

MISRC Working Paper

Final Version Appears in:

Kauffman, R. J., March, S. T., Wood, C. A. "Agent Sophistication: Design Aspects for Data-Collecting Agents." *International Journal of Intelligent Systems in Accounting, Finance, and Management (IJISAFM)*. Forthcoming

DESIGN PRINCIPLES FOR LONG-LIVED INTERNET AGENTS

Robert J. Kauffman

Carlson School of Management
University of Minnesota

Salvatore T. March

Owen Graduate School of Management
Vanderbilt University

Charles A. Wood (contact author)

Carlson School of Management
University of Minnesota
Minneapolis, MN 55455
Phone: 612-624-8030; Email: cwood@csom.umn.edu

Forthcoming in the *International Journal of Intelligent Systems in Accounting, Finance and Management*, Fall 2000.

ABSTRACT

Prior research on intelligent Internet agents has failed to address the needs of *long-lived data-collecting agents*, focusing instead on *short-lived transaction agents*. Transaction agents typically run for a few seconds and retrieve information for a single transaction. With the staggering growth of electronic commerce, researchers and practitioners will want to design long-lived data-collecting agents that intelligently search for, retrieve, interpret, categorize, and store vast amounts of related information each time that they run. Such agents can run over the course of days rather than seconds and can be used by practitioners for decision support applications or by researchers as part of an empirical research methodology. This paper proposes a framework for agent sophistication, and emphasizes a number of design concepts for long-lived Internet agents, including *intelligence, validation, concurrency, recovery, monitoring, and interactivity*. These concepts are used in the development of an illustrative tool called *Electronic Data Retrieval Lexical Agent (eDRILL)*, an object-oriented data-collecting agent. eDRILL is designed using the *Unified Modeling Language (UML)* and is written in Java. It gathers research data from an online auction.

Keywords: Agent design, agents, data collection, intelligent agents, long-lived agents, object-oriented design, UML

ACKNOWLEDGMENTS

An earlier version of this paper appeared in the *Proceedings of the 1999 Workshop on Information Technology and Systems (WITS '99)*, held in Charlotte, North Carolina in December 1999. The authors

thank Sudha Ram, Mark Nissen, Chris Dellarocas, Jungpil Hahn, and Gove Allen, as well as the WITS reviewers, for their helpful comments and criticisms on an earlier version of this work. We also thank three anonymous referees for their suggestions on a later version of this work, when it was under review at the *International Journal of Intelligent Systems in Accounting, Finance and Management*.

INTRODUCTION

Software agents are computer programs that mimic human actions. These agents are increasingly being used to help potential customers automatically search, aggregate, and evaluate the reams of information available on different electronic commerce (EC) Web sites (Jennings and Wooldridge, 1998; Wooldridge and Jennings 1995; Nwana, 1996; Shoham, 1993) rather than retrieving all necessary data themselves. For example, MySimon.com, Amazon.Com's Shop the Web, Excite Inc.'s Jango.Com, and OneLook.Com's Bottom Dollar use agent technology to compare prices for specific computers, toys, CDs, shirts, perfumes, book titles or other items available from multiple Web sites. Just as buyers use short-lived *transaction agents* to retrieve small amounts of data, researchers can use long-lived *data-collecting agents* to retrieve large amounts of empirical data with far less effort and far more fidelity than traditional research techniques such as surveys, field studies, or experiments. Similarly, practitioners can use long-lived data-collecting agents to gather data needed for decision support and system applications. Researchers and practitioners alike can utilize agent technology to develop long-lived data-collecting agents that intelligently search for, retrieve, interpret, categorize, store, and process the huge volume of electronic commerce information that is available on the Web.

Developing such agents can be daunting, however. *First*, there is no standard way of defining data on a Web site (Machlis, 1999). The Extensible Markup Language (XML) holds promise for standardizing data definitions on Web pages. To date, however, no such standards have been adopted for common use. *Second*, Hypertext Markup Language (HTML), which dominates commercial Web sites, has no such conventions (Weil, 2000). Hence, each agent must be manually created for a specific Web site to collect data such as buyer identification, item descriptions, and prices from HTML pages as defined on that site. *Third*, transaction agents typically run for a few seconds and collect a single piece of information from a small number of well-known Web sites. By contrast, data-collecting agents can run for several hours -- or even days -- and possibly collect different types of data from a larger number of Web sites. *Fourth*, data-collecting agents may need to engage in sophisticated search, interpretation, categorization, monitoring, and analysis activities to determine which sites to visit and when to visit them or to navigate through different Web sites in order to acquire the needed data.

Research on *agent design* is relatively new and concentrates on transaction agents (Wooldridge and Jennings, 1995; Maes, Guttman, and Moukas, 1999). Design considerations for long-lived data-collecting agents have been essentially ignored. The need to maintain connections to other computers over an extended length of time while performing complex data collection and analysis activities presents design challenges that are well beyond those addressed by current software development environments, even those oriented toward agent development. A design framework for such Internet agents must capture

and represent capabilities such as *intelligence, interaction, concurrency, validation, monitoring, and recovery*. These are not easily represented in current agent design methodologies, making it difficult for researchers to communicate their specific requirements for long-lived agents to software developers.

This paper presents a new framework for developing agents that incorporates considerations for long-lived data-collecting agents. We first describe the unique aspects of such agents, focussing on the various roles they may play. We argue that identifying and explicitly representing these roles significantly simplifies the agent design process. To build this framework, we draw heavily upon literature in three areas, Internet transaction agents, robotics, and long-lived database transactions. We use components of the *Unified Modeling Language (UML)* (Booch, Rumbaugh, and Jacobson, 1997; Fowler and Scott, 2000) in a framework-based methodology to capture and represent these roles and the additional complexity inherent in data-collecting agents. The methodology is illustrated in the development of **eDRILL** (*Electronic Data Retrieval Lexical Agent*), a data-collecting agent used to study the behavior of rare coin buyers at an online auction.

BACKGROUND LITERATURE

In this section we first review the relevant existing literature related to developing Internet agents. As discussed above, this literature deals primarily with short-lived and relatively simple transaction agents. Although we found no literature that specifically addresses long-lived agent development, we apply principles found in the long-lived database transactions literature to agent design. We also evaluate the capabilities required of long-lived agents and argue that additional framework components are needed for their representation.

Transaction Agents on the Internet

Several articles address the effects of Internet agents on e-commerce (Jennings and Wooldridge, 1998; Maes, 1994). Maes, Guttman, and Moukas, make predictions about the future of such agents (1999) and discuss the use of several different kinds of agents to aid consumers in collecting information about purchases. Data collected by such agents is usually of a small amount and a single type, and is typically used by a single person to make a single decision. Other authors predict a future where monitoring agents will continuously search the Web for deals on behalf of consumers (Mougayar, 1998; Leebaert, 1998).

In this research, we define transaction agents as short-lived agents that run for several seconds and aid in conducting a single transaction on the Internet. These include, for example, agents that retrieve the best price for an item from several different vendors or agents that make purchase decisions for a buyer. In contrast, data-collecting agents are long-lived agents that run for hours or days to collect large

amounts of data for research or decision-making purposes. These can include, for example, market research aimed at identifying buying patterns for a specific product or scientific research aimed at testing specific theories and hypotheses.

Mui and McCarthy (1987) developed an interesting example of this kind of capability: a *Financial Security Analyzer (FSA)* that uses artificial intelligence techniques to scan for documents on the Security Exchange Commission's (SEC) *Electronic Data Gathering and Retrieval (EDGAR)* system. FSA acts as an agent of the financial decision maker quickly scanning for corporate 10K reports that meet certain criteria, giving the financial decision maker more time to analyze SEC data. FSA is one of the first data-collecting agents that came into meaningful practical use; however, it has limited capabilities. It only functions in a single EDI environment and lacks the flexibility that must be present in tools that can collect data and perform analysis in the multiple Web-page environments that are prevalent today. In addition, although Mui and McCarthy describe the FSA agent as a proof of concept, they do not delve into design principles or into the coding techniques used to develop it.

Etzioni and Weld (1994), the authors of the Jango agent, describe important aspects of a "softbot" development. They assert that an agent should be *goal oriented*, *charitable* (intelligent), *balanced* (aware of costs), and *integrated* with a user interface. Jennings and Wooldridge (1998, pp. 1-5) continue this discussion and describe three characteristics of intelligent agents when explaining the design of short-lived agents. *First*, they should be *responsive*, or able to respond to environmental changes in a timely fashion. *Second*, they should be *proactive*, or able to exhibit opportunistic, goal-directed behavior and initiate appropriate actions. *Third*, they should be *social*, or able to interact with humans or other agents. We extend these requirements for all agents to include a number of other considerations critical to developing long-lived agents used for intense data collection.

Agents in Robotics

Agent development initially grew out of robotics research. Software agents ran inside autonomous robots and allowed them to perform simple tasks, such as welding or other assembly line tasks (Groover et al., 1986). Robotic agents contain some of the same processes that are found in Internet agents. *First*, depending on the application, robotic agents are required to have some specified level of responsiveness or proactivity, and some social interaction as described by Jennings and Wooldridge (1998). *Second*, robots often have some level of intelligence embedded in their software that typically uses rules to navigate through a variable environment (Lewis, et al., 1996). *Third*, robots often incorporate some form of restart and recovery procedure (Lewis, et al., 1996). *Fourth*, robots often contain complex programming constructs, such as concurrency achieved through asynchronous multithreading (Pardo-Castellote, et al., 1995).

However, robotic agents differ from Internet agents in several important ways. *First*, in robotic agents recovery from an abnormal termination can often be accomplished by a simple restart from a pre-specified initial state (Lewis, et al., 1996; Pardo-Castellote, et al., 1995). Recovery for an Internet agent typically requires *context sensitivity* and varies according to the history of work previously performed. Long-lived agents require a level of persistence so that they do not simply restart, but so that they also restore the state of the agent before the abnormal termination occurred and possibly requiring a review of the previously collected data. *Second*, robotic agents normally do not interact with a user, but rather concentrate on a clearly specified and invariant task, such as welding on a machine line (Lewis, et al., 1996; Pardo-Castellote, et al., 1995). By contrast, most data collecting agents require significant user interaction and must vary their behavior based on it. *Third*, robotic agents are *task-centric*. The design of robotic agents deals primarily with the task to be performed (Pardo-Castellote, et al., 1995; Coste-Maniere, 1996). However, Internet agents are *data-centric*. They must acquire, interpret and transfer data, possibly modifying their behavior based on the data they acquire. They do not simply perform a repetitive task.

The robotics literature on agent design concentrates on individual programming tasks and how these tasks interact with the robotic hardware. What is noticeably missing from the robotics design literature is a framework that can be used to design their software, specifically, an agent design framework.

Long-Lived Database Transactions

Database designers are concerned about recovery from concurrent transactions that are *long-lived* (Bernstein, et al., 1987). Concurrent, long-lived transactions occur because multiple database users can access the database at the same time, each using separate, long-lived transactions. This frequently occurs in databases that support large-scale engineering and design applications. Simple transaction logging and versioning mechanisms are ineffective since the unit of work involved in a single transaction may be extensive and the way in which a transaction is processed may depend on intermediate results from other transactions. Transaction management in such an environment is extremely complex and difficult.

Keen and Dally (1993, 1997) developed *extended ephemeral logging (XEL)* to address this type of problem. XEL logs database transactions based on a series of fixed-length, queued, short-lived logs, with one log to be used for each transaction. These logs can be deleted once the transaction is complete, thus allowing maximum space and recovery efficiency for concurrency control processes.

With long-running agents, similar concurrency problems are encountered. A single agent fires off multiple threads that need to be controlled individually. When backing up or restoring, agents should be able to recover data that has been previously collected and to continue collecting data at the same point as

before the recovery. In this research we incorporate techniques similar to XEL recovery and concurrency concepts. These techniques enable data collecting agents to recover from failures such as connection line drops or server crashes.

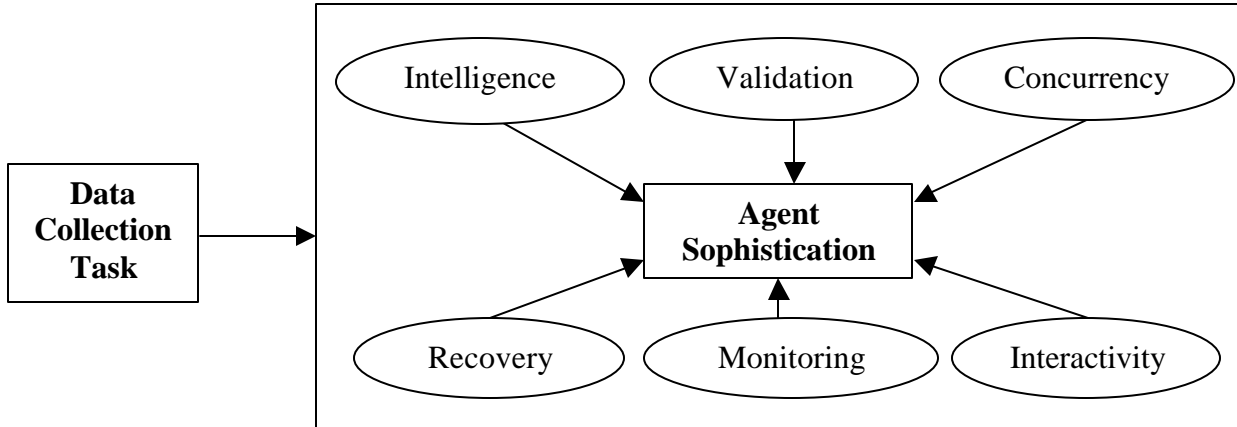
FRAMEWORK DEVELOPMENT

We introduce the concept of *agent sophistication* to reflect the capabilities required of Internet agents. All computer programs require features such as error-free (or near error-free) performance, a certain degree of efficiency, and suitability to a goal-oriented task. However, there are specific aspects of agent sophistication that must be considered when developing Internet agents in addition to standard design features common to most computer programs.

Preliminaries

We identify six sophistication features that needed to be considered: *intelligence, concurrency, validation, recovery, monitoring, and interactivity*. The specific requirements of a data-collecting agent for each feature are dependent upon the scope of the data collection task. Figure 1 illustrates this interaction and presents a framework that forms the basis of our *Agent Development Framework*.

Figure 1. Agent Development Framework



We do not claim that our *Agent Development Framework* design methodology is *ontologically complete* (Wand and Weber, 1992; Wand and Wang, 1996), rather our methodology specifically addresses shortcomings of existing agent methodologies based on *agent development experience*. An ontological analysis of agent design may, in fact, be fruitful, but is beyond the scope of this paper. In addition, agents, as software, are subject to the myriad of design elements found in traditional software design methodologies. For parsimony's sake, we present the set of design constructs that we believe will

help researchers and developers to think more clearly about Internet agent design, especially in the case of long-lived agents. We now turn to a fuller discussion of the agent sophistication features that comprise our *Agent Development Framework*.

The Six Agent Sophistication Features

Data Collection Task is determined by the scope of the application and describes the role of the agent in it. This translates into specific agent requirements or capabilities. The scope of the data collection task should conceptually determine the sophistication of an agent. For example, if a research project requires in-process decision-making on the part of the researcher, validation to determine if data are acceptable, and recovery from Web site crashes when data collection is in process, then an extremely sophisticated agent must be developed if these tasks are allocated to the agent. Conversely, if a research project simply requires retrieval of key terms from single web page, a relatively simple agent would suffice.

Although some descriptive elements are visible in our framework (e.g., one can measure an Internet agent's sophistication in terms of the six characteristics), we intend that they be used primarily as the basis for assessing what features should result from software development activities that lead to the creation of such an agent. A researcher should first define the data collection task and then decide on the appropriate level of sophistication needed for each feature. Characterizing levels and capabilities for each feature enables the development of support tools and reusable components that allow a designer to configure an appropriate agent.

Our framework is intended to aid in the development of long-lived data-collecting agents, but is applicable as well to short-lived agents. Data-collecting agents function as high-powered, sophisticated substitutes for the researcher's own constrained human efforts in data collection (e.g., consider human limitations with respect to patience, exhaustive search, attention to appropriate details and so on). Long-lived agents differ from short-lived agents in several fundamental ways, including the volume of data collected (megabytes compared to bytes), the type of data collected (a potentially large number of different types of incidents related to multiple actors compared to a small number of a single type of incident related to a single actor), and the life of the agent (hours or days compared to milliseconds or seconds). As a result, short-lived agents typically need not be as sophisticated as long-collecting agents. They need not permanently store data, they need not coordinate the collection of time-dependent and time-sensitive data and they do not need to recover data lost as a result of system failures.

Conversely, long-lived agents are concerned with each of these issues and more. Hence, features such as monitoring, concurrency, recovery, and validation are much more important to long-lived agents

than they are to short-lived agents. Features such as intelligence and interactivity can be equally important to both types of agents. We next consider the various features in our framework in greater detail.

Intelligence

Intelligence embedded in an agent allows it to respond to queries, to make decisions, and to learn from its experiences -- capabilities normally associated with human intelligence. *Intelligent agents* use rules and rule hierarchies to interpret the data and determine how to behave or how to change their behavior based on input and feedback from their environment. Jennings and Wooldridge (1998) point out that because an intelligent agent can act in place of a human expert, intelligent agents are *proactive*. However, care must be taken to ensure that the rules used by the agent to interpret the data are accurate and match the expert's knowledge within tolerable error limits. Note that while agents cannot possess the same level of intelligence as human experts, they do not suffer from traditional limitations of fatigue, cognitive overload, or biases that may affect the traditional researcher's decision processes.

The concept of intelligence in computer software has sparked a debate over whether software can possess *artificial intelligence (AI)*. Searle (1990) differentiates strong and weak AI. *Strong AI* indicates that a machine can actually "think" like humans by virtue of incorporating an AI computer program. Such programs initiate action, have beliefs that change, and have motivation. *Weak AI* occurs when computers implement a series of rules to imitate human action. Searle argues against the possibility of strong AI on the grounds that machines lack *intentionality* and, although a program can follow instructions to receive symbols for input and generate symbols for output, programs do not have any innate understanding of what they are doing. Searle charges AI proponents with attempting to incorporate more human qualities into machines than can be rationally justified. Other authors (Dennett 1991, p. 435), in response to Searle, argue that all that matters is the end result. If a computer program can imitate human thought and action with no distinction, then that computer program has strong AI.

Based on Searle's (1991) and Dennett's (1991) philosophical discourse, there are many different levels of AI that can be incorporated into agent design, but typically, these levels are grouped into three categories. *First*, agents without intelligence, termed *simple agents* or just *agents*, retrieve all specified data from a site without making any interpretations. *Second*, agents with simple rules to follow based on experts' or users' input are termed *expert agents*. Expert agents use weak AI in that they follow rules usually found in some form in a repository to mimic normal or nominal actions of the user. *Third*, agents use strong AI to initiate action, to have beliefs that can change, and to have motivation. Even though we want to remove ourselves from the debate of the possibility of strong AI, we contend that it is commercially infeasible at this time to develop Internet agents that exhibit strong AI. Currently, most

commercially available Internet agents that are categorized as "intelligent agents" are simple or expert agents. They tend to be rule-based agents that use weak AI to mimic some human behavior.

To stress the difference between traditional AI and intelligent systems, we define the *intelligence construct* in our framework as the ability of an agent to use rules to mimic human responses. Developers need to consider what, if any, human behavior needs to be mimicked. Knowledge bases that incorporate syntactic rules can be used to lead to specific behavior from an agent based on the researcher's needs. Agent behavior and intelligence, therefore, need to be considered when developing an agent. Intelligence is especially useful when *lexical parsing* is required. This involves viewing freeform text to identify the same characteristics of an item as an expert would from a verbal description (Plaunt and Norgard, 1998).

Validation

Validation is the capability of an agent to ensure that the data are of the proper type (e.g., date, time, number, etc.), are properly identified, and follow the rules specific to the research task. Validation includes the wide range of data quality characteristics (Wand and Wang, 1996) applicable to the task, such as the proper range and selection criteria, age, level of aggregation, units (e.g., U.S. dollars or British pounds), and so forth. In contrast to *validating agents* that perform this task, *non-validating agents* simply retrieve the specified data. Validation can enhance the extent to which an agent is proactive. For example, if certain rules must be followed for data to be valid, the agent can take appropriate action even if these rules are violated. If a database is used, part of the validation task can be delegated to the DBMS. However, if the database rejects invalid data, the agent may need to be informed or special precautions may need to be taken in the analysis of the collected data. By identifying invalid data, validating agents can process data much more effectively than non-validating agents.

Concurrency

Concurrent agents are agents that perform several tasks at the same time. In contrast, *single process agents* are agents that operate sequentially, and only can perform only one task at a time. A system that can run more than one task at a time can add to the efficiency of a program by taking advantage of wasted CPU cycles while other programs are waiting for responses from remote Web servers.

Concurrency is usually implemented through *multi-threading*. Multi-threaded agents have a central control thread that *spawns* processes (or *agent threads*) that each run concurrently. While some agent threads wait for information from Web sites, others can process data that have already been retrieved. The result is a more efficient use of computer resources and less idle CPU time. Hence, concurrency can reduce the time needed for data collection and add efficiency to large-scale data collection efforts.

For many small tasks, especially those typical of transaction agents, single process agents are suitable because the overhead required to manage multiple processes may be greater than the benefits achieved. However, for large data collection efforts or for fast multiple-site transaction information, concurrency will result in agents that retrieve data faster than is possible from single process agents. Concurrency is especially important for agents that need to monitor more than one web site at the same time. A single process agent can only monitor one Web site at a time, which may cause a monitoring agent to omit valuable data.

Mobility is a subset of an agent's capability to perform concurrency. *Stationary agents* run on a single computer. Data is collected from remote sites usually through the TCP/IP protocol, but the processing of the data occurs on a local machine, usually the machine of the researcher. *Mobile agents*, on the other hand, are spawned by a central controlling agent, but they request (and are granted) time to run processes concurrently on both local and remote sites. Currently, there is no standard for developing mobile agents. Nor is there broad-based agreement in the marketplace on the technical design requirements for mobile agents.

Recovery

Recovery describes the extent to which an agent is able to deal with situations that interrupt its data collection activities. For example, errors can occur with network connections, with remote Web servers, and with the middleware or database management systems that serve them. Agent exception handling is relatively new to the IS literature. Klein and Dellarocas (1999) describe multi-agent interactions where agents are difficult to maintain, understand and reuse because the relatively simple normative behavior of an agent becomes obscured by a potentially large body of code devoted to handling exceptional conditions. Those authors suggest generic routines that can be easily interfaced by agents in an agent system. For long-lived agents, the approach is similar. Generic error routines can be inherited from a common ancestor in an object-oriented development environment, allowing each agent to concentrate on its task without requiring massive error handling for every agent. Recovery routines can then be added that will enhance an agent's ability to recover from an exception.

Recovering agents can recover gracefully from any data collection interruption. For instance, imagine a data collection agent that has run for four hours collecting data from a remote Web server. Without any notice, that remote Web server goes down for "unscheduled maintenance" until the next morning. A recovering agent will demonstrate *persistence*, so that the next day, when the remote Web server comes back online, the agent will begin where it left off. A *non-recovering agent* will fail to work until the data has been erased or will erase existing data itself and start over. A *semi-recovering agent* will review the data it has previously retrieved, but will not re-process it.

Monitoring

Some agents simply collect a snapshot of the data available from a Web site. However, *monitoring agents* track events as they happen. Monitoring agents periodically access data from a Web site looking for changes. For example, it may be appropriate for a portfolio manager to monitor stock prices or for a corporate procurement specialist to record incremental price changes from various suppliers listed on the Web. Such data collection would be infeasible with traditional methods that force individuals to manually access a Web site, but would be simple for a monitoring agent.

Monitoring agents fall into two categories. *Continuous-monitoring* agents are designed to continue running during the entire monitoring process. *Scheduled-monitoring* agents are designed to run periodically, and often use scheduling software to run at a specific time during each hour, each day, or some other specified time period. Both these agents are designed to collect data and add it to data already collected and stored, usually in some database.

Interactivity

Interactivity gauges the extent to which an agent is able to modify its behavior based on the data it collects. *Non-interactive agents* simply retrieve pre-defined data content from pre-determined Web sites. *Interactive agents*, on the other hand, are *responsive* in that they not only collect data from a given web site, they also perform actions based on the data collected (Jennings and Wooldridge, 1998). For example, interactive agents can be used to provide passwords and user names, execute links to other pages, or enter data into fields in order to obtain the desired data. Note that the difference between interactivity and intelligence is that intelligent agents make decisions based on the data that are collected and interaction with an "expert" or user who the agents want to emulate, while interactive agents perform a set of *predetermined* procedures based on data that is collected.

DESIGNING A LONG-LIVED INTERNET AGENT

Long-lived data-collecting agents can be used to collect megabytes of data from the Web. These allow researchers to investigate many different kinds of phenomena such as, examining new patterns of user behavior, the dynamics of the new electronic markets of the Internet, and a spectrum of other search, usage and site effectiveness-related issues. Indeed, data-collecting agents can obtain vast amounts of data to enable research that would be difficult or impossible using traditional data-collecting techniques. Online auctions, such as on eBay or on Amazon.Com, have become premier spots for buyer and seller interactions, and transaction-making. eBay hosts literally millions of concurrent auctions each month. These auctions allow sellers to reach a wide variety of buyers through the Web, and they allow buyers to use the Web to find items either at a bargain or that are not available in traditional markets.

The Application Context: Transacting Collectible Coins Via the Internet

Our specific research interests in the electronic auction context deal with the relationship between observed seller behavior and buyer behavior. Very often, such situations involve problems with *informational asymmetries*, affecting the extent to which transactions can be made and how marketplaces perform on behalf of the participants. To illustrate the various concepts that we have discussed and to provide a "proof-of-concept" in this context, we next describe the design of an agent that gathers data from an online auction related to the items that are being bought and sold. The data pertains to three different levels of analysis: the *auction-level* (e.g., time started and finished, item being auctioned, etc.), the *seller-level* (in terms of observed seller reputation) and the *bidder-level* (e.g., maximum bid per item, time of bid, etc.).

Although some auctions transact goods whose quality is relatively or truly constant (e.g., computer peripherals and software), others transact goods whose quality is uncertain (e.g., used automobiles or collectible "Beanie Babies"). When the quality is uncertain, pictures and large amounts of free-form text are used to describe the condition of the good for sale. Such information is often difficult (though not impossible) for a data-collecting agent to process. There are, however, instances in which relevant though complex information pertinent in transaction-making context has been effectively codified. Rare coins and postage stamps are examples of such goods that have been sold and bought in market contexts for centuries.

The coin market is especially interesting in this respect. Over past decades, coin collectors have developed a special language to describe the quality of a coin in great detail. Indeed, the special language has taken on such importance in enhancing transaction-making that its use is a standard practice. Understanding the quality of a coin is crucial for the buyer, since without knowledge about quality, it is difficult to establish a basis from which to construct the coin's value, and, thus, the amount of money that should be bid to buy it. For instance, Gibbs (1999), who provides definitions for the spectrum of different observed quality for coins, defines the *Extremely Fine* grade as "light overall wear on highest points, but with all [coin] design elements sharp and clear" and the grade *Very Good* as "[the coin] design and surface are well worn, main features are clear but flat." Such a sophisticated language allows coin collectors to clearly communicate the quality of their items for sale.

eDRILL: An Electronic Data Retrieval Lexical Agent

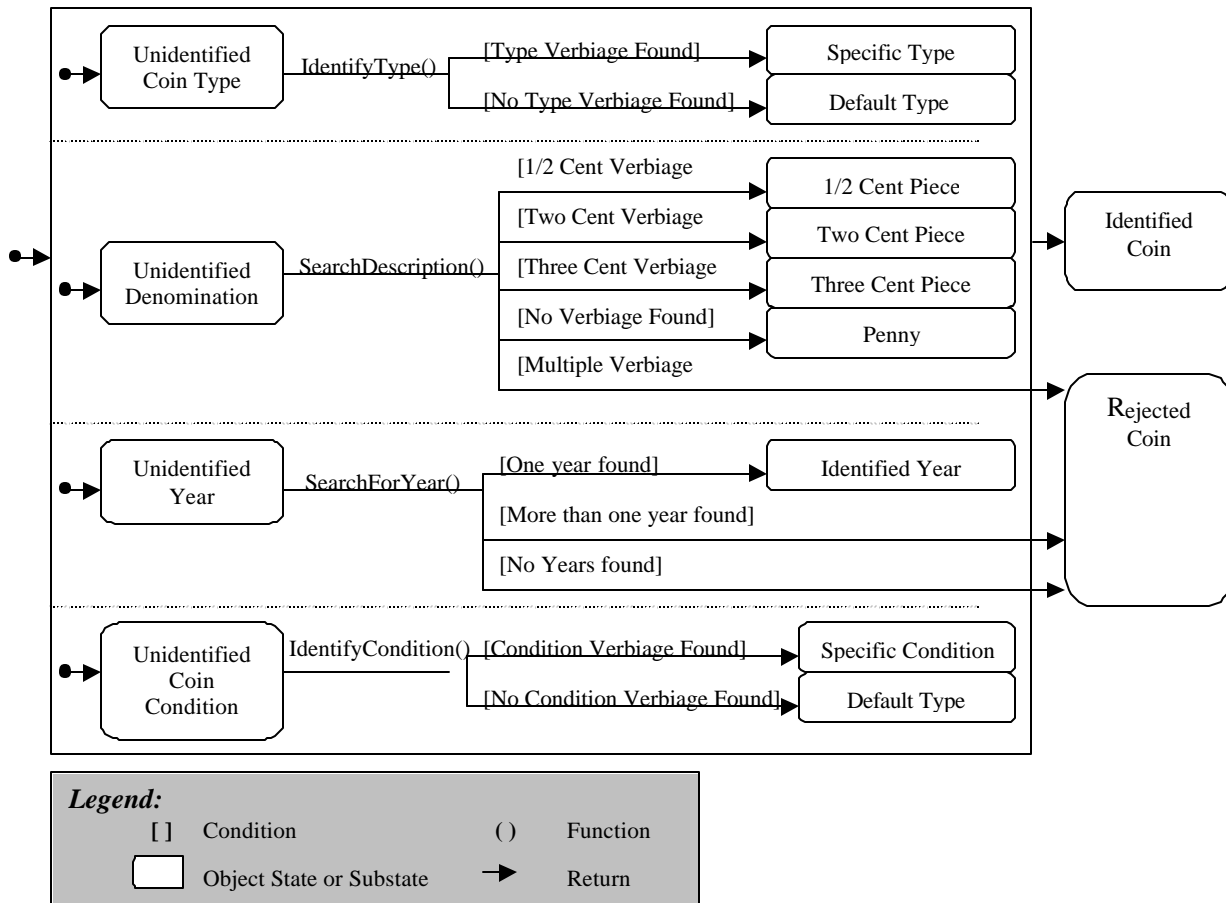
Using the framework described in Figure 1, we designed and implemented a research tool called the *Electronic Data Retrieval Internet Lexical (eDRILL)* Agent. eDRILL electronically "drills" into online auctions and collects data on bidder behavior specifically dealing with rare coin auctions. It can

lexically parse freeform text to determine the actual grade of the coin derived from the item name or item description listed in the online auction.

eDRILL includes rules that mimic intelligence because the agent takes the place of the researcher and must collect and process data, just as the researcher would. It must be able to differentiate coin types and grades, and must be able to identify comparable coins from a verbal description. Furthermore it must identify buyers and sellers and capture "last bids" for each bidder for comparable coins. To perform this task, programmatic rules are combined with a knowledge base that describes special features available for each coin to determine type, year, and denomination. This rule-based process allows the agent to identify coins in the same manner as a human expert would. We use a *UML concurrent state diagram* (Fowler and Scott, 2000) to specify how intelligence is used to identify a coin based on its description (Figure 2).¹ Each box identifies the state or sub-state of the identification while each function and condition combination defines a set of rules used to identify type, denomination, and mint year. If eDRILL cannot identify a coin or there are multiple coins for sale within one bid, eDRILL does not include that coin in the study.

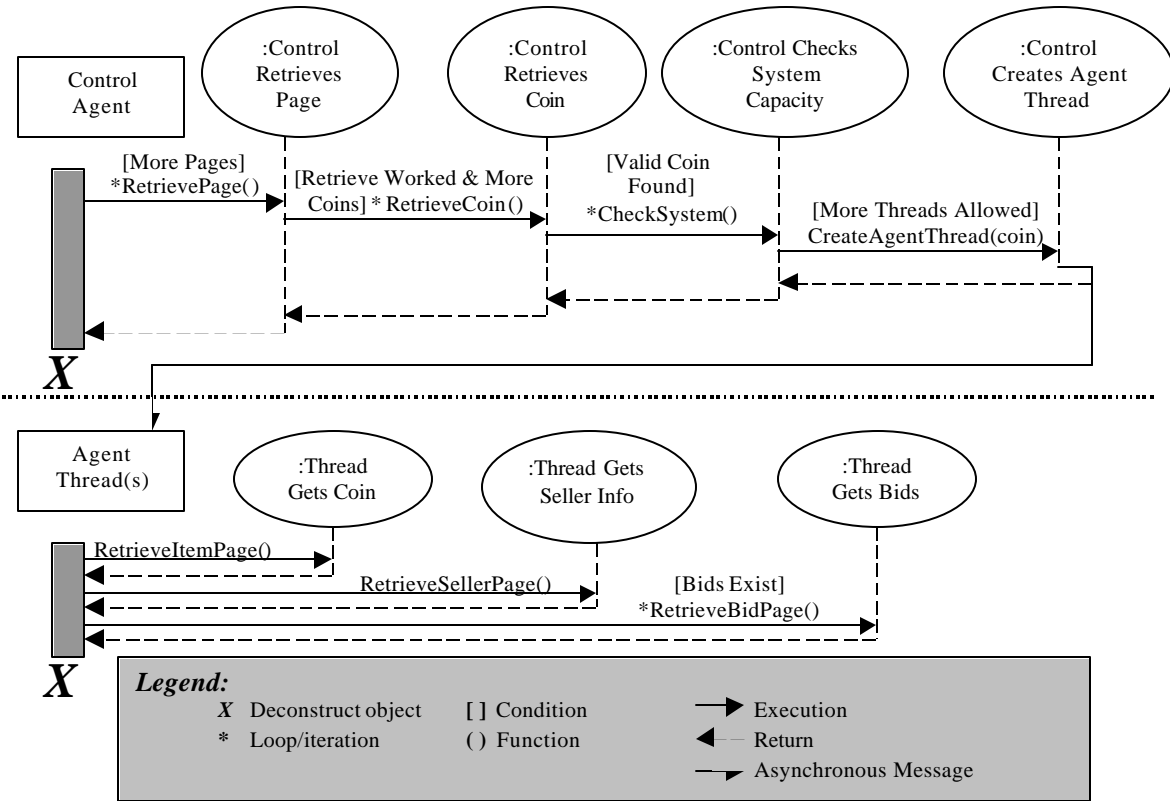
¹ Because our design, by itself, is not ontologically complete, we chose to place our constructs within the *Unified Modeling Language (UML)* to specify each of the agent sophistication features described above. The UML is ontologically complete (Fowler and Scott, 2000). It also provides us with a convenient means of expressing design concepts and reflects the current "best practice" in industry (Booch, Rumbaugh and Jacobson, 1997; Fowler and Scott, 2000). UML is actually a series of representation formalisms and tools that developers can integrate to describe complex software requirements concepts. Thus, UML is well suited for describing design features in our *Agent Development Framework*.

Figure 2. Intelligence Diagrammed with a UML Concurrent State Diagram



eDRILL validates dates and denominations to ensure that only appropriate data are used. This step is especially important since the format of online auction Web sites changes based on the information they contain. This agent also is interactive. It interacts with the researcher to determine what coins are valid for this research and interacts with the online auction to navigate through its various web sites and pages to find the information it needs. For each coin tracked, eDRILL examines four different Web pages to retrieve bidding information and seller reputation information. With thousands of coins to be tracked, tens of thousands of Web pages must be accessed. For this enormous task, the agent utilizes concurrency. eDRILL creates multiple thread agents, each of which can utilize the CPU while others are waiting for information from other computers. This allows for more efficient processing and can significantly reduce task run-times. Figure 3 uses a *UML concurrent sequence* (Fowler and Scott 2000, p. 69) to describe eDRILL's concurrency capabilities.

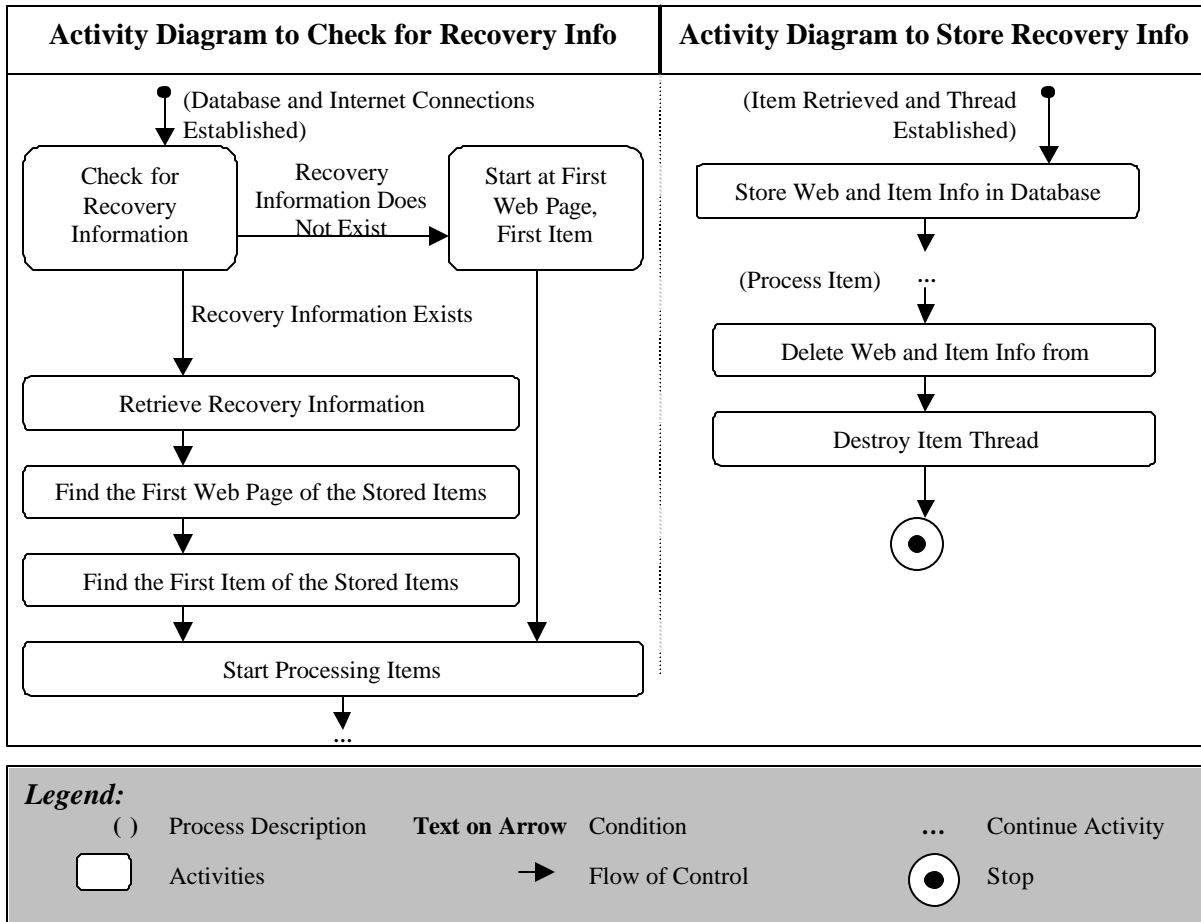
Figure 3. Agent Concurrency Diagrammed with a Concurrent Sequence Diagram



eDRILL utilizes multi-threading to increase efficiency. A *control agent* continuously retrieves new coins to be processed while checking system capacity to see if new *agent threads* can be created to process each coin. If a new thread can be created, the control thread sends an *asynchronous message* (indicated by the half-arrowhead) to initiate a new agent thread so that the new agent thread can execute simultaneously with the control agent while the control agent continues to initiate and monitor other threads. The control agent and all agent threads perform their processes concurrently to take maximum advantage of the computer's CPU, memory, and knowledge base.

To retrieve the information needed for our research design from the online auction, eDRILL ran for two days. When problems occurred during execution, such as a lost server connection, a database error, or an unexpected result, the agent terminated. eDRILL was designed to start at about the same point it left off. Recovery is complicated because multiple threads are extracting data from multiple web pages concurrently. eDRILL records the state of all threads in progress, and is able to re-start each appropriately. Figure 4 shows *UML activity diagrams* that describe eDRILL's recovery capabilities.

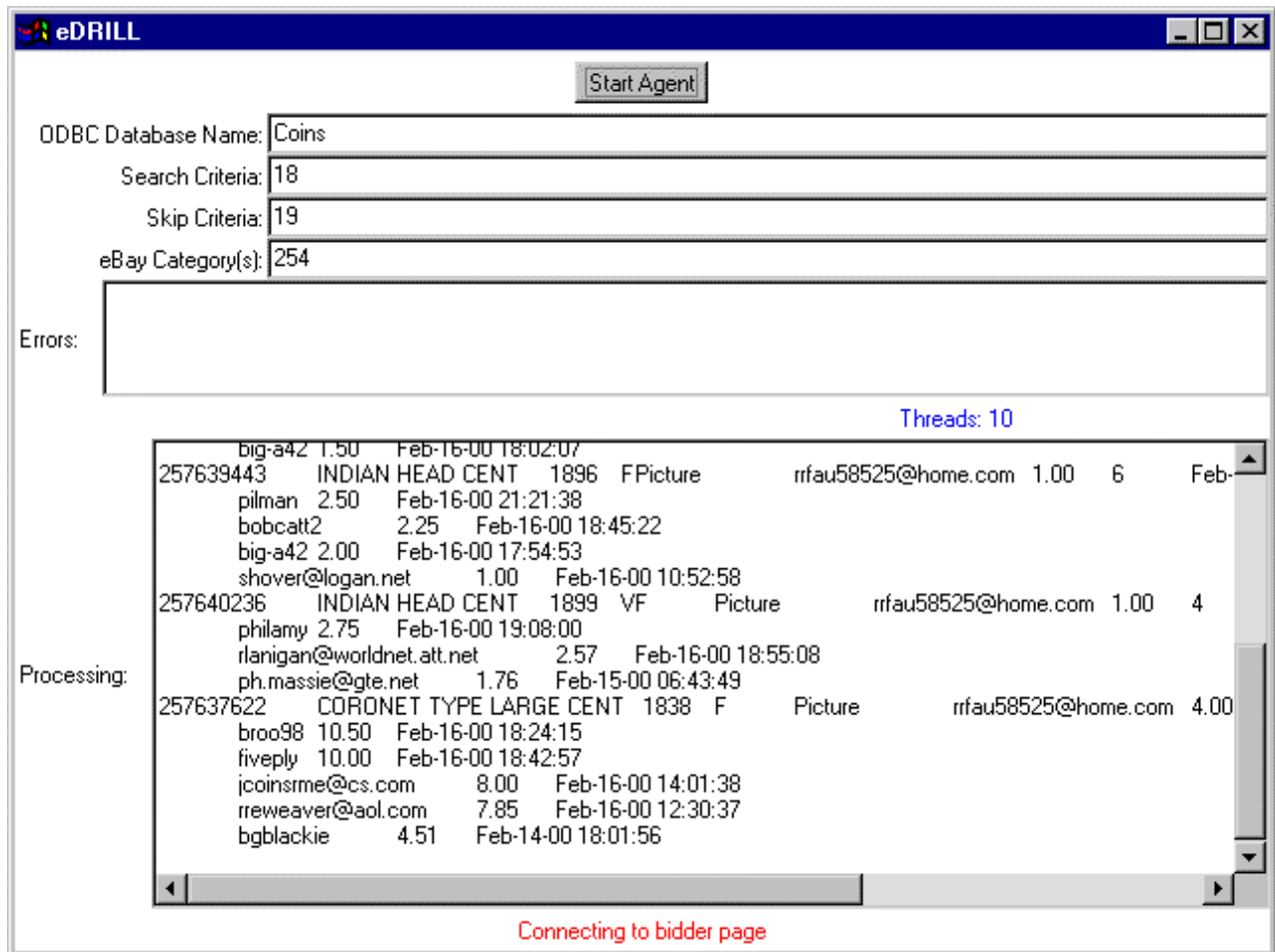
Figure 4. Recovery Diagrammed with Activity Diagrams



ILLUSTRATING THE POWER OF A DATA-COLLECTING AGENT

Data-collecting agents run over the Internet, accessing sites, then retrieving, parsing and storing information from those sites. Data-collecting agents can retrieve megabytes of data from a large population. Their use resolves many of the precision, realism and generality issues found in other data-collecting methodologies (McGrath, 1992). The prior section used UML diagrams to document the agent sophistication requirements of eDRILL, a data-collecting agent we used to study buyer behavior in rare coin auctions. In this section, we describe the implementation and use of eDRILL in this context. eDRILL is implemented in the Java programming language and runs in a Windows environment. Figure 5 shows a screenshot of eDRILL in the process of collecting coin data from an online auction.

Figure 5. eDRILL and Its Use on Online Auctions for Coin Collecting Data



As shown in figure 5, the user can eDRILL and view its process. The user must define an ODBC database in the *ODBC Database Name* field. This database is used to store the data that is collected. In this case, the “Coins” database is used. The user can optionally enter *Search criteria* and *Skip criteria* to include only those records containing specific text and skip criteria that skips records containing specific text. In Figure 5, records containing “18” are included to include only those coins made in the 19th century (18xx). Records containing “19” are skipped to help alleviate any coins containing both an 18 and a 19 (e.g., “1918”, “1888 Indian Head and 1909 VDB Penny”, etc.). Finally, the user can enter one or ebay category in the *eBay Category(s)* field that describes the auction category or categories that are “drilled” to collect data. In Figure 5, Category “254”, eBay’s rare penny category, is entered.

When the *Start* button is clicked, the agent reports its processing and any errors that it may encounter. Also reported are the number of errors, the number of open threads, the current task that eDRILL is attempting. eDRILL can be used to collect bid and item data from any eBay site. Use of this

tool not only allows data collection that would be beyond the scope of individual data collection efforts, but eDRILL also ensures a consistently correct data entry, eliminating potential human coding errors.

Exploratory Analysis

We next illustrate the power of eDRILL to capture relevant data for examining "micro-level" behavior in the context of an Internet auction. We present two separate empirical analyses: an exploratory examination of bid timing and a quasi-experiment involving graphical pictures that are presented along with trade item information, and the extent to which these promotes greater willingness-to-pay on the part of bidders.

Background. There are a number of studies that have examined Internet auctions from a managerial perspective.² Beam and Segev (1998) provide background on auction markets on the Internet, all the way back to their earliest appearance in 1995 and 1996. In addition, Beam, Segev and Shantikumar (1996) discuss Internet auctions (in terms of the early version of www.onsale.com) as an environment for business transactions and negotiation. Van Heck and Vervest (1998) offer advice to senior managers about how to make sense of the emerging electronic markets of the Internet. And, Bakos (1998) discusses Internet auctions in the broader context of "electronic marketplaces," and their characteristics and function in the emerging Internet economy.

In addition, there are a number of "behavioral" studies that examine the manner in which Internet auction buyers and sellers interact at a micro-level of analysis, and what this means for the design, operation, and performance of such markets. Bapna, Goes and Gupta (2000a) offer a typology of Internet auctions, and discuss their mechanics, why common assumptions about buyer and seller behavior may not hold in such markets, and other structural characteristics. On the basis of their analysis of e-auction data, they describe how lot size, opening bid amount, the magnitude of closing bids, and the pre-specified bid increment all affect the revenue of the Internet auction intermediary. A related paper by the authors explores the micro-performance of multi-bid item Internet auctions (Bapna, Goes and Gupta, 2000b), and reports on extensive data collection from various auction sites on the World Wide Web

Vakrat and Seidmann (1999) examine how much consumers were willing to pay for identical products offered through online auctions versus online catalogs. They obtained data from 473 online auctions (e.g., www.surplusauction.com and www.onsale.com), as well as from Internet catalog sellers

² In addition to these Internet auction-focused studies, there is a wealth of theoretical and empirical papers that deal with the design, mechanics and performance of auctions in other non-Web settings. They include, for example, Bulow and Roberts (1989), Harris and Raviv (1981), McAfee and McMillan (1987), Milgrom (1989), Milgrom and Weber (1982), Myerson (1981), Rothkopf and Harstad (1994), Varian (1995) and Vickery (1961). This body of literature from Accounting, Economics and Finance provides the theoretical basis for research exploration in the area of Internet auctions.

(e.g., www.egghead.com, www.pricescan.com and www.jango.com). Their data analysis revealed that consumers prefer e-auctions of short duration and expect greater discounts for more expensive items. A second study by the authors (Vakrat and Seidmann, 2000) of 324 online auctions analyzed bidder arrivals in Internet auctions, and found that about 70% of the bidders arrive during the first half of time the auction is open. In addition, the authors reported that "a higher minimum bid tends to result in fewer bidders, while offering more units resulted in an increased number of bidders" (Wall Street Journal, 1999). With the emergence of new data collection tools, many studies in this vein become possible, increasing the likelihood that researchers and managers can better understand the rich fabric of participant behavior in electronic marketplaces.

Bid Timing. We consider one aspect of micro-level bidder behavior in the context of Internet-based electronic rare coin auctions. Using the information gathered by eDRILL, we explored bid timing and the effects of an auction and item characteristics on what a bidder is willing to bid. Gathering and coding information on 49,021 bids would have been impossible without either technical assistance (such as an agent) or the express permission and assistance of the online auction to provide data and allow us to analyze it. Table 1 gives descriptive statistics of these bids.

Table 1. Descriptive Statistics of Bid Data Gathered by eDRILL

Descriptive Category	Observation
Number of Bids	49,021
Number of Items That Received Bids	12,742
Number of Bidders	7,537
Number of Sellers That Sold Items	1,924
Average Days Left in Auction When Bidders Entered Their Final Bid	2.1
Average Bid	\$46.40
Average Sale Price	\$60.71

Conventional wisdom and anecdotal evidence state that the most bid activity in an online auction takes place in the last hours of the auction. Given that, it is surprising that there is a relatively high average of 2.1 days left in an auction when bidders enter their final bid for items in our data set. We performed further exploratory analysis that may help explain why the conventional wisdom differs from the information gathered by our agent. To do this, we placed bidders into one or more of three categories. Table 2 shows these categories, which include:

- (a) bidders whose final bids for items occurred only during last day of the auctions where they bid;
- (b) those whose final bids occurred on the last day but also at other times;
- (c) those that never bid on the last day; and,
- (d) the total of all bidders.

Table 2. Bidder Exploratory Analysis

	Bidders Who Bid on the Last Day			Total (d)
	Only (a)	w/ Other (b)	Not (c)	
Number of Bids	7,782	36,858	4,381	49,021
Number of Bidders	2,589	2,595	2,353	7,537
Number of Bidder-Seller Combinations	6,284	26,638	3,581	36,503
Number of Winning Bids	3,528	8,856	358	12,742
Ratio: Winning Bids to Bids	45.3%	24.0%	8.2%	26.0%
Ratio: Bids to Bidders	3.0	14.2	1.9	6.5
Ratio: Sellers to Bidders	2.4	10.3	1.5	4.8

The anecdotal evidence of a large portion of the bidding occurring at the end of an auction appears to be somewhat true. It seems that the *most successful* bidders are those that only bid on the last day of an auction, with a 45.3% success rate.³ This means that if you bid on an item during the last day, you were 45.3% likely to get the item, even considering that other last-day bidders also had bid on the item. As can be seen in Table 2, bidding early is hardly ever a successful strategy, with only 8.2% of those early bidders who have never bid on the last day actually winning. However, with 2,353 bidders never bidding on the last day, nearly 31% appear to employ this early-bidding strategy. At first glance, it seems quite irrational for these 31% to take time to investigate items for sale that are not ending that day.

Bapna, Goes and Gupta (2000a) also point out how a burst of bidders arrives at the beginning of an auction, while new bidders trickle in throughout the rest of the auction. They describe three different types of bidders that have different bidding patterns in multi-item auctions: evaluators, participators, and opportunists (Bapna, Goes and Gupta 2000b). *Evaluators* place a single high bid early in the auction. Evaluators do not typically bid in non-traditional (single-item) auctions. *Participators* place minimum bids and follow an auction to its completion. *Opportunists* enter the market at the final moments before the auction closes. Evaluators apparently are the only type among the three that bid early in the auction, and they tend to bid higher to increase their chance of winning the auction. Furthermore, Bapna, Goes and Gupta indicate that this type of bidder is not viable in traditional single-item auctions. It seems unlikely

³ The reader should keep in mind that the strategy of bidding infrequently and primarily on the final day of an auction is quite well understood in the marketplace, even by amateurs. For example, electronic auction bidding software is readily available via the Web and it enables the sort of bidding behavior that is consistent with what we know about the last day and last minute dynamics of bidding in Internet auctions and elsewhere. See, for example, information on “TurboBid,” a piece of software that permits a high volume bidder on eBay to “snipe” or bid just prior to the close of an auction, to enhance the chances of taking the deal at a price that is no higher than necessary to win (www.etusa.com/auction/robobid.htm). This software permits the bidder to set up for stretch run bidding on eBay for between 20 and 300 seconds prior to the auction’s close.

that evaluators are the cause for the high number of early bidders, and participators and opportunists tend to bid up to the final moments of the auction.

Now, consider three facts presented in Table 2. *First*, early bidders are less likely to win the auction. *Second*, bidders who do not bid on the last day of an auction bid on items from far fewer sellers. Their sellers-to-bidders ratio of 1.5 (compared with 2.4 for last-day bidders and 10.3 for other bidders) indicates that they are much less likely to bid on items from several bidders, and rather concentrate on one or a few sellers. *Third*, early bidders do not seem to bid on as many items, having a bids-to-bidders ratio of only 1.9. Now consider that online auctions allow users to set up as many selling and bidding handles as they have email addresses. Sellers and bidders could be setting up as many *anonymous* accounts as they wish.

Although these facts are not conclusive, considering the fact that online auctions allow users to set up as many selling and bidding handles as they have email addresses, they offer preliminary support for an hypothesis that *sellers are bidding on their own items*, trying to influence a final bid price. *First*, a seller would not want to win her own auction, but would rather bid to give the illusion of outside interest in an item or to ensure the item does not sell for too little. She would prefer that someone else buy her item at an inflated bid price. *Second*, sellers who are bidding on their own items would not necessarily be buyers in the auction. Many would tend to only bid on their own items to influence the price. This would cause those early bidders to show interest in fewer sellers and in fewer items.

There are other possible explanations too, but none fit as well as the hypothesis that sellers are bidding on their own items. *Bottom-fishing* (trying to find items for extremely low prices) would cause bidders to seek out items for a low price, bid on them and hope that they get them at a bargain. However, while bottom-fishing could explain why bidders bid early because lower bids can be submitted early, it goes against the second and third observations. Because bottom-fishers would look at several items, there is no reason to think that bottom-fishers would bid on fewer items or limit themselves to fewer sellers. *Inexperienced users* may want to limit their number of bids and sellers until they more fully understand the online auction market microstructure. But there is no reason to assume that inexperienced users would not follow an item to the final day, especially since they will have fewer items to follow than many-item users and would be interested to see if their bid succeeded.

Also, as the *winner's curse* literature states, uncertainty about value leads to bidders paying too much for an item in an auction (Lederer, 1994). Since it is likely that the inexperienced bidders are most likely to be uncertain about a coin's value, they are the ones that tend to pay too much for an item in an auction and, therefore, end up winning, not losing, the auction by paying too much. As this is only exploratory analysis, future research involving more carefully crafted research questions and a more

refined experimental setup are required to see if our hypothesis that sellers bid on their own item early in the auction is indeed true. More research is required in this area.

Bidder Behavior. Experiments have often been used by social scientists to control the way in which data are gathered to study complex relationships among constructs. By stripping out environmental "noise" in a lab setting, researchers can more accurately examine those relationships. Lately, however, some researchers have pointed out the weaknesses of lab experiments. Lave (1980) points out that the environment often affects people's actions, and that you cannot accurately judge people's actions once you take these people out of their environment. McGrath (1992) discusses how experiments allow a high degree of precision, but are "dilemmatic" because they lack realism and generality. Reips (1997) goes on to discuss the limitations of experiments that include sample size limitations, experimenter bias, and corrupted subjects that skew results. With a little creativity a researcher can often use data collected by an agent to perform a quasi-experiment with identical stimuli and controls to that of the corresponding lab experiment. Quasi-experimentation using agents has the potential to resolve all of the weaknesses of lab experiments described by Lave, McGrath, and Reips.

Using the eDRILL data, *we examine how a picture affects the behavior of individual bidders.* The availability of pictures gives product information beyond what the seller provides via description and allows for easy comparison among products. Again, we initially examined 49,021 bids. As described earlier in this paper, rule-based algorithms determined the specific year, denomination, and type of coin as well as the grade. These were derived from the item name and description listed by the seller. Grades were assigned numerically using ordinal number assignments. These assignments corresponded to ratings used by expert coin collectors. Any coin that could not be graded or dated was dropped from the study. Any multiple coin listing (e.g., "1858 Flying Eagle and a 1968 Indian Head both for cheap") was also dropped. One of the "industry bibles" for coin collecting, *Coin World*, was used to identify specific coins (Gibbs, 1999).

Data that existed had to be manipulated in three ways for this experiment. *First*, since demand is measured through price, the highest bid was dropped. Losing bids contain information as to how high a price the bidder is willing to pay for an item, but the winning bidder may have been willing to pay more if challenged. *Second*, to be considered in our sample, a bidder needed to bid on the *exact same coin* (e.g., "1855 Coronet Large Cent") in *exactly the same condition* (e.g., "VG-8 or Very Good") at least *twice*, at least once *with a picture* (stimulus) and at least once *without a picture* (control). All those that did not fit into this category were dropped from the data set. *Third*, an average bid price was computed for each picture and non-picture group. Then a percentage was assigned for each purchase based on the difference between the average amount for that bidder/product combination and the actual amount paid. Based on

the research questions and propositions, several hypotheses were tested. Two variables, *AverageBid* and *AveragePercent*, were derived for each product:

$$\mathbf{AverageBid} = (\mathbf{Average(bids\ with\ pictures)} + \mathbf{Average(bids\ without\ pictures)}) / 2$$

The *AverageBid* variable was derived to minimize effects of bidders who bid on many of the same products. Using the *AverageBid*, *AveragePercent* is derived as the percentage a bid is above or below the *AverageBid*:

$$\mathbf{AveragePercent} = (\mathbf{Bid} - \mathbf{AverageBid}) / \mathbf{AverageBid}$$

For example, the bidder listed in Table 3 bid on three very fine 1855 coins. The first two of the coins had pictures while the last one did not. Table 3 shows the sample bids for this item, including the calculated *AverageBid* and *AveragePercent*.

Table 3. Sample Bids

Bidder	Item Name	Picture	Bid	Average Bid	Average Percent
joe23	1855 LARGE CENT "YOUNG HEAD TYPE" LOOK "N/R"	Yes	\$9.25	\$9.3975	98%
joe23	1855 Large Cent, VF.*PIC*	Yes	\$13.00	\$9.3975	138%
joe23	1855 LARGE CENT- REAL NICE [<i>Very fine rating in the extended description</i>]	No	\$7.67	\$9.3975	82%

AverageBid for this series of bids is calculated by the following formula:

$$\mathbf{AverageBid} = (((\mathbf{\$9.25} + \mathbf{\$13.00}) / 2) + \mathbf{\$7.67}) / 2 = \mathbf{\$9.3975}$$

AveragePercent for this series of bids is calculated by the following formula:

$$\mathbf{AveragePercent}_1 = \mathbf{\$9.25} / \mathbf{9.3975} = \mathbf{98\%}$$

$$\mathbf{AveragePercent}_2 = \mathbf{\$13.00} / \mathbf{9.3975} = \mathbf{138\%}$$

$$\mathbf{AveragePercent}_3 = \mathbf{\$7.67} / \mathbf{9.3975} = \mathbf{82\%}$$

Of the 49,021 bids, there were 499 different bidders who bid on the exact same coin⁴ *at least twice*, at least once with a picture [stimulus] and at least once without a picture [control]. Picture bids were then compared with average percent. Table 4 shows the results of the study.

Table 4. Picture Premium Effects

Test	Picture Premium	t-Statistic	N
1999	6.4%	3.39***	303
2000	13.0%	5.00***	196
Total	9.0%	5.84***	499

Legend: *** = significant at the 1% level

This exploratory study shows an increasing effect of pictures on the price bidders were willing to pay to purchase rare coins. In 1999, items with pictures commanded a *picture premium* of a 6.4% compared to identical items without pictures. In the January 2000 data, the picture premium increased to 13.0%. If this trend towards an increase picture premium were to be sustained over time, this would indicate some sort of learning curve. It might occur *en masse*, with the result that bidders begin to refuse to pay full price if a seller offers a coin without an accompanying picture to enable the buyer to gauge quality. Clearly, more research is required to reach any definitive conclusions.

eDRILL Evaluation

To close out this discussion we concentrate on the methodology itself. In the first study, data was collected on 49,021 bids. Analysis of this data revealed some unexpected information about the micro-level behavior of coin bidders in an Internet-based auction. Since this was exploratory research, new data can be easily collected using the same data-collecting agent to gather data for rigorous empirical tests to determine the likelihood of results. In the second study, the same number of bids, 49,021, was distilled to include just 449 subjects who were measured to see how the existence of a picture affected their bidding patterns.

Now consider a corresponding lab experiment. The reader should recognize that it would be difficult to find 449 subjects for *any* experiment in just about any context. Furthermore, setting up such an experiment that would accurately and realistically recreate a setting that could be generalized to the online auction environment would be a monumental task. Using agents, we were able to get a much larger and truly representative sample. We were able to define a stimulus and control, as with an experiment, and we were able to observe actions without any contamination of subjects. We ended up

⁴ In this study, coins are defined as the same if they have the same mint year (e.g., 1898, 1802), same denomination (e.g., penny, two-cent piece), same type (e.g. double-die, New Orleans Mint, 7 strike over 8, etc.), and same condition as defined by expert coin collectors (e.g., MS-64 Uncirculated, VG-10 Very Good, VF-20 Very Fine, etc.)

with an initial test that is also generalizable and realistic, all with reduced cost and reduced time via a data-collecting agent that was designed using our framework.

RESEARCH CONTRIBUTION AND FURTHER STUDY

This research makes three contributions. *First*, we introduce design principles for long-lived agents, such as those used to collect data for research purposes. Most agent design research concentrates on short-lived transaction agents that require seconds to run and retrieve little data. As a result, it ignores concepts such as concurrency and recovery that are vital to long-lived agents that run over longer periods of time and need store significant amounts of data for alter analysis. *Second*, we present a framework of features significant to the development of long-lived agents and describe how this framework has been utilized to define a general methodology for the development of all Internet agents. This methodology was then illustrated in the development of eDRILL, an Internet agent used to collect data on bidder behavior in an electronic auction.

Third, we show how data gathered with our agent in exploratory and quasi-experimental designs provides an efficient mechanism for the discovery of new information and perspectives about micro-level buyer and seller behaviors in Internet-based electronic auctions. Although the results presented here are only meant to be illustrative, they nevertheless provide the reader with a good idea of how a data-collecting agent can be applied in the context of a well-defined research effort. Because of the powerful data collection allowed by data-collecting agents, these studies shed some light on some interesting electronic commerce phenomena. Although we discuss comparisons with other methods for data collection, further research is needed to understand how to maximize the effectiveness of this new approach.

This research is valuable for researchers and practitioners. Researchers will be able to take the concepts developed here to develop their own agents. EC research will develop faster once the ability to collect data is enhanced. Practitioners will also be able to consider the elements of agent sophistication when developing their own agents. By considering other factors when developing their agents, their agents will be useful for longer-lived tasks. Both marketing departments who need to conduct large-scale research efforts and software developers who design agents for use on their Web sites will benefit from the insights provided in this research.

REFERENCES

Bakos Y. 1998. Towards Friction-Free Markets: The Emerging Role of Electronic Marketplaces on the Internet. *Communications of the ACM* 41(8): 35-42.

- Bapna R, Goes P, Gupta A. Forthcoming, 2000a. Online Auctions: Insights and Analysis. *Communications of the ACM*.
- Bapna R, Goes P, Gupta A. Forthcoming, 2000b. A Theoretical and Empirical Investigation of Multi-item On-line Auctions. *Information Technology and Management*.
- Beam C, Segev A, Shantikumar G. 1996. Electronic Negotiation Through Internet-Based Auctions, *Working Paper 96-WP-1019*, Fisher Center for Management and Information Technology, Haas School of Business, University of California, Berkeley, CA.
- Beam C, Segev A. 1998. Auctions on the Internet: A Field Study. *Working Paper 96-WP-1019*, Fisher Center for Management and Information Technology, Haas School of Business, University of California, Berkeley, CA.
- Bernstein PS, Hadzillacos V, Goodman N. 1987. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, Reading, Massachusetts.
- Bogonikolos N, Fragoudis D, Likothannasis L. 1999. Archimedes: An Intelligent Agent for Adaptive Personalized Navigation within a Web Server. In *Proceedings of the 32nd International Conference on Systems Science (HICSS)*, Maui, HI, Shriver B, Sprague R (eds). IEEE Computer Society Press, Los Alamitos, CA.
- Booch G, Rumbaugh J, Jacobson I. 1997. *Unified Modeling Language Users Guide*. Addison Wesley Longman, Inc., Reading, MA.
- Bulow J, Roberts J. 1989. The Simple Economics of Optimal Auctions. *Journal of Political Economy* 7(5): 1060-1090.
- Bernstein PS, Hadzillacos V, Goodman N. 1987. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, Reading, Massachusetts.
- Chavez A, Maes P. 1996. Kasbah: An Agent Marketplace for Buying and Selling Goods. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, London, UK.
- Coste-Maniere E, Wang HH, Rock SM, Peuch A, Perrier M, Rigaud V, Lee MJ. 1996. Joint evaluation of mission programming for underwater robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, Minneapolis, MN. IEEE Computer Society Press, Los Alamitos, CA: 2492-2497.
- Dennett DC. 1991. Consciousness Imagined. In *Consciousness Explained*, Dennett DC. Little, Brown and Company, Boston: 431-435.
- Etzioni O, Weld D. 1994. A Softbot-Based Interface to the Internet. *Communications of the ACM* 37 (7): 72-75.
- Fowler M, Scott, K. 2000. *UML Distilled 2nd Edition: A Brief Guide to the Standard Object Modeling Language*. Addison Wesley Longman, Reading, MA.
- Gibbs W. 1999 *Coin World Guide to U.S. Coins: Prices and Value Trends*. New York, Signet Publishing.
- Groover MP, Weiss M, Nagel RN, Odrey NG. 1986. *Industrial Robotics*. McGraw-Hill, New York.
- Harris M, Raviv A. 1981. Allocation Mechanism and the Design of Auctions. *Econometrica* 49 (6): 1477-1499.

- Jennings NR, Wooldridge MJ. 1998. A Brief Introduction to Software Agent Technology. In *Agent Technology: Foundations, Applications and Markets*, Jennings NR, Wooldridge MJ (eds). Springer-Verlag, New York: 29-48.
- Keen JS, Dally WJ. 1997. Extended Ephemeral Logging: Log Storage Management for Applications with Long-lived Transactions. *ACM Transactions on Database Systems* 21 (1): 1-42.
- Keen JS, Dally WJ. May 1993. Performance Evaluation of Ephemeral Logging. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Washington DC: 187-196.
- Klein M, Dellarocas C. Exception Handling in Agent Systems. 1999. In *Proceedings of the Third International Conference on Autonomous Agents*, Seattle, Washington. ACM Press, New York: 62-68.
- Lave J. 1980. What's Special about Experiments as Contexts for Thinking. *The Quarterly Newsletter of the Laboratory of Comparative Human Cognition* 2(4): 86-91.
- Lederer, P. 1994. Predicting the Winner's Curse. *Decision Science* 25 (1): 79-101.
- Leebaert D (ed). 1998. *The Future of the Electronic Marketplace*. The MIT Press, Cambridge, MA.
- Lewis FL, Fitzgerald M, Liu K. 1996. *Robotics*. ACM Computing Surveys 28(1): 81-84.
- Machlis S. March 22, 1999. Agent Technology. *Computerworld* 33 (12): 69.
- Maes P. 1994. Agents That Reduce Work and Information Overload. *Communications of the ACM* 37 (7): 30-40.
- Maes P, Guttman RH, Moukas AG. 1999. Agents That Buy and Sell. *Communications of the ACM* 42(3): 81-87.
- McAfee RP, McMillan J. 1987. Auctions and Bidding. *Journal of Economic Literature* 25: 699-738.
- McGrath JE. 1992. Dilemmatics: The Study of Research Choices and Dilemmas. In *Judgment Calls in Research*, McGrath JE, Martin J, Kulka RA (Eds.). Sage Publications, Beverley Hills: 69-102.
- Milgrom P. 1989. Auctions and Bidding: A Primer. *Journal of Economic Perspectives* 3: 3-22.
- Milgrom P, Weber R. 1982. A Theory of Auctions and Competitive Bidding. *Econometrica* 50: 1089-1122.
- Mougayar W. 1998. *Opening Digital Markets: Battle Plans and Business Strategies for Internet Companies*. McGraw-Hill, New York.
- Mui C, McCarthy WE. 1987. FSA: Applying AI Techniques to the Familiarization Phase of Financial Decision Making. *IEEE Expert* 2(3): 34-41.
- Myerson RB. 1981. Optimal Auction Design. *Mathematics of Operations Research* 6: 58-73.
- Pardo-Castellote G, Schneider SA, Cannon RH. 1995. System Design and Interfaces for Intelligent Manufacturing Workcell. In *Proceedings of the International Conference on Robotics and Automation*, Nagoya, Japan. IEEE Computer Society Press, Los Alamitos, CA.
- Plaunt C, Norgard BA. 1998. An Association-based Method for Automatic Indexing with a Controlled Vocabulary. *Journal of the American Society for Information Science* 49 (10): 888-902.
- Nwana H. 1996. Software Agents: An Overview. *Knowledge Engineering Review* 11(3): 1-40.
- Reips UD. 1997. Das Psychologische Experimentieren im Internet [Psychological experimenting on the Internet]. In *Internet für Psychologen*, Batinic B(ed). Göttingen, Germany: Hogrefe, p. 245-265. (English abstract available at: <http://www.psych.unizh.ch/genpsy/Ulf/Lab/WWWExpMethod.html>.)

- Rothkopf MH, Harstad RM. 1994. Modeling Competitive Bidding: A Critical Essay. *Management Science* 40 (3): 364-384.
- Searle JR. January 1990. Is the Brain's Mind a Computer Program? *Scientific American* 262(1): 26-31.
- Shoham Y. June 1993. Agent-Oriented {P}rogramming. *Artificial Intelligence* 60: 51-92.
- Sloman A, Logan B. 1999. Building Cognitively Rich Agents: Using the SIM_AGENT Toolkit. *Communications of the ACM* 42(3): 71-77.
- Tu HC, Lyu ML, Hsiang, J. 1999. Agent Technology for Website Browsing and Navigation. In *Proceedings of the 32nd International Conference on Systems Science (HICSS)*, Maui, HA, Shriver B, Sprague R (eds). IEEE Computer Society Press, Los Alamitos, CA.
- Vakrat, Y, Seidmann, A. 1999. Can Online Auctions Beat Online Catalogs? In *Proceedings of the 20th International Conference on Information Systems (ICIS '99)*, Charlotte, NC, De P, DeGross J (eds).
- Vakrat, Y, Seidmann, A. 2000. Implications of the Bidders' Arrival Process on the Design of Online Auctions. In *Proceedings of the 33rd Hawaii International Conference on Systems Science (HICSS)*, Maui, HI, Sprague R (ed). IEEE Computing Society Press, Los Alamitos, CA.
- Van Heck E, Vervest P. 1998. How Should CIO's Deal With Web-Based Auctions? *Communications of the ACM* 41(7): 99-100.
- Varian, H. "Economic Mechanism Design for Computerized Agents," USENIX Workshop on Electronic Commerce, July 11-12, 1995, New York, NY.
- Vickery W. 1961. Counter-speculation Auctions and Competitive Sealed Tenders. *Journal of Finance* 41: 8-37.
- Wall Street Journal. September 23, 1999. A Special Background Report on Trends in Industry and Finance: Online Auctions May Elicit Lower Prices for Consumers Than Online Catalogs.
- Wand Y, Weber RY. 1992. On the Ontological Expressiveness of Information System Analysis and Design Grammars. *Journal of Information Systems*,
- Wand Y, Wang RY. 1996. Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM* 39(11): 86-95.
- Wang HH, Rock SM, Lee MJ. June-July 1996. OTTER: The Design and Development of an Intelligent Underwater Robot. *Autonomous Robots* 3(2-3): 297-320.
- Weil N. Feb 12, 2000. Web Publishers Hinge Their Future on XML. *InfoWorld* 22(7): 10.
- Williamson O. 1979. Transaction-Cost Economics: The Governance of Contractual Relations. *Journal of Law and Economics* 22(2): 233-261.
- Wooldridge M, Jennings N. 1995. Intelligent Agents: Theory and Practice. *Knowledge Engineering Review* 10(2): 115-152.

Robert J. Kauffman is Associate Professor of Information and Decision Sciences, Carlson School of Management, University of Minnesota. His degrees are from the University of Colorado, Cornell University and Carnegie Mellon University, and he has held faculty positions at New York University and the University of Rochester. His research interests center on IT strategy in financial services, economics and information systems, and electronic commerce. His articles appear in *Information Systems Research*, *MIS Quarterly*, *Communications of the ACM*, *Journal of Management Information Systems*, *Organization Science*, *IEEE Transactions on Software Engineering*, and *International Journal of Electronic Commerce*. He has twice co-chaired the annual *Workshop on IS and Economics (WISE)*, and is currently co-chairing sessions at the *Hawaii International Conference on Systems Science* on economics and electronic commerce, and information technology and organizational strategy. He is a current associate editor for the *Journal of Management Information Systems*, *International Journal of Electronic Commerce*, *Journal of Association of Information Systems*, and *Management Science*.

Salvatore T. March is the David K. Wilson Professor of Management at the Owen Graduate School of Management, Vanderbilt University. He received a BS in Industrial Engineering (1972) and MS and PhD degrees in Operations Research (1975, 1978) from Cornell University. His primary teaching responsibilities and research interests are in the areas of Information System Development, Logical and Physical Database Design, Distributed Database Design, and Object-Oriented Languages, Development Tools, and Methodologies. His research has appeared in journals such as *ACM Computing Surveys*, *ACM Transactions on Database Systems*, *Communications of the ACM*, *IEEE Transactions on Knowledge and Data Engineering*, *Information and Management*, the *Journal of MIS*, *Information Systems*, and *Information Systems Research*. He has served as the Editor-in-Chief for *ACM Computing Surveys* and as an Associate Editor for *MIS Quarterly*. He is currently an Associate Editor for *Decision Sciences Journal*, *Information Systems Research*, and *Information Systems Frontiers*. He has been involved in organizing and program committees for numerous conferences including the *Entity-Relationship Conference*, *ACM SIGMOD Conference*, *IEEE Data Engineering Conference*, the *Workshop on Information Technologies and Systems (WITS)*, and the *International Conference on Information Systems (ICIS)*. He served as a member of the Program Executive Committee for *ICIS '99*. He is currently a member of the Steering committee for *WITS*.

Charles A. Wood is a Ph. D. Candidate at the Carlson School of Management, University of Minnesota. His research interests center on the effects of electronic commerce (EC) technology on information asymmetry (IA), and how these changes in IA affect EC seller behavior and strategy. In addition, his research includes object-oriented development principles and system development. Some of his research appears in *Information Systems Frontiers*. He is the sole or lead author of several books dealing with database and Web development in a visual environment, including *OLE DB and ODBC Developer's Guide* (formerly entitled *Visual C++ 6 Database Developer's Guide*) and *Visual J++ 6 Secrets* from IDG publishing, *Visual J++* from Prima Publishing, and *Special Edition Using PowerBuilder* and *Special Edition Using Watcom SQL* (now Sybase SQL Anywhere) from QUE publishing. He has contributed to four books, also dealing with database and Web development, including five chapters dealing with Visual C++ database development in *Visual C++ 6 Unleashed* from SAMS publishing, two chapters dealing with computer aided software engineering (CASE) and object-oriented development using a relational database in *Client/Server Unleashed* from SAMS publishing, a chapter discussing object-oriented development in *Using Turbo C++ for Windows* from QUE publishing, and one chapter discussing database application development in *PowerBuilder 4* from Comdex Computer Publishing (India). He has been involved internationally in numerous conferences, discussing database-centric and Web-centric topics such as SQL optimization, Java development, and database design.