate students (and advanced undergraduates) for more than ten years. In 2000, Leon was involved in an Australian government project on the rapid diffusion of technology from academia to industry using agent technology as an example. With the benefit of hindsight, the project was premature. Providing students with the models presented in this book has greatly increased their ability to conceptualize and design agent-oriented systems. The models have been progressively polished over the past three years and are now ready for broader engagement. By the time the reader finishes reading this book, we hope that his or her capacity and confidence for the design and modeling of agent-oriented systems will have improved.

Leon Sterling, *Melbourne, Australia, November 2008*
Kuldar Taveter, *Tallinn, Estonia, November 2008*