Since the early 1990s, software agents—also known as intelligent agents, knowbots, softbots, or bots for short—have been the subject of a great deal of speculation and marketing hype. This sort of hype has been fueled by “computer science fiction”—personified images of agents reminiscent of the robot HAL in Stanley Kubrick’s movie 2001: A Space Odyssey or the “Roboboy” Dave in the movie Artificial Intelligence. As various chapters in the text demonstrate, software agents have come to play an increasingly important role in EC—providing assistance with Web searches, helping consumers comparison shop, and automatically notifying users of recent events (e.g., new job openings). This appendix is provided for those readers who want to learn a little more about the general features and operation of software agents in a networked world such as the Web.

WHY SOFTWARE AGENTS FOR EC, ESPECIALLY NOW?

For years pundits heralded the coming of the networked society or global village. They imagined an interconnected Web of networks linking virtually every computer and database on the planet, a Web that science fiction writer William Gibson dubbed the matrix. Few of these pundits envisioned the problems that such an interconnected network would bring. One exception was Alvin Toffler, who warned in his book Future Shock (1970) of an impending flood, not of water, but of information. He predicted that people would be so inundated with data that they would become nearly paralyzed and unable to choose between options. Whether that has occurred is an open question. There is no doubt, however, that today’s world of networked computers—intranets, Internet, and extranets—has opened the floodgates.

INFORMATION OVERLOAD

Consider some simple facts (Harris 2002):

- In 2001, it was estimated that over 10 billion (nonspam) e-mail messages were sent per day. The figure is expected to grow to 35 billion messages per day by 2005.
- The amount of unique information being produced worldwide is doubling every year. In 2001, the world created an estimated 6 exabytes ($10^{17}$ bytes) of new information. In 2002, the figure was 12 exabytes.

Taken together, that is more information than was accessible in the entire 300,000 years of human history.

Regardless of the metric used (e.g., growth in the number of networks, hosts, users, or traffic), the Web is still growing rapidly. In 2003, the public Internet contained about 1 trillion pages and was increasing at a rate of approximately 8 million pages per day. Unfortunately, end users are often overwhelmed. They spend most of their time navigating and sorting through the available data, spending little time interpreting, and even less time actually doing something about what they find. The end result is that much of the data we gather goes unused. For example, according to the Gartner Group (Kyte 2002):

- The amount of data collected by large enterprises doubles every year.
- Knowledge workers can analyze only about 5 percent of the data.
- Most of knowledge workers’ efforts are spent trying to discover important patterns in the data (60 percent or more); a much smaller percentage is spent determining what those patterns mean (20 percent or more); and very little time (10 percent or less) is spent actually doing something based on the patterns.
- Information overload reduces knowledge workers’ decision-making capabilities by 50 percent.

What is the solution to the problem of data overload? Paul Saffo, director of the Institute of the Future, asks, how do we reduce “the flood of data to a meaningful trickle?” (Saffo 1989).

DELEGATE, DO NOT NAVIGATE

As far back as 1984, Alan Kay, one of the inventors of Windows-based computing, recognized the problems associated with point-and-click navigation of very large data repositories and the potential utility of “agent-information overload” (Kay 1984). More recently, Nicholas Negroponte, director of MIT’s Media Lab, echoed the same theme in his acclaimed book, Being Digital (1995). Negroponte said:

“Future human computer interfaces will be rooted in delegation, not the vernacular of direct manipulation—pull down, pop-up, click—and mouse interfaces. ‘Ease of use’ has been such a compelling goal that we sometimes forget that many people don’t want to use the machine at all. They want to get something...”
Appendix D: Software (Intelligent) Agents

There is a pressing need to automate tasks performed by administrative personnel and knowledge workers, especially in the fields of decision support and repetitive office activity. In a fast-paced society, time-strapped people need new ways to minimize the time spent on routine personal tasks, such as shopping for groceries or travel planning, so that they can devote more time to professional activities (see Perez 2002).

VALUE OF SOFTWARE AGENTS IN A NETWORKED WORLD

A major value of employing software agents with intranet, Internet, and extranet applications is that agents are able to assist in locating and filtering data. They save time by making decisions about what is relevant to the user. They are able to sort through the network and the various databases effortlessly and with unwavering attention to detail in order to extract the best data. They are not limited to hard (quantitative) data; they can also obtain soft data about new trends that may cause unanticipated changes (and opportunities) in local or even global markets. With an agent at work, the competent user’s decision-making ability is enhanced with information rather than paralyzed by too much input. Agents are artificial intelligence’s answer to a need created by Internet-networked computers (see Wooldridge 2000).

Information access and navigation are the major applications of software agents in today’s intranet, Internet, and extranet worlds, but there are also other reasons why this technology is expected to grow rapidly:

- **Mundane personal activity.** In a fast-paced society, time-strapped people need new ways to minimize the time spent on routine personal tasks, such as shopping for groceries or travel planning, so that they can devote more time to professional activities (see Perez 2002).
- **Search and retrieval.** It is not possible to directly manipulate a distributed database system containing millions of data objects. Users will have to relegate the task of searching and cost comparison to agents. These agents will perform the tedious, time-consuming, and repetitive tasks of searching databases, retrieving and filtering information, and delivering it back to the user.
- **Repetitive office activity.** There is a pressing need to automate tasks performed by administrative and clerical personnel in functions such as sales or customer support in order to reduce labor costs and increase office productivity. Today, labor costs are estimated to be as much as 60 percent of the total cost of information delivery (Abushar and Hirata 2002).
- **Decision support.** Increased support for tasks performed by knowledge workers, especially in the decision-making area, is needed. Timely and knowledgeable decisions made by these professionals greatly increase their effectiveness and the success of their businesses in the marketplace.
- **Domain experts.** It is advisable to model costly expertise and make it widely available. Expert software agents could model real-world agents such as translators, lawyers, diplomats, union negotiators, stockbrokers, and even clergy.

To date, the list of tasks to which commercially available agents and research prototypes have been applied includes the following: advising, alerting, broadcasting, browsing, critiquing, distributing, enlisting, empowering, explaining, filtering, guiding, identifying, matching, monitoring, negotiating, organizing, presenting, querying, reminding, reporting, retrieving, scheduling, searching, securing, soliciting, sorting, storing, suggesting, summarizing, teaching, translating, and watching.

Overall, software agents make the networked world less forbidding, save time by reducing the effort required to locate and retrieve data, and improve productivity by off-loading a variety of mundane, tedious, and mindless tasks.

A BRIEF HISTORY OF INTELLIGENT AGENTS

The concept of software agency is surprisingly old. Over 50 years ago, Vannevar Bush (Bush 1945) envisioned a machine called the Memex that enabled users to navigate through oceans of data and information. In the 1950s, John McCarthy conceived the Advice Taker (McCarthy 1958), a software robot living and working in a computer network of information utilities (much like today’s Internet). When given a task by a human user, the software robot could take the necessary steps or ask advice from the user when it got stuck. The futuristic prototypes of intelligent personal agents, such as Apple Computer’s Phil or Microsoft’s Bob, perform complicated tasks for their users following the same functions laid out by McCarthy in his Advice Taker.

Although modern approaches to software agency can trace their roots to these earlier visions, current research started in the mid-1980s and has been influenced by work done in a number of fields, including artificial intelligence (e.g., reasoning theory and artificial life), software engineering (e.g., object-oriented programming and distributed processing), and human–computer interaction (e.g., user modeling and cognitive engineering). One of the most promising new directions for software agents is the development of semantic Web. Semantic Web could be the technology needed for making software agents much more effective in their decision-making ability.
“smarter” (e.g., see Hendler 2001) in discovering information on the Internet and in databases. Details on the semantic Web are provided at the end of this appendix.

**DEFINITIONS AND CONCEPTS**

Outside the realm of computers, the term agent is well defined. It derives from the concept of agency, which is to employ someone (such as a theatrical agent) to act on your behalf. An agent represents a person or organization and interacts with others to accomplish a predefined task.

In the computer realm, things are not so simple. There are almost as many definitions for the term **software (intelligent) agent** as there are people employing it. Here are some examples:

- "An agent is anything that can be viewed as perceiving its environment through sensors and acting on that environment through effectors." (Russell and Norvig 1995, p. 33)
- "Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed." (Maes 1995, p. 108)
- "An intelligent agent is software that assists people and acts on their behalf. Intelligent agents work by allowing people to delegate work that they could have done to the agent software. Agents can, just as assistants can, automate repetitive tasks, remember things you forgot, intelligently summarize complex data, learn from you, and even make recommendations to you." (Gilbert 1997, p. l)
- "[An agent is] a piece of software that performs a given task using information gleaned from its environment to act in a suitable manner so as to complete the task successfully. The software should be able to adapt itself based on changes occurring in its environment, so that a change in circumstances will still yield the intended result." (Hermans 1996, p. 14).

Software agents can be distinguished from regular computer software programs along several dimensions, as shown in Exhibit D.1.

In addition to the differing definitions, individual researchers have also invented a variety of synonyms for the term agent in order to promote their particular brand of software agency. Included among the alternatives are intelligent agent, software robot, knowbot (knowledge-based robot), softbot (intelligent software robot), taskbot (tasked-based robot), autonomous agent, personal assistant, and digital software (intelligent) agents

**EXHIBIT D.1 Software Agents Versus Traditional Software Programs**

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Regular Software</th>
<th>Software Agents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nature</td>
<td>Static</td>
<td>Dynamic.</td>
</tr>
<tr>
<td>Manipulation</td>
<td>Direct: User initiates every action</td>
<td>Indirect: Autonomous.</td>
</tr>
<tr>
<td>Interactivity</td>
<td>Noninteractive</td>
<td>Dialogues are fully interactive. Actions may be initiated by either the user or the agent system. Interacts with user and with other agents.</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Never changes, unless changed by a human or an error in the program</td>
<td>Adapts, learns.</td>
</tr>
<tr>
<td>Temporal continuity</td>
<td>Runs one time, then stops to be run again when called</td>
<td>Persistent: Continues to run over time.</td>
</tr>
<tr>
<td>Response</td>
<td>Predictable: Does what you tell it to, even if you didn’t mean what you said</td>
<td>Interprets what you mean, not what you say. In the best of circumstances, actions are based on rules, but they may change over time or in reaction to different circumstances.</td>
</tr>
<tr>
<td>Autonomy, independence</td>
<td>Follows instructions</td>
<td>May initiate actions, as well as respond to instructions. May be mobile, traveling to other servers. Dispatches simultaneously to accomplish various parts of a task in parallel.</td>
</tr>
<tr>
<td>Mobility</td>
<td>Stays in one place</td>
<td></td>
</tr>
<tr>
<td>Concurrency</td>
<td>Generates process in one dedicated server with limited processing power</td>
<td>Can travel and interact with local entities, such as data bases, file servers and stationary agent, through message passing.</td>
</tr>
<tr>
<td>Local interaction</td>
<td>NTBL:Accesses data across network using client-server architecture</td>
<td></td>
</tr>
</tbody>
</table>

*Source: Based on Feldman and Yu (1999).*
proxy. The different terms can be confusing, but they do serve a purpose. Not only do they capture our attention—the term knowbot is certainly more engaging than agent—but they also denote the character of the various agents, the roles they play (e.g., performing tasks), and the features they possess (e.g., intelligence). Throughout this appendix and this book we use the terms intelligent agent and software agent interchangeably, which is how agents are presented in the not-so-technical literature.

INTELLIGENCE LEVELS
Definitions of agents are greatly dependent on the agents’ levels of intelligence, which are described by Lee et al. (2002) as follows:

- **Level 0 (the lowest).** These agents retrieve documents for a user under straight orders. Popular Web browsers fall into this category. The user must specify the URLs where the documents are. These agents help in navigating the Web.
- **Level 1.** These agents provide a user-initiated searching facility for finding relevant Web pages. Internet search agents such as Google, Alta Vista, and Lycos are examples. Information about pages, titles, and word frequency is stored and indexed. When the user provides key words, the search engine matches them against the indexed information. These agents are referred to as search engines.
- **Level 2.** These agents maintain users’ profiles. They then monitor the Internet and notify the users whenever relevant information is found. An example of such an agent is WebWatcher (search for WebWatcher at cs.cmu.edu). Agents at this level are frequently referred to as semi-intelligent or software agents.
- **Level 3.** Agents at this level have a learning and deductive component of user profiles to help a user who cannot formalize a query or specify a target for a search. DiffAgent (CMU) and Letizia (MIT) are examples of such agents. Agents at this level are referred to as learning or truly intelligent agents.

Similar to the concept of levels is the idea of “agent generation.” For a description of these generations today and in the future, see Murch and Johnson (1999).

CHARACTERISTICS OF SOFTWARE AGENTS: THE ESSENTIALS
Although there is no commonly accepted definition for the term software agent, people think of several possible traits when they discuss software agents. Four of these traits—autonomy, temporal continuity, reactivity, and goal driven—are essential to distinguish agents from other types of software objects, programs, or systems. Software agents possessing only these traits are often labeled simple or weak. Virtually all commercially available software agents are of this sort.

Besides these essential traits, a software agent may also possess additional traits such as adaptability, mobility, sociability, and personality. Typically, these latter traits are found in more advanced research prototypes. In this section, we will consider the essential traits. The other traits will be covered in later sections.

AUTONOMY
As Maes (1995) points out, regular computer programs respond only to direct manipulation. In contrast, a software agent senses its environment and acts autonomously upon it. A software agent can initiate communication, monitor events, and perform tasks without the direct intervention of humans or others.

Autonomy implies that an agent takes initiative and exercises control over its own actions (Huhns and Buell 2002), displaying the following characteristics:

- **Goal-orientation.** Accepts high-level requests indicating what a human wants and is responsible for deciding how and where to satisfy the requests. These are referred to by Hess et al. (2000) as homeostatic goal(s).
- **Collaboration.** Does not blindly obey commands but can modify requests, ask clarification questions, or even refuse to satisfy certain requests.
- **Flexibility.** Actions are not scripted; the agent is able to dynamically choose which actions to invoke, and in what sequence, in response to the state of its external environment.
- **Self-starting.** Unlike standard programs directly invoked by a user, an agent can sense changes in its environment and decide when to act.

TEMPORAL CONTINUITY
A software agent is a program to which a user assigns a goal or task. The idea is that once a task or goal has been delegated, it is up to the agent to work tirelessly in pursuit of that goal. Unlike regular computer programs that terminate when processing is complete, an agent continues to run—either actively in the foreground or sleeping in the background—monitoring system events that trigger its actions. You can think of this attribute as “set and forget.”
REACTIVITY
A software agent responds in a timely fashion to changes in its environment. This characteristic is crucial for delegation and automation. The general principle on which software agents operate is “When X happens, do Y,” where X is some system or network event that the agent continually monitors (Gilbert 1997).

GOAL DRIVEN
A software agent does more than simply respond to changes in its environment. An agent can accept high-level requests specifying the goals of a human user (or another agent) and decide how and where to satisfy the requests. In some cases, an agent can modify the goals or establish goals of their own.

OTHER COMMON TRAITS
Some software agents also possess other common traits, as discussed below.

Communication (Interactivity)
Many agents are designed to interact with other agents, humans, or software programs. This is a critical ability in view of the narrow repertoire of any given agent. Instead of making a single agent conduct several tasks, additional agents can be created to handle undelegated tasks. Thus, communication is necessary in these instances. Agents communicate by following certain communication languages and standards such as Agent Communication Language (ACL) and Knowledge Query and Manipulation Language (KQML) (Bradshaw 1997; Jennings et al. 1998).

Intelligence and Learning
Currently, the majority of agents are not truly intelligent because they cannot learn; only some agents can learn. This learning goes beyond mere rule-based reasoning because the agent is expected to use learning to behave autonomously. Although many in the artificial intelligence (AI) community argue that few people want agents who learn by “spying” on their users, the ability to learn often begins with the ability to observe users and predict their behavior. One of the most common examples of learning agents is the wizards found in many commercial software programs (e.g., in Microsoft Office applications). These wizards offer hints to the user based on patterns the program detects in the user’s activities. Some of the newer Internet search engines boast intelligent agents that can learn from previous requests the user has made.

For a comprehensive discussion of these and additional characteristics, see Hess et al. (2000).

MOBILE AGENTS
Agents can be classified into two major categories: resident and mobile. Resident agents stay in the computer or system and perform their tasks there. For instance, many of the wizards in software programs are designed to carry out very specific tasks while a person is using their computer. Mobile agents, on the other hand, move to other systems, performing tasks there. A mobile agent can transport itself across different system architectures and platforms. EC agents are mobile.

Mobility refers to the degree to which the agents themselves travel through the network. Some agents are very mobile; others are not. Mobile scripts can be composed on one machine and shipped to another for execution in a suitably secure environment; in this case, the program travels before execution, and therefore no static data need be attached. Finally, agents can be mobile with state, moving from machine to machine in the middle of execution and carrying accumulated state data with them. Such agents can be viewed as mobile objects that travel to agencies (other agents or computing systems representing other entities) where they can present their credentials and obtain access to services and data managed by the agencies. Agencies can also serve as brokers or matchmakers, bringing together agents with similar interests and compatible goals and providing a meeting point at which they can safely interact.

Mobile agents can move from one Internet site to another and send data to and retrieve data from the user, who can focus on other tasks in the meantime. This can be very helpful to a user. For example, if a user wanted to continuously monitor an electronic auction that takes a few days, the user essentially would have to be online continuously for days. Software applications that automatically watch auctions and stocks are readily available. For example, a mobile agent that watches stocks travels from site to site, looking for information on a certain stock as instructed by the user. If the stock price hits a certain level or if there is news about the stock, the agent alerts the user. What is unique about a mobile agent is that it is a software application that moves on its own to different computers to execute (Murch and Johnson 1999; Yan et al. 2001).

Resident (nonmobile) agents can be defined by two dimensions (see Exhibit D.2a, on next page) whereas mobile agents are defined in a three-dimensional space (Exhibit D.2b). For example, in Exhibit D.2, note that expert systems, which are not mobile, may fall below the threshold line, and thus are regular (nonmobile) software agents. True intelligent agents are listed above the threshold line.
USE OF MOBILE AGENTS IN E-COMMERCE

Mobile agents are useful in the context of e-commerce for a number of reasons (Yeo 2002):

1. **They reduce the network load.** The distributed system model used in the business world today often generates considerable network traffic, which is due to the interaction of applications with the server. Mobile agents allow the interaction to take place locally by dispatching themselves to the destination host. This means that the computations are moved to the data, rather than the data to the computer.

2. **They overcome network latency.** Critical real-time systems in e-business, such as online stock trading, require immediate response to events, with no delay. Mobile agents offer a solution to this need, because they can be dispatched from a central controller to act locally and to directly execute the controller’s command.

3. **They execute asynchronously and autonomously.** Mobile devices often require an established connection with the server to perform their function. However, maintaining the connection for mobile devices could be very expensive or not technically feasible. Mobile agents can work independent of the mobile device asynchronously and autonomously so that the mobile device can reconnect to the server at a later time to transmit the data.

4. **They are naturally heterogeneous.** Computer networks are heterogeneous. Different business entities have different standards on their networks. Mobile agents are generally computer- and transport-layer independent and depend only on the execution environment (e.g., Java Run Time Environment). Therefore, they can work within a wide range of computer systems and with different hardware configurations.

APPLICATIONS OF MOBILE AGENTS

According to Yeo (2002), mobile agents are well suited to the following applications:

- **E-commerce.** A commercial transaction may require real-time access to remote resources, such as stock quotes, and perhaps even agent-to-agent negotiations.

- **M-commerce.** Intelligent agents can facilitate m-commerce, operating in a wireless environment (see Matskin and Tveit 2001).

- **Personal assistance.** Mobile agents’ ability to execute on remote hosts makes them suitable assistants for performing tasks in the network on behalf of their creators. Mobile agents acting remotely
operate independently of their limited network connectivity; their creators can even turn off their computers. For example, to bid in an auction, a user can send a mobile agent to interact with the bidding agent to monitor the price change and to perform auto bidding according to the instructions of the creator.

- **Secure brokering.** An interesting application of mobile agents is in collaborations in which not all the collaborators are trusted. In such a scenario, parties could let their mobile agents meet on a mutually agreed secure host. Collaboration could then take place at the host without risk of the host taking the side of one of the visiting agents.

- **Workflow applications and groupware.** The nature of workflow applications includes support for the flow of information among coworkers. Mobile agents are especially useful here, because, in addition to mobility, they provide a degree of autonomy to the workflow item. Individual workflow items fully embody the information and behavior they need to move through the organization independent of any particular application.

- **Searching and filtering.** Collecting information from a network often amounts to searching through vast amounts of data for a few relevant pieces of information. On behalf of a user, a mobile agent could visit many sites, search through the information available at each site, filter out the irrelevant information, and build an index of links to pieces of information that match a search criterion.

- **Monitoring and notification.** Sometimes information is not spread out across space but across time (e.g., financial data). New information is constantly being produced and published on the network. Agents can be sent out to wait for certain kinds of information to become available.

- **Interface agents,** to databases etc., is a common category (Conway and Koehler 2000). They facilitate access to databases and certain applications and resources. They make access to these resources easy—almost automatic.

### MOBILE AGENT TOOLS

A number of agent technologies have been introduced by several companies and research institutes in recent years. Examples of commercial agent technologies are IBM’s Aglets, ObjectSpaces’s Voyager, Mitsubishi’s Concordia, British Telecommunications’ ZEUS, IntelliOne Technologies’ Agentbuilder, and Living Systems AG’s Living Markets. In addition, Microsoft’s Agent Version ASDK V1.1 Beta, developed by IBM Japan, is fully compliant with Java Development Kit (JDK) 1.1. ASDK V1.2.0 promises to work with Java 2.

Another tool is D’Agents (previously known as Agent Tcl), a mobile-agent system under development at Dartmouth College. D’Agents uses a scripting language as its main language but provides a framework for incorporating additional language features. The latest release is D’Agents 2.0. The public release of D’Agents 2.0 supports only Tcl agents, but the internal release supports Tcl, Java, Python, and Scheme agents. The Java and Scheme modules are available to the public by request.

### AGENT SECURITY

As indicated earlier, although the mobile agent (MA) paradigm has many advantages over the traditional programming paradigm, it also raises several security issues. Automatically executing arbitrary codes on any host can be dangerous. Moreover, in order to fulfill their tasks, MAs must be able to access and configure security-sensitive resources (Reiser and Vogt 2000). Hence, it is possible that agents could leak or destroy sensitive data and disrupt the normal functioning of the host system.

Conversely, because agents may carry sensitive data, they need to be protected against tampering by the hosts they visit (Karnik and Tripathi 2000). For example, say a customer’s agent wants to acquire system information and bring it to another system. If the information is not encrypted, then the second system can see the information of the first one. In addition, mobility introduces new threats whereby a hostile host may refuse to execute agents’ codes or to transfer agents to successive execution sites (Corradi et al. 1999).

In general, security issues in an MA system can be analyzed from four different perspectives (Marques et al. 1999): protecting hosts from (1) access by unauthorized parties, (2) attacks by malicious agents and protecting agents, (3) from attacks by other agents, or (4) attacks by malicious hosts.

### SIMPLE SOFTWARE AGENTS: HOW DO THEY WORK?

Let’s look more deeply at simple software agents to understand how they work. Exhibit D.3 (next page) depicts the operation of a “simple” software agent possessing the essential traits described earlier. The operation of a simple agent is best understood in the context of an example: Virtually all PC-based or Internet-based e-mail packages provide end users with the ability to create agents that scan incoming and outgoing e-mail messages and carry out some predefined action based on the content of the message. Let us see how these agents operate.
AUTOMATING A SINGLE SET OF TASKS WITHIN A SINGLE APPLICATION

Simple agents work within the context of a single application and focus on a single set of tasks with a circumscribed set of outcomes. Much of the work done by these agents automates simple repetitive tasks that could be performed by a person, if that person had the time, the inclination, or was available to do so. This is certainly the case with e-mail agents.

E-mail agents operate within the confines of an e-mail package. Their sole purpose is to scan incoming and outgoing messages, looking for various keywords that have been designated by the end user and performing one or more of a handful of possible operations, such as deleting the message, forwarding the message, or storing the message within a given folder. For example, an agent at hotmail.com decides which mail to place in the “junk mail” file. It will also block any mail the user asks to be blocked.

Clearly, these are all tasks that could be performed by the end users. Yet, agents have their advantages. They never sleep (unless the application or the system is shut down). They send you an automatic reply: “Professor Turban is away until July 10.” They are always available, even when the end user is away from their desk. They are never bored, and they never miss work. Some executives, managers, and knowledge workers receive more than 100 to 200 messages a day, and reviewing these messages can be a tedious, time-consuming, and error-prone task. Automating the review with an e-mail agent can offload some of the review process and “adminis-trivia.”

The goals of a simple software agent are explicitly specified by an end user. This is done by either creating a set of “if/then/else” rules or a script that predefines the actions to be taken by the agent when certain conditions arise. The actions are invoked by the agent without end-user intervention.

In the case of an e-mail agent, the goals usually come in the form of if/then rules. Users do not actually input if/then statements. Instead, the end user fills in a form or dialog box, and the rules are generated from the choices. The dialog box shown in Exhibit D.4 comes from the “Out-of-Office” agent provided with Microsoft’s Exchange and Outlook e-mail packages. In this case, the end user specifies both a set of keywords for one or more fields in an e-mail message—in this instance, the end user has entered “John Smith” in the “From” field and an action to be “Forwarded” to “Sara Jones.” When the end user clicks “OK,” the underlying if/then rule is automatically generated. For Exhibit D.4, the rule would read something like, “When the ‘out-of-the-office’ switch is on, if a message arrives that has the exact words ‘John Smith’ in the ‘From’ field, then ‘Forward’ the message to ‘Sarah Jones.’”

Once a set of goals has been established, the agent fades into the background, out of sight from the end user, waiting for some system event to occur. The event might be a mouse click or keystroke, the passage of time (measured by the system’s internal clock), the arrival of a message flowing across a communication port, a modification to a database, or the deletion or saving of a file in a particular directory. When an event of interest occurs, the agent performs its designated task(s) according to the goals that have been specified. With e-mail agents, the events of interest are the arrival or sending of e-mail messages at or from the end user’s desktop.

When a triggering event occurs, the agent begins its logical processing. The processing algorithms used by an agent differ from one application to the next. However, the processing usually involves pattern matching of a single item (e.g., a document) against each of the if/then rules that have been established by the end user. If an item satisfies a particular rule, the action specified in the “then” part of the rule is exe-
In the case of an e-mail agent, the agent simply compares an incoming or outgoing message against all of the end user’s if/then rules. If a message satisfies the conditions specified in the “if” part of the rule, the actions designated in the “then” part are carried out by the agent.

As of 2002, popular e-mail programs such as Outlook Express, Eudora, and Netscape Mail included dozens of agents for e-mail creation and management, such as autorespond, new mail notification, and spell checking (see Metz et al. 2002). Typical e-mail agents include surfcontrol.com, netnanny.com, Mailfil from agentland.com, and Spam Eater from wantdbest.com.

LEARNING AGENTS

Software agents are called intelligent agents or learning agents if they have the capacity to adapt or modify their behavior—that is, to learn. Simple software agents, such as e-mail agents, lack this capacity. If a simple software agent has any intelligence at all, it is found in the subroutines or methods that the agent uses to do pattern matching. However, these subroutines or methods are built into the program and cannot be modified by the agent. At present, few, if any, commercially available agents have the ability to learn. However, some research prototypes have this capability.

A number of these prototypes have been developed by Pattie Maes and her colleagues at the MIT Media Lab. They have created a series of “user interface” agents employing machine learning techniques and inspired by the metaphor of a “personal assistant.” In Maes’ (1995) words:

Initially, a personal assistant is not very familiar with the habits and preferences of his or her employer and may not even be very helpful. The assistant needs some time to become familiar with the particular work methods of the employer and organization at hand. However, with every experience the assistant learns, either by watching how the employer performs tasks, by receiving instructions from the employer, or by learning from other more experienced assistants within the organization. Gradually, more tasks that were initially performed directly by the employer can be taken care of by the assistant. (p. 109)
As Maes suggests, there are four ways for an interface agent to modify its behavior:

1. **“Look over the shoulder” of the user.** An agent can continually monitor the user’s interactions with the computer. By keeping track of the user’s actions over an extended period of time, the agent can discern regularities or recurrent patterns and offer to automate these patterns.

2. **Direct and indirect user feedback.** The user can provide the agent with negative feedback either in a direct or indirect fashion. Directly, the user can tell the agent not to repeat a particular action. Indirectly, the user can neglect the advice offered by an agent and take a different course of action.

3. **Learn from examples given by the user.** The user can train the agent by providing it with hypothetical examples of events and actions that indicate how the agent should behave in similar situations.

4. **Ask the agents of other users.** If an agent encounters a situation for which it has no recommended plan of action, it can ask other agents what actions they would recommend for that situation.

Examples of commercial personal assistants, according to Agentland.com, are BonziBuddy ([bonzi.com](http://bonzi.com)), Desktop Wizard ([e-clips.com.au](http://e-clips.com.au)), and Noa ([madoogali.com](http://madoogali.com)).

### LEARNING AGENTS: AN EXAMPLE

The major difference between the operation of an intelligent learning agent and the workings of a simple software agent is in how the if/then rules are created. With a learning agent, the onus of creating and managing rules rests on the shoulders of the agent, not the end user. To understand this difference, let us examine the operation of an intelligent e-mail agent.

#### An Intelligent e-Mail Agent

Maxim is an intelligent e-mail agent that operates on top of the Eudora e-mail system ([Maes 1994](#)). This agent relies on a form of learning known as case-based reasoning. Maxim continually monitors what the user does and stores this information as examples. The situations are described in terms of fields and keywords in the message (i.e., the “From,” “To,” and “Cc” lists, the keywords in the “Subject” field, and so on), and the actions are those performed by the user with respect to the message (e.g., the order in which the user reads it, whether the user deleted or stored it, and so on). When a new situation occurs, the agent analyzes its features based on its stored cases and suggests an action to the user (such as read, delete, forward, or archive).

The agent measures the confidence, or fit, of a suggested action to a situation. Two levels of confidence are used to determine what the agent actually does with its suggestion. If the confidence is above the “do-it” threshold, the agent automatically executes the suggestion. If the confidence is above the “tell-me” threshold, the agent will offer a suggestion and wait for input from the user. Exhibit D.5 is an intelligent e-mail agent from Maxim, software written at MIT, displays a series of messages, along with those suggestions that exceed the tell-me threshold.

### VALUE OF LEARNING AGENTS

Learning agents also address the problem of end-user competence. With simple software agents, the end user is required to recognize when an agent should be used, create the agent, specify the rules to be used by the agent, and modify or edit the rules to account for changing interests and work patterns. Learning agents remove all these impediments (although they are not a total panacea).

Learning agents such as Maxim operate on the assumptions that the application involves a substantial amount of repetition and that the repetition does not vary considerably from one end user to another. Without these assumptions, there is no way to build the requisite levels of confidence, nor is there any need to learn the underlying rules if a general set of rules can be applied to all users. Additionally, critics argue that most people do not want intelligent agents “looking over their shoulders” ([Greif 1994](#)). They contend that simple software agents that require end users to fill out forms are easy to use and provide enough utility for the average end user.

### MULTIAGENTS AND COMMUNITIES OF AGENTS

Agents can communicate, cooperate, and negotiate with other agents. The basic idea is that it is easy to build an agent that has a small amount of specialized knowledge. However, in executing complex tasks that require much knowledge, it is necessary to employ several software agents in one application. These agents need to share their knowledge, or the results of applying this knowledge together may fail.

An example is routing among telecommunications networks. Information can pass through a network controlled by one company into another network controlled by another company. Computers that control a telecommunications network might find it beneficial to enter into agreements with other computers that control other networks about routing packets more efficiently from source to destination.
Another example is wireless devices, which are continuously increasing their functionality. Wireless devices offer more than Internet access and e-commerce support; they also enable device-to-device communication. You can take a photograph with your digital camera in one location and transmit pictures wirelessly, in seconds, to your office. Intelligent agents embedded in such devices facilitate this interaction.

Each of these situations is an instance in which computers control certain resources and might be able to help themselves by strategically sharing this resource with other computers. With wireless devices, the resource might be a person's time, whereas with a telecommunications network, the resource might be communication lines, switching nodes, or short- and long-term storage. In each situation, the computers that control these resources can do their own job better by reaching agreements with other computers.

**MULTIAGENT SYSTEMS**

With multiagent systems, no single designer stands behind all of the agents. Each agent in the system may be working toward different goals, even contradictory ones. Agents either compete or cooperate (Decker et al. 1999). In a multiagent system, for example, a customer may want to place a long-distance call. Once this information is known, agents representing the carriers submit bids simultaneously. The bids are collected, and the best bid wins. In a complex system, the customer's agent may take the process one step further by showing all bidders the offers, allowing them to rebid or negotiate.

A complex task is broken into subtasks, each of which is assigned to an agent that works on its task independently of others and is supported by a knowledge base. Acquiring and interpreting information is done by knowledge-processing agents that use deductive and inductive methods, as well as computations. The data are refined, interpreted, and sent to the coordinator, who transfers to the user interface whatever is relevant to a specific user's inquiry or need. If no existing knowledge is available to answer an inquiry, knowledge creating and collecting agents of various types are triggered.

Of the many topics related to multiagent systems, we will present the following ones: negotiation in e-commerce, coordination, collaboration, communities of agents, and agent networking.

**MULTIAGENT NEGOTIATION IN E-COMMERCE**

A considerable amount of research and development is being done on multiagent negotiation systems in e-commerce (Beer et al. 1999). Consider a situation where agents cooperate to arrange for a person's summer vacation in Hawaii: The person's agent notifies sellers' agents about the potential traveler's needs for a
hotel, plane tickets, and a rental car, and the sellers’ agents submit bids. The person’s agent collects the bids and tries to get lower rebids. The sellers’ agents can use rules for the negotiations.

The process of negotiation and its relationship to bidding processes are being studied by researchers. Two issues that are being examined include the following: Can automated agents learn strategies that enable them to effectively participate in typical, semistructured, multi-issue business negotiations? What is required and how does it work? For an example of automated online bargaining, see Lin and Chang (2001). Other negotiation-related issues have been investigated by Yan et al. (2001) and Beer et al. (1999). Related to negotiation is intermediation, which is an advanced feature for e-commerce. Valera et al. (2001) describe a multiagent system that incorporates intermediation.

COORDINATION

Coordination is a key factor in the success of multiagent systems. The purpose of the coordination mechanism is to manage problem solving so that cooperating agents work together as a coherent team. Coordination is achieved by exchanging data, providing partial solution plans, and enforcing constraints among agents. Coordination can be done by analyzing related activities; for example, coordination is part of order processing (see Mondal and Jain 2001), synchronization, structured group mediation, and information sharing. See Jamali et al. (1999) for more on coordination.

COLLABORATION

Lotus Notes/Domino is a comprehensive collaborative software product. It includes Notes Agents, which operate in the background to automatically perform routine tasks for the Notes user such as filing documents, sending e-mail, looking for particular topics, or archiving older documents. These agents can be created by designers as part of an application for automating routine tasks, such as progress tracking or serving as reminders of overdue items, or for performing more powerful functions, such as manipulating field values and bringing data in from other applications. Agents can be private, created by the user and used only by the user, or they can be shared, created by a designer and used by anyone who has access to the application or database.

Because an agent typically represents an individual user’s interests, collaboration is a natural area for agent-to-agent interaction and communication. IBM, for example, is exploring multiagent interaction through several research efforts.

Another example of collaborating agents is provided by Bose (1996), who proposed a framework for automating the execution of collaborative organizational processes performed by multiple organization members. The agents emulate the work and behavior of human agents. Each agent is capable of acting autonomously, cooperatively, and collectively to achieve the collective goal. The system increases organizational productivity by carrying out several tedious watchdog activities, thereby freeing humans to work on challenging and creative tasks. For example, Bose (1996) describes an example of a travel authorization process that can be divided into subtasks delegated to agents.

Another example involves scheduling a meeting. Several agents can cooperate in proposing meeting times and places until a mutually acceptable schedule is found. Note that a simpler case is that of a single agent that checks the calendars of the participants to determine when all of them are free and then books a free meeting room and notifies the participants.

A multiagent system for assigning air cargo to airline flights in creative ways is presented by Zhu et al. (2000), and the collaboration issue has also been researched by Nardi et al. (1998). Yet another example was proposed by Wang et al. (2002) for a “society” of agents to monitor financial transactions for irregularities. The concept of agent societies (see the following) is also discussed by Huhns (2002a). Finally, Wu and Sun (2002) describe collaboration in multiagent bidding for EC.

A successful commercial system was developed by IBM for improved planning and scheduling of operations for certain papermaking plants (Bassak 2003). The details are provided in Jamali (1999).

COMMUNITIES OF AGENTS

Elofson et al. (1997) introduce the concept of communities of agents behaving in believable ways in the entertainment industry. Communities of agents are groups of agents collaborating on a complex task. They may work as one entity, or they may represent several entities (e.g., buyers, sellers, and support services). Agents, by definition, represent someone or something. There is great scope for more sophisticated agents of this form to be used in movies and games, possibly even generating a new genre of interactive movies. This concept was expanded to a society of agents by Huhns (2002b). A society of agents is a broader grouping than a community of agents; a society may include several (or many) communities of agents.

AGENT NETWORKING

A system for implementing an “ecology of distributed agents” was created at MIT Media Lab. The system supports applications of distributed agents in various environments, including wireless ones (Minar et al. 2000).
READY FOR THE FUTURE

Only a few years ago, discussions about software agents were always qualified with the phrase, “In the future, software agents will…” The future, in terms of software agents, is finally here. The Web has proven to be a fertile ground for practical applications of software agents. Auction watchers, comparison shoppers, personal Web spiders, newshounds, site recommenders, and portfolio assistants are some of the agents operating in today’s world of EC. It does not stop there, however. If anything, the pace at which existing and experimental agent technologies are being applied to the virtual world has quickened.

A good place to monitor new applications of software agents and to keep an eye on both their near- and long-term future is the BotSpot Web site (botspot.com). The BotSpot offers a compendium of existing commercial applications and products, as well as pointers to ongoing research sites (academic and commercial). One of the key research sites to watch is MIT’s Software Agent Group (agents.media.mit.edu). A number of MIT’s research projects have made their way to the Web.

Commercial applications of software agents raise a number of issues about personal privacy. Take, for instance, a software agent that recommends new CDs to a customer. To do this, the agent builds a personal profile on the customer, collecting demographic information and information about the types of music they like and the CDs that they have purchased in the past. Based on this information, the agent compares the customer’s profile with the profiles of others. It then generates recommendations by finding the profiles of people like that customer and seeing what they have purchased but that this customer has not. In other words, “birds of a feather” should like the same things. Commercial agents of all sorts are not only privy to consumers’ personal profiles, but also to their personal actions. As users move about the Web under the guidance of an agent, the agent has the potential of knowing where a user has been and where the user is likely to go. The question becomes, “Whose information is it?”

Fortunately, commercial suppliers of agent technology are at least aware of the privacy issues. For instance, Firefly, which was one of the early companies trying to bring “recommendation” agents to the Web (now part of Microsoft’s Passport system), has proposed a series of standards such as P3P (the Platform for Privacy Preferences), which makes individual control and informed consent the key operating principles of software agents. The standards are continuously being considered by W3C (w3c.org) and by the Internet community at large. Only time will tell how effective these standards will be.

WIRELESS AGENTS

With the increased number of wireless e-commerce applications (m-commerce), it has been a challenge to make these services more personalized and to handle constraints such as bandwidth and screen size. For an overview of software agents developed for m-commerce, see Matskin and Tveit (2001). A special category of wireless agents are wearable ones (see Starner 2002).

SEMANTIC WEB

Intelligent agents need to be able to communicate in a flexible way. A key issue faced by those developing intelligent agents for the Internet is that most information on the Web is not designed for other software programs to use. Traditionally, knowledge over the Web has been represented in a format called HTML (Hypertext Markup Language). It was designed for humans and is based on the concept that information consists of pages of text and graphics that contain links. HTML is powerful in controlling the appearance of a Web page.

Unfortunately, HTML does not present the content in a way that another software system can easily understand and use. Software agents do not care about the appearance of a Web page, but only the contents. Furthermore, some data may be hidden in the database, which makes it very difficult for a software agent to acquire the data. For example, if a travel agent needs to find a flight from New York to Tokyo on Wednesday afternoon, it would not be able to do so on an HTML-based Web page, because the Web page does not contain information that allows the agent to locate the flight schedule (unless the agent has the permission to use a query language to access the database). It is also difficult for a software agent to find a product item from a Web page containing many products.

Semantic Web is a solution to this problem. It is a tool that provides a content presentation and organization standard so that content can be shared safely among different software applications. With mutually understandable semantic constructs, knowledge on the Web is easier for software agents to access, understand, and share. When Alex receives an e-mail invitation and needs to make a travel arrangement to a conference, software agents can integrate his personal preferences with data from different airline, hotel, and rental car reservation sites to make the best arrangements (Hendler 2001).
DEFINITION

Semantic Web is part of the effort to incorporate meanings and relationships among concepts into Web information. As described by Berners-Lee et al. (2001), semantic Web is an extension of the current Web in which information is given well-defined meanings, better enabling computers and people to work in cooperation. Semantic Web is meant to enable an environment in which independent, Internet-connected information systems can exchange knowledge and action specifications, resulting in the execution of an activity acceptable to all systems involved (Swartz 2002). In other words, semantic Web is an approach for presenting information on the Web in a machine-sharable format. Semantic Web intends to integrate meanings and relationships of concepts into Web representation.

In practice, semantic Web is a collaborative effort led by World Wide Web Consortium (W3C), with participation from a large number of researchers and industrial partners. W3C defines semantic Web as “the representation of data on the World Wide Web. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML (extensible markup language) for syntax and URIs for naming” (W3C.org/2001/sw/2003). The concepts of RDF, XML, and URI are described at the W3C site.

One other tool used in Web semantics is an ontology. Ontology is defined in the Oxford English Dictionary as “the science or study of being.” In artificial intelligence, we usually attribute the concept of ontology to the specifications of a conceptualization; that is, defined terms and the relationships among them. Simply put, ontology is a set of terms related to a knowledge domain, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topics. For example, the ontology of cooking and cookbooks may include ingredients, procedures for processing the ingredients, and the differences among various cooking styles. For semantic Web, an ontology is a collection of related statements, which together specify a variety of relationships among data elements and ways of making logical inferences among them. Ontologies are a type of hierarchical thesaurus in which each subheading inherits all the characteristics of the headings above it.

ADVANTAGES AND LIMITATIONS OF SEMANTIC WEB

Using semantic Web to design intelligent agents has the following advantages:

- **Easy to understand.** Semantic Web demonstrates objects and their relationships as graphic templates for easy understanding.
- **Easy resource integration.** It is easier to integrate systems and modules designed in semantic Web. This also makes it easier for system analysis and maintenance.
- **Saving development time and costs.** Semantic Web allows incremental ontology creation, enabling more rapid system development and lower development costs.
- **Automatic update of content.** Because agents can easily locate a specific knowledge on semantic Web, they can have functions to update or import contents automatically. This adds the level of intelligence to the software agents.
- **Easy resource reuse.** The ontology-based annotations can turn briefings into reusable resources.

The limitations of semantic Web include the following:

- **The graphical representation may be oversimplified.** For example, using an arrow to represent a relation between two instances is unable to show more complicated multiparty relations.
- **Additional tools for searching content and building references to preexisting instances are needed** for effective use of semantic Webs.
- **Ontologies may not be correctly defined.** In some cases, the outcome of this could be severe. It is still hard to prove the completeness or correctness of a defined ontology.
- **When agents deal with a semantic Web containing information that is inconsistent, incorrect, or unreliable, the agents could become contaminated or be misled.**
- **Because the semantic Web allows agents from different systems to communicate and share information, security is a key concern. Security is always a problem for an open system.**

SEMANTIC WEB SERVICES

Semantic Web services are a combination of semantic Web and Web services. One of the first applications of semantic Web services is MusicBrainz (musicbrainz.org), which provides a large database of music metadata for sharing (Swartz 2002). By 2002, it already had more than 300,000 tracks. The idea of MusicBrainz can be traced back to the Internet Compact Disc Database (CDDDB) project, which was started in 1996. After CDDDB was acquired by a content delivery company and no longer open for free use, several projects were created to replace it. One of them, the CDIndex, later became MusicBrainz.

The major function of MusicBrainz is to provide semantic information over the Web for other systems to use. When a consumer purchases a new CD and inserts the CD into their computer, the audio
player will probably come up with a generic name (such as Audio CD23), complete with track 1, track 2, and so on. If the consumer were using MusicBrainz, the audio player would have attempted to connect to the MusicBrainz server to see if that CD's metadata were available. If the metadata were available, the user's CD player would have renamed the CD and the tracks. If the metadata were not available and the user had filled in the names for their own use, then MusicBrainz would ask the user whether they wanted to share the information. If the user did want to share, the information would be sent to MusicBrainz for sharing with other users. MusicBrainz provides the metadata in RDF.

Because MusicBrainz's data format is open and in the RDF format, it can be repurposed for numerous applications. For example, file-sharing systems (such as Napster, Freenet, or Audio Galaxy) can use the metadata to provide more information about the MP3s that are available for download or to make it easier to search for a song. Artists can provide links so that appreciative fans can donate money if they like the music.

**STANDARDS FOR SOFTWARE AGENTS**

**COMMUNICATION LANGUAGES**

The development of an effective, rich agent communication language (ACL) is one of the keys to the success of intelligent agents. KQML is the first ACL that has seen substantial use. It is the de facto standard for agent communication languages. However, there are still no fixed specifications, no interoperable implementations, and no agreed-upon semantics. The FIPA (Foundation for Intelligent Physical Agents, fipa.org) is currently addressing all of these problems with its FIPA ACL, which is a well-specified standard based on formal semantics. A newer language is Formal Language for Business Communication (FLBC) (see Moore 2000).

In addition to standardizing the agent communication language, FIPA also seeks to standardize other aspects of agent technology. These include architectural guidelines and specifications for constricting agents and agent platforms, defining open-standard interfaces for accessing agent management services, the human–agent interaction part of an agent system, security management and facilities for securing interagent communication, software agent mobility, and technologies enabling agents to manage explicit, declaratively represented ontologies. Standardization efforts are also underway within Object Management Group (OMG, omg.org) and the Mobile Agent System Interoperability Facility (MASIF).

**OPEN PROFILING STANDARD**

The Open Profiling Standard (OPS) provides Internet site developers with a uniform architecture for using personal profile information to offer individuals tailored content, goods, and services that match their personal preferences while protecting their privacy.

The idea works like this: Individuals have a Personal Profile that contains their personal information. The profile is constructed by computer; people may be asked to answer questions, or their movement on the Net is followed to construct the profile. The user's computer stores this profile (and can, at the user's option, be securely stored in a corporate-wide or global directory). The first time that an individual visits a Web site that supports OPS, the Web site will request information from the Personal Profile. The individual has the choice of releasing all, some, or none of the requested information to the Web site. In addition, if the Web site collects additional information about the individual's preferences, it can (with the individual's permission) store that information in the Personal Profile for future use. On subsequent visits, the individual can authorize the Web site to retrieve the same personal information without asking permission each time.

**AGENTX**

The Internet Engineering Task Force's AgentX (objs.com/survey/ietf.htm) is a standardized framework for extensible Simple Network Management Protocol (SNMP) agents, the network management protocol for TCP/IP. It defines processing entities called master agents and subagents; a protocol, AgentX, used to communicate between them; and the procedure by which the extensible agent processes SNMP protocol messages. In SNMP, agents can monitor the activity in the various devices on the network and report to the network console workstation.

**MORE RESOURCES ON E-COMMERCE AGENTS**

In addition to the e-commerce software agents cited in various chapters in the text, the following are some additional suggestions for further research or study:

- Kim and Lee (2003) describe EC agents and relate them to various EC models.
Maes et al. (1999) and Papazoglou (2001) provide comprehensive overviews of agents in EC, whereas Mandry et al. (2000–2001) examine opportunities and risks in electronic markets. Menczer et al. (2002) discuss intelligent query agents for EC. Mondal and Jain (2001) discuss their multiagent system for sales-order processing, in which each agent carries out a different task. Several studies deal with issues related to the implementation of intelligent agents in EC; see, for example, Ram (2001) and Wagner and Turban (2002).

Sugumaran (2002) developed a framework that relates agents to knowledge management for e-commerce applications. A similar framework is presented by Zahir (2002).

RESOURCES ON SOFTWARE AND INTELLIGENT AGENTS

A large number of Web sites contain useful information about software agents. The following are some representative examples:

- One of the best places to start is the University of Maryland’s Web site on intelligent agents (agents.umbc.edu). Start with Agents 101 at agents.umbc.edu/introduction. The site has downloadable papers and reports and an extensive bibliography with abstracts (see “Publications and Presentation”).
- BotSpot (botspot.com) has comprehensive information about e-commerce agents and other agents (see also internet.com). Cutting-edge developments are provided by Don Barker (pcai.com).
- MIT Media Lab (search for media projects at media.mit.edu) provides a list of agent projects and much more.
- Intelligent Information Interfaces (i3net.org) provides information on agent-related research in Europe.
- Carnegie Mellon University has several agent-related programs (search for software agents at cs.cmu.edu/research).
- IBM operates an Intelligent Agents Center (research.ibm.com and research.ibm.com/infoecon).
- Stanford University has several research teams developing agent technology (search for Knowledge Systems Laboratory at stanford.edu).
- Agentland.com is another “must” place to visit. It contains an up-to-date list of dozens of agents classified into e-commerce and entertainment. Some of the agents and development tools can be downloaded.
- The Computer Information Center (compinfo-center.com/) facilitates collaboration and technology transfer about agent development.
- The University of Michigan has several agent development projects (eecs.umich.edu). An extensive list of resources is also available at ai.eecs.umich.edu.
- PC AI magazine (pcai.com) contains a vast amount of resources on intelligent agents organized by topic.
- The National Research Council of Canada (nrc.ca) provides an artificial intelligence subject index for agents.
- The Xerox Palo Alto Research Center (parc.xerox.com) provides information on software agents in general and on multiagent systems in particular.

In addition to references, articles, and application cases, you can find a list of leading vendors, some with customers’ success stories. Related intelligent systems are covered as well. For example, Microsoft Agent is a software technology that enables an enriched form of user interaction that can make using and learning to use a computer easier and more natural. With the Microsoft Agent set of software services, developers can easily enhance the user interface of their applications and Web pages with interactive personalities in the form of animated characters. These characters can move freely within the computer display, speak aloud (and display text on screen), and even listen for spoken voice commands (see msagentting.org). You can download Microsoft Agent at microsoft.com/products/msagent/downloads.htm.

KEY TERMS

Learning agents 9  Ontology 14  Semantic Web services 14
Mobile agents 5  Open Profiling Standard (OPS) 15  Simple agents 8
Mobility 5  Resident agents 5  Software (intelligent) agents 3
Multiagent systems 11  Semantic Web 14
REFERENCES


Bush, V. “As We May Think.” Atlantic Monthly, July 1945.


Perez, E. “Intelligent Agents: It’s Nice to Get Stuff Done for You.” *Online* 26, no. 3 (May/June 2002).


---

**ADDITIONAL READINGS**


