AI Magazine Volume 23 Number 1 (2002) (© AAAI)

Heterogeneous Agent Systems A Review

P. Ravi Prakash

The notion of software agents has been around for more than a decade. Since its beginning, the definition of agent, like the definition of intelligence, has been quite controversial and often provoked hot discussions. Questions such as the following normally come up in such arguments: What is an agent? Should a piece of software be categorized as an agent by looking at its behavioral characteristics or by the methodology using which it was produced? Is a printer daemon an agent? If a piece of software is not an agent, is there a way to make it an agent? Many attempts have been made to define the notion of agent or agency, ranging from quite generic definitions to restrictive definitions.

This book adopts a generic definition of an *agent:* a piece of code that does a small and well-defined job by offering services. Some of its characteristics are declarative specification, autonomous behavior, and interactiveness. This book primarily concentrates on abstracting the data sources and related software to make them agents, hence this less restrictive definition. Although this book might not put an end to the debates mentioned earlier, it tries to answer the most practical question of how to convert a normal program into an agent.

Until now, most of the books in this field have discussed the issues in an informal manner or only theoretically without talking about concrete implementations, leaving the reader wondering, "It is all fine, but how do I implement an agent?!" This book fills the gap to some extent. It extensively talks about the implementation of agents, using the IMPACT agent development environment. Also, most of the books in this field are a collection of articles written by different authors, often lacking coherence and consistency. Although written by seven authors, the treatment in this book is quite coherent and consistent.

The book starts off with the following questions: What is an agent? If a piece of software is not an agent, how do you make it an agent? As a response, the authors give 10 desiderata for agents and agent platforms. These desiderata roughly cover issues such as the accessing of heterogeneous data sources, a declarative framework for specifying actions, types of reasoning, security, efficiency, reliability, and validation. Of these issues, validation of an infrastructure by deploying a set of applications seems out of place. The main contributions of this book are an approach to making a normal program an agent and providing a practically implementable formal theory of agent construction and agent interactions.

Heterogeneous Agent Systems, V. S. Subrahmanian, Piero Bonatti, Jürgen Dix, Thomas Eiter, Sarit Kraus, Fatmaözcan, and Robert Ross, Cambridge, Massachusetts, MIT Press, 580 pp., $60.00, ISBN 0-262-19436-8.

To discuss and illustrate the concepts discussed throughout the book, three motivating examples are considered. This approach is one of the good features of this book because using these examples throughout gives the book a sense of continuity. The motivating examples are a department store application, a flight terrain application, and a supply-chain management application. The department store application is supposed to proactively provide information and make multimedia presentations of the products depending on the profile of the customer. The flight terrain application is primarily used to maintain the path of a flight, taking into consideration the current position and three-dimensional terrain information provided by satellites and terrain databases, respectively. The supply-chain management application takes care of the inventory levels and the ordering logistics for production companies.

All the applications, especially the last two, have characteristics that make them suitable for using agents. First, they consist of multiple, largely independent, and well-defined tasks to be done. Second, there are independent data sources, and these sources are independently updated. Third, the actions executed can vary depending on the circumstances. Fourth, the entities in the applications need to reason with uncertain data and the beliefs they hold about other entities in the domain.

This book can roughly be divided into three parts: (1) basic concepts, (2) implementation, and (3) advanced concepts.

The first part discusses basic concepts such as the IMPACT architecture, service description language, the converting of legacy data and software into agents, and the development of agents. In chapter 2, the IMPACT architecture is discussed. One distinguishing feature of IMPACT is that it supports fuzzy matchmaking using concept hierarchies. The service description language is discussed in chapter 3. Chapter 4 contains the core concepts of the book—converting, by use of code calls, legacy data and software application program interfaces (APIs) into services. After converting the APIs and defining the constraints, the agent code looks similar to a PROLOG program. Chapter 6 discusses the components of agent programs—action base and action constraints—which is followed by a discussion of the syntax and semantics of the agent programs. Part 1 is not very formal and is highly readable.

The second part discusses implementation issues. The IMPACT server implementation and protocol details are given in chapter 5. Chapter 12 discusses the implementations for issues such as the compiling of the agent programs and safety checks.

The third part (chapters 7 to 11), consuming roughly half the book, is devoted to advanced concepts such as belief, temporal and uncertainty reasoning, and security. This part is very formal; it is meant for researchers interested in the formal treatment of the theory. Others are advised not to venture into reading these chapters, at least during the first read, lest they might think that programming agents is an esoteric discipline that is beyond the reach of the everyday programmer!

Chapter 13 discusses a logistics application developed by the authors to illustrate the concepts discussed in the book. This chapter is quite weak; it does not cover most of the concepts discussed in the book, such as belief reasoning and temporal reasoning. Considering the supposedly significant features of agents introduced early on in the book, a more complex application that demonstrates the usefulness of the features should have been chosen.

One noticeable exclusion is agent communication mechanisms. There is no mention of communicative acts; performatives; or FIPA (Foundation for Intelligent Physical Agents), the upcoming agents standard.

There are significant advantages to applying this paradigm to make normal programs into agents, especially legacy code. The first and foremost is the declarativeness that is achieved. It allows easier program comprehension; modification; and correctness checking, manual as well as automatic. It also allows you to plug high-level reasoning capabilities, such as constraint, belief, and temporal and uncertainty reasoning. into the program (agent). However, it is obvious that not all programs require such complex and sophisticated mechanisms: For example, the department store example does not need it, but applying this paradigm to the flight terrain application might be quite worthwhile. If the flight terrain application was programmed in a traditional way, it would not have the advantages of declarative programming mentioned earlier. Also, the advantages of an agent architecture such as a matchmaking service will be missed, although it is debatable whether fuzzy matching is appropriate for all applications. Thus, if a program is complex enough, then using this paradigm will be more advantageous than the traditional way of programming.

Each chapter starts with an overview and ends with a section on related work containing brief discussions of other approaches, along with a comparison with their approach. For all the algorithms discussed in this book, the complexity analysis, as well as experimental results, are given. There is an appendix giving the codes for the motivating examples. There is an impressive list of references at the end of the book. One thing lacking in this book is exercises. Including them for at least some chapters would have made this book more useful for academic purposes. The editing should have been tighter: A few grammatical errors have slipped through. Most of the pages look a bit cluttered because of mathematics symbols. Indentation for the items in the enumerated and bulleted lists and a bold font for theorem, definition, and example headings would have improved the readability.

This book might not be suitable as a primary textbook for a course on agents because of its specificity and lack of exercises. However, it will be useful as a supplementary textbook or a reference book. It will be of use to the agent programmers (practitioners) who want to get a concrete idea about the implementation issues. This book also might interest those who want to acquire an understanding of the practical implementations of the advanced concepts such as reasoning with beliefs, uncertainty, and security with respect to agents. The bulk of the book, I feel, is for the researchers with a bent for formal treatment.

Acknowledgment

I would like to thank my colleague, M. Sasikumar, for his valuable comments on this review.

P. Ravi Prakash is a senior staff scientist at the National Centre for Software Technology, India. He received his B.Tech in computer science and engineering from JNTU, Kakinada, India. His research interests include intelligent agents, multiagent systems, planning and scheduling, soft computing, and parallel and distributed computing. His e-mail address is ravi@ncst.ernet.in.