450 pp.
ISBN 0-262-52234-9
$45.00

Reviewed by: El-Sayed M. El-Alfy
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030
eabdelka@stevens-tech.edu

Software has dramatically changed from being a passive tool to being an intellectually active assistant. In the last few years, significant research has been directed toward active software entities that can act autonomously and adapt to changes in dynamic environments. Furthermore, these entities can communicate and collaborate with each other and with humans to simplify many complex tasks. Real-world applications have been developed with user interfaces that can actively assist and improve user interaction with the software. Applications of active software entities in industry and academia include distributed project management, communication systems, network routing, manufacturing, planning and scheduling, and information and knowledge management.

Motivated by the increasing complexity of future large-scale software systems of a dynamic and distributed nature, developers are creating nonhuman intelligent objects that are self-organizing and that can react to various situations in their environments. These objects are called "agents." The terms "agent" and "agency" describe any object with different levels of smartness, ranging from simple programs to more complicated knowledge-based systems. Other common terms found in the literature are "software agents," "intelligent agents," "intelligent software agents," "softbots," and "knowbots." Many of these agents apply the principles of artificial intelligence to facilitate the process of developing powerful distributed software systems. Several research projects have been devoted to developing various aspects of software agents, such as learning, mobility, security, human-agent interaction, agent-agent collaboration, and limited resource sharing among agents.

Software Agents, edited by Jeffrey Bradshaw, is an excellent introduction to software agent technology. It addresses an increasingly important area of artificial intelligence and software development. The book is a solid, comprehensive collection of research papers on various aspects of software agents. These papers were written by a number of active researchers and software developers from both academia and industry. The book covers software agent-based systems in technical detail. The careful compilation of papers, covering the state of the art and explaining in a single resource a wide variety of aspects of software agents, makes it valuable for all people who want to know and understand the various concepts, theoretical and applied issues, standards, architectural design, operations, technologies, capabilities, and future trends of software agents. It is also useful for people looking for areas of further research and development.

Following an introductory chapter by Bradshaw, the material is divided into three parts. The topics introduced in the first part, "Agents and The User Experience," give an extensive introduction to software agents. Questions such as "What are software agents?," "Why do we need them?," and "What are their capabilities and limitations? are addressed. The first part also introduces topics about human-agent interaction, intelligent interface design, knowledge acquisition, delegation, information sharing, and coordination.

The first chapter in this part, "How might people interact with agents?" by Donald A. Norman, covers various aspects of human-agent interactions and how to incorporate these aspects into the design of agent-based systems. "Agents: From direct manipulation to delegation" by Nicholas Negroponte discusses positive contributions of agents to the design of intelligent user-computer interfaces.

Continuing on the same theme of intelligent user interfaces, "Interface agents: Metaphors with character," by Brenda Laurel, presents key characteristics of agent-based interfaces, such as agency, responsiveness, competence, and accessibility. In "Designing agents as if people mattered," Thomas Erickson identifies a number of important problems in designing agents and discusses how to adapt their functionality so that they will behave as people expect. Ben Shneiderman, in "Direct manipulation vs. agents: Paths to predictable, controllable, and comprehensible interfaces," presents a different vision of computers as intelligent systems and discusses some design concerns about the often misleading term "intelligence."

The second part of the book, "Agents for Learning and Intelligent Assistance," deals with various issues in incorporating agent technology into the education system: agent programming, cooperative learning, and agent architecture. Software modularity is covered at several levels of abstraction, for example, procedural and object-oriented programming paradigms as well as agent-oriented programming and software with mental states.

"Agents for information sharing and coordination: A history and some reflections," by Thomas W. Malone et al., views some of the authors' earlier work on Information Lens and OVAL (an acronym for the key component of the system: Objects, Views, Agents, and Links). It also describes some applications to demonstrate the power and generality of the OVAL system.

Pattie Maes, in "Agents that reduce work and information overload," presents an approach that helps the interface agents to become "personal assistants" by allowing them to learn from the user or from other agents through direct and indirect feedback. It also describes four examples of existing agent-based systems that were built using the learning approach. The theme of "KidSim: Programming Agents without a programming language," by David C. Smith et al., is how to program without a programming language. The authors create a powerful toolkit by combining two ideas: graphical rewrite rules and programming by demonstration.

"Lifelike computer characters: The Persona project at Microsoft Research," by Gene Ball et al., reviews the requirements for an assistant-like interface. It also discusses a lifelike conversational interface using natural spoken language and gives an overview of the Persona system.

Guy A. Boy, in "Software agents for cooperative learning," presents a design approach for agent-based systems for cooperative learning by combining three key concepts: active documents, software agents, and organization. It also gives some examples of software agents for cooperative learning such as problem-solving, case-based learning, and incidental leaning. The last chapter in this section, "M: An architecture of integrated agents," by Doug Riecken, describes several aspects of a general architecture that integrates different reasoning processes (agents) including spatial, temporal, functional, structural, and causal.

Topics related to agent-oriented programming are covered in the last part of the book, "Agent Communication, Collaboration, and Mobility." Various approaches to agent communication, agent interoperability, information gathering, open agent architectures, and mobile agents are presented.

"An overview of agent-oriented programming" by Yoav Shoham presents his vision of agents and agent-oriented programming (AOP) according to their mental states, such as beliefs, commitments, and capabilities. The chapter also discusses the key components of the AOP framework and discusses related work. Tim Finin et al., in "KQML as an agent communication language," answer the question of how agents exchange information and share knowledge. They first present the approach of knowledge sharing effort (KSE), point out its problems, and suggest solutions to the problems identified. Also covered are agent communication languages and their desired features and the Knowledge Query and Manipulation Language (KQML).

"An agent-based framework for interoperability" by Michael R. Genesereth describes various approaches for software interoperation and how agent-based systems can be used to attack many relevant problems.

The design of an agent-based system to integrate the large number of distributed and heterogeneous information sources is presented in "Agents for information gathering" by Craig A. Knoblock and José-Luis Ambite. Issues such as the organization of agents and the domain model of their knowledge, communication languages and protocols, query processing, learning, and related work are covered.

In "KAoS: Toward an industrial-strength open agent architecture," Bradshaw et al. describe the KAoS agent architecture and how it addresses the same limitations of current agent technology including scalability, infrastructure, security, and semantics. They also introduce some applications of KAoS and discuss its strengths.

Philip R. Cohen and Hector J. Levesque, in "Communicative actions for artificial agents," critique KQML as an agent communication language based on a communicative actions called "performatives." They also suggest a theory of speech acts and provide a comparison of this theory to other agent communication languages such as AOP and Telescript.

"Mobile agents" by James E. White explores the concept of a new communication paradigm—mobile agents—and presents a solid overview of Telescript technology as well as related work. Telescript technology is an object-oriented programming language that allows interoperability for network services.

Although writing a book in such a rapidly evolving technology is a difficult task, this compilation of papers does an outstanding job in providing a concise foundation of the development of software agent-based systems. It is well organized and clearly illustrated, which makes it easy to read and understand. It also gives a detailed overview of up-to-date research results and covers a wide range of aspects of software agent-based systems: human-software interaction, mobility, communication, learning, and adaptability. The book is an invaluable source of information, well-structured and clearly presented, with rich and comprehensive coverage of both historical and contemporary research topics. I recommend it for those in the field of software agents, mobile computing, and distributed systems. It could be used as a text for an artificial intelligence or software development course, supplemented by other reading assignments on more recent developments in the field. It is suitable for undergraduate computer science and engineering majors. It is also a good resource for software developers and artificial intelligence scientists and would be useful to those in other disciplines who want to gain knowledge of various aspects of software agents and their capabilities.