

The Dynamics of the UMDL Service Market Society

Edmund H. Durfee, Tracy Mullen, Sunju Park,
José M. Vidal, and Peter Weinstein*

AI Laboratory, EECS Department, University of Michigan
1101 Beal Avenue, Ann Arbor, MI 48109-2110
{durfee, mullen, boxenju, jmvidal, peterw}@umich.edu

Abstract. One of our goals when building the University of Michigan Digital Library (UMDL) has been to prototype an architecture that can continually reconfigure itself as users, contents, and services come and go. We have worked toward this goal by developing a multi-agent infrastructure with agents that buy and sell services from each other using our commerce and communication protocols. We refer to the services and protocols offered by this infrastructure as the Service Market Society (SMS). Within the SMS, agents are able to find, work with, and even try to outsmart each other, as each agent attempts to accomplish the tasks for which it was created. When we open the door to decentralized decision-making among self-interested agents, there is a risk that the system will degenerate into chaos. In this paper, we describe the protocols, services, and agent abilities embedded in the SMS infrastructure that combat such chaos while permitting flexibility, extensibility, and scalability of the system.

1 Introduction

A library serves a community of users by making available information content and services that are valued by that community. In a digital library, the information content and services are electronically available, and the user communities are no longer geographically defined. Realizing a digital library therefore includes difficulties in digitizing contents, computerizing services, and networking together users, services and contents. A tremendous amount of work has gone into all of these areas, and while many challenges remain, increasing numbers of digital libraries with interoperating components are appearing.

But even as these difficulties are being overcome, the result can well be an overwhelming tangle of possible information sources without the structure and selectivity that renders a library navigable. In other words, if the administration of a traditional library is challenging, the administration of a digital library can border on impossible due to the magnitude of content and services available, the rate of change in what is available, the size of a user population that is not bounded by physical proximity, and the evolving nature of that population.

* The authors are listed alphabetically. This work was supported, in part, by the NSF/DARPA/NASA Digital Library Initiative under grant CERA IRI-9411287.

The focus of this paper is on meeting this challenge. That is, in this paper we take for granted a rich network of interoperating information providers, consumers, and services. We also assume that it is impractical to marshal all of the information sources and services to address every information need for every user. Choices must be made, such that services are applied where they are expected to yield the most benefit. Traditionally, these choices are made by administrators who: identify, characterize, register, and track a user community; seek out and include the content that will benefit that community; and provide the most valuable services for tasks such as organizing, searching, abstracting, and disseminating the content.

Because we see the pace of library evolution increasing in the digital age, we instead embrace an alternative approach where we want to move as much of the administrative overhead into the digital infrastructure as possible. Rather than requiring frequent and costly intervention by human librarians, the library should evolve guided by the policies dictated to the infrastructure. The infrastructure should encourage:

- **Flexibility:** It should be able to embody a wide variety of policies to realize different flavors of information economies (public libraries, corporate libraries, university libraries, personal libraries, . . .)
- **Extensibility:** Providers and consumers of information goods and services should have incentives to join the information economy and be able to find their counterparts.
- **Scalability:** As the plethora of users, goods, and services grows, the underlying, computerized administration of the library should not bog down.
- **Robustness:** The infrastructure should support resistance against any agent manipulating the system to its own benefit and at the expense of other agents.

Toward this end, the University of Michigan Digital Library (UMDL) is structured as a collection of agents that can buy and sell services from each other using our commerce and communications infrastructure. To many, the concept of commerce in a library is heresy, and conjures images of library patrons having to pay as they go. Before continuing, therefore, let us briefly dispel this misunderstanding. Treating a library as an information economy provides us with a well-studied framework for making decentralized decisions about the allocation of the limited information goods and services that are available. Poor allocation decisions can lead to users receiving poor service, or none at all! Yet, users need not necessarily be aware of the underlying economic decisions being made, just as they are generally shielded from the decisions library administrators must make as they weigh the pros and cons of subscribing to a particular monograph series versus hiring another reference librarian. Thus, in many cases, users will be unaware of the underlying infrastructure, though if it is working well they will reap the benefits.

Indeed, our initial agent infrastructure forms the core of the UMDL system that has been deployed to several schools in southeastern Michigan, where it is used by hundreds of students daily. For the most part, the services used by the students have been selected and added to the deployed UMDL system by project members, and with only hundreds of students performing similar tasks, issues of flexibility, scale, extensibility, and robustness have not been stressed. This has allowed our research efforts to begin developing answers to these issues and experimenting with our answers before they have become critical to the wider deployment of the system. In this paper, we outline the definition and design of the infrastructure for the UMDL information economy, and the kinds of agents that exist in it, that allow decentralized (scalable) ongoing configuration of an extensible set of users and services. We refer to the services/protocols offered by this infrastructure as the Service Market Society (SMS).

The SMS requires the integration of numerous agent technologies for knowledge exchange, commerce, learning, and modeling. Some of our earlier publications [1] have already explained the general agent architecture of the UMDL, along with the agents' communication system. In this paper we concentrate on the *dynamics* of the system (i.e. how agents find and interact with each other) and on the abilities and resources that the agents need in order to effectively participate in the economy. We describe our prototype SMS in which a changing population of agents can find each other, enlist each other's aid (for a price), decide on the terms of an interaction, and learn to differentiate among providers. We use our prototype to demonstrate how these technologies contribute to providing a flexible, extensible, scalable, and robust digital library. The agents and services we describe, unless otherwise noted, are implemented within the UMDL, but to avoid disrupting the use of the deployed system the experimental SMS agents are ignored by agents that are supporting the UMDL educational mission. As the kinks are worked out of experimental agents, therefore, deploying them has been trivial: an agent stops describing itself to the UMDL as experimental, and thus is available to the deployed system. Recently, for example, instances of many of the agents described in this paper (including auction agents, the auction manager, and the price monitor) have become part of the UMDL system used by the schools.

We start with an overview of the SMS, and its component technologies: the UMDL ontology and the auctions. Next, we illustrate how these technologies support resource allocation decision-making with a simple example of balancing load, and support the identification of resource needs based on supply and demand. We then turn to issues of whether our infrastructure encourages extensibility by looking at the impact of adding profit-seeking agents that reason about other agents into the mix. Finally, we summarize our work that shows how learning agents add a measure of robustness to the UMDL SMS. As the reader will surmise, the SMS brings together several related threads of work, and this paper attempts to summarize these pieces and how they fit together. The work addresses only one (high-level) facet of the digital library enterprise, concentrating on describing, finding, allocating, and evaluating services, and is as yet

incomplete even for this aspect of digital libraries.² Yet, we hope to convince the reader of the promise of this approach for meeting some of the long-term challenges in managing large-scale, open, rapidly-evolving digital libraries.

1.1 Other Digital Library Infrastructures

There is a tremendous amount of work going on on many facets of digital libraries; entire conferences and journals are devoted to the topic, and surveying all of the excellent work in the field is beyond the scope of this paper. Here, we will only mention a few projects that directly address aspects of digital library infrastructures. The InfoBus [9] system extends the current Internet protocols with a suite of higher-level information management protocols. This work concentrates on the details of the communication protocols which are, in fact, quite similar to those used in the UMDL³. They have even implemented some commerce interfaces [4]. However, in contrast to the work in this paper, they do not address the higher-level problem of agents finding each other (i.e. finding the services they want) and effectively taking part in the economy. The Alexandria Project [2] also addresses the issue of communication between users, collections, and mediators, but it does not use an economic framework for making resource allocation decisions.

2 THE SERVICE MARKET SOCIETY

The UMDL SMS implements a market-based multi-agent system where agents buy and sell services from each other. Agents can be added to or removed from the system at any time. Instead of having to rely solely on internally designed agents, the UMDL SMS is designed to attract outside agents to provide new services. These agents in turn are motivated by the long-term profit they might accrue by participating in the system.⁴

The raw resources of a digital library consist of collections of information, where a collection might be a database of journal articles or it might be a collection of related web pages. Collection Interface Agents (CIAs) provide access and search services for these collections. Various middlemen, or *mediator* agents

² For example, the current SMS does not address issues of privacy, security, and even payment. In the description of “buying and selling” in SMS that follows, our current simple model assumes that buyers and sellers are each keeping track of their “balances” and honestly incrementing (decrementing) as services are sold (bought). We have attempted to make our system compatible with the various payment mechanisms being developed elsewhere.

³ Both projects use the ILU CORBA implementation. Some degree of interoperability between the two projects has been achieved.

⁴ While designed to encourage third-party agent/service development, to date the third-party services in the UMDL have been introduced by the project team. Bootstrapping the process to the point where third-parties have sufficient incentive to join is, not surprisingly, hard!

(such as QPAs, see below), transform the raw information resources into finished products or services which the end-user desires. Each user, or library patron, has a User Interface Agent (UIA) which interacts with the UMDL on his or her behalf to acquire these services. Finally, within the system are *facilitator* agents (such as the Registry, SCA, auctions, etc. as described below) which facilitate the process of agents classifying, locating, and connecting agents to provide more comprehensive services.

Figure 1 shows a simple scenario which we will be using throughout this paper. In this scenario, users want to find sources of information for various topics (e.g. history, science or mathematics) and various audience levels (e.g. middle school, high school, or professional). Query Planning Agents (QPAs) [10], act as middlemen, accepting queries from users and returning collections related to those queries. Initially, the user sends a query, via a web-based Java interface, to the UIA. The UIA must then find an agent that can service this query. In order to allow agents to find the services they need, we have implemented an ontology of services. Every agent must be able to describe the services it wishes to buy or sell using terms from the UMDL ontology.

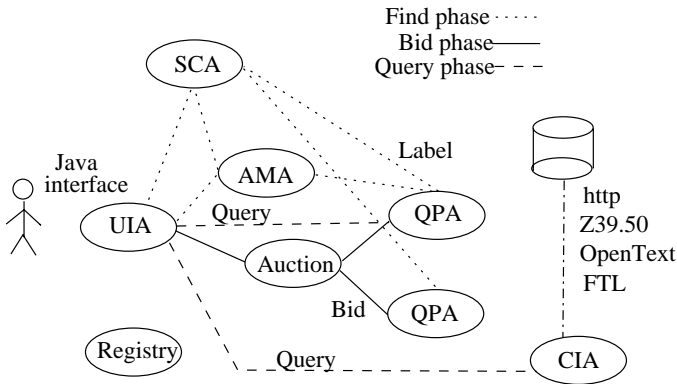


Fig. 1. The UMDL Service Market Society. The User Interface Agent (UIA) describes its needs to the Service Classifier Agent (SCA) and is told the appropriate service label. The UIA asks the Auction Manager Agent (AMA) for an auction where the service is sold, and goes to that auction. The Auction matches the UIA with an agent providing the service (in this case, a Query Planning Agent (QPA), which returns the name of a Collection Interface Agent (CIA)). The SMS supports any number of UIA, CIA, QPA, and Auction agents.

In order to join the UMDL, every agent must register its unique agent-id with the Registry. An agent can then advertise by sending⁵ its service description to

⁵ Agent communications use KQML primitives implemented using CORBA objects,

the Service Classifier Agent, as shown by the dotted lines, and receiving a service-label. The SCA automatically classifies the advertised service descriptions into a subsumption-based taxonomy. This organization enables the SCA to match requests for services to “semantically close” descriptions, to recommend the most appropriate services out of those that are currently available. We encourage agents to describe their needs with as much detail as possible. The SCA can then recommend the available services with the least general subsuming descriptions.

Once an agent has acquired the appropriate service-label from the SCA, and is ready to buy/sell that service, it sends the desired label to the Auction Manager Agent (AMA) (as shown by dotted lines) which returns a list of auctions which sell matching services. The agent then sends buy or sell bids to a particular Auction agent (solid lines). UMDL *auctions* operate by collecting offers from participating agents and determining agreements consistent with those offers, matching individual buyers and sellers.

In our scenario, UIAs want to buy query planning services while QPAs want to sell this service. Once a match is found for the UIA, it will send its query to the matching QPA. The QPA returns the appropriate CIAs and the UIA then forwards its query to them. The CIAs can translate UMDL queries to a variety of protocols (e.g. http, Z39.50, FTL, etc.) and return the appropriate documents.

The agents in our system are free to return services of differing qualities and to disagree on the exact quality of a given service. We, therefore, expect that some agents might try to take advantage of other agents in the system by manipulating either their buy/sell bids, or the quality (and, therefore, the production cost) of the service they provide. We call such agents *strategic* agents. These strategic agents will need to be able to assess the quality and value of a service received, or to determine the exact price to charge. They can make these determinations by either using knowledge about the expected arrival of buyers/sellers, or by learning models of the other agents. In this paper, we investigate how the addition of these strategic agents changes the dynamics of the UMDL SMS. We find that strategic agents increase the robustness of the system. We also find that the SMS market works to minimize the amount of strategic thinking that the agents can gainfully engage in, but does not completely eliminate the need for having some agents with strategic abilities, even if these abilities are not always used.

3 THE UMDL ONTOLOGY

We use ontologies to encode declarative descriptions of complex agent services. Declarative descriptions are required to establish a space of services independent of the implementation of the current set of agents.

Representing complexity in a digital library, and any other “real-world” domain, demands expressiveness that is substantially more powerful than that of relational databases. Services have many descriptive dimensions (attributes),

see [1] for details.

may be partially described at any level of granularity (with any combination of dimensions), and may be viewed from many perspectives (accessed by different sequences of attribute values). For example, an agent might seek a service to recommend a collection of articles on volcanoes, for high-school audiences, to be contracted for via auction. Alternatively, the agent might seek an auction that sells a service to recommend collections. (In Figure 2, follow links either from “recommend-dlcollection” at the top, or “auction” at the bottom of the figure). In an ontology, both perspectives can be represented simultaneously, and retrieval supports queries on any level of granularity. Declarative formalisms with less expressiveness than ontologies, such as relational databases, force a commitment to particular levels of granularity and particular perspectives.

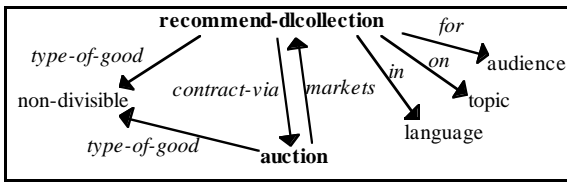


Fig. 2. Multi-dimensional, multi-perspective complexity

To promote reuse, the UMDL ontology is divided into nested modules (each of which is itself called an ontology) [12]. The most general includes library content and services that we consider to be part of a “generic” digital library. The second module adds concepts specific to the UMDL implementation, such as auctions. The third module describes agent services. We call this last ontology “dynamic” because agents define new service concepts at runtime. In contrast, “static” ontologies are either fixed, or are changed slowly over time by committees of persons.

3.1 The Service Classifier Agent

In the current SMS, UIAs, QPAs and Auction agents use the same SCA, and thus subscribe to the same ontologies. Figure 3 illustrates these agents’ interactions with the SCA. Agents are in ovals, ontologies in rectangles, and the arrows connecting agents represent messages, paraphrased in natural language. These messages are expressed in Loom, a description logic in the KL-ONE family [5]. To communicate with the SCA, agents use terminology from the nested ontologies. In the figure, the thin line around the SCA is jagged to show that the set of available terms is dynamic; both QPAs and the AMA add concepts to the agent services ontology. The messages are shown with font styles that correspond to the ontology labels (plain, italics, and upper-case), to highlight the source of their terminology.

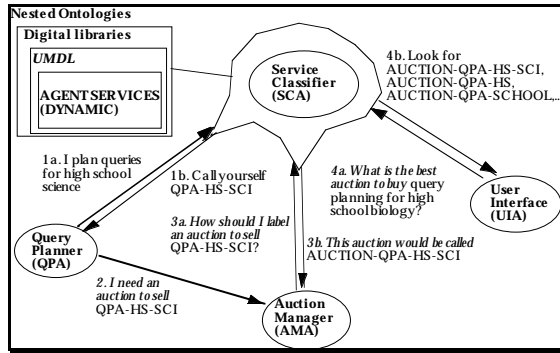


Fig. 3. Agent interaction with the SCA in the SMS

Figure 3 illustrates the ability of the SCA to construct new terminology from existing terms at runtime.⁶ Concepts in the SCA’s dynamic ontology hide knowledge, just as words chunk meaning in natural language. Auctions sell some service, but they don’t need to know anything about that service. The AMA asks the SCA for a service label for an auction, using the QPA’s service label, but it does not know anything about that service. The SCA classifies the auction service using characteristics inferred from the QPA’s service label. Thus, the SCA can respond to the UIA’s request for an auction, which is phrased in terms from the static ontologies, rather than phrased in terms of the label which the AMA used to define the auction service. This appropriate hiding of knowledge reduces overall system complexity, and increases reusability and maintainability.

The most dramatic contribution of the SCA to the SMS derives from its declarative description of services, and its ability to rank available services given a target and search strategy. For example, when a new QPA provides a service that better meets a UIA’s needs, this fact is captured by the service description. The UIAs can then automatically switch to buying services from the new QPA.

4 MARKET-BASED NEGOTIATIONS– AUCTIONS

Since UMDL cannot know at design-time what services will be available in the future or what the best negotiation mechanisms are for any given situation, its languages and protocols have been designed for flexibility and extensibility. In the

⁶ Note that by “new terminology” we mean new, meaningful combinations of the core terms in the ontology. There are a huge number of ways that the core terms in the ontology can be composed into descriptions, only a small number of which will be meaningful. New terms are those that are dynamically created because they are useful. If the primitive terms in the nested ontologies cannot express a concept in any combination, then the static ontologies must be extended; we assume that at this level the growth of the ontologies requires human intervention.

case of negotiation mechanisms, UMDL uses a generalized auction specification to allow goods and services to be offered under a wide variety of terms [6].

While generalized auctions are not the only possible kinds of negotiation mechanisms, they offer many advantages. They provide a structured, yet flexible market infrastructure promoting automated negotiation due to the following characteristics:

1. **Mediated** (vs. Unmediated): Buyers do not have to separately find and contact every seller; a useful property in a large-scale and dynamic environment. Information about current status can be easily and uniformly disseminated.
2. **Price** (vs. Barter): Under certain circumstances, price minimizes communication between agents.
3. **Formal** (vs. Informal): Standardized offers simplify communication between agents.

Auction specifications consist of parameters which can be tuned to reflect the type of good being sold, timing requirements, or mechanism properties desired. For example, a query planning service might be sold differently depending on how it is bundled (e.g., per-query, subscription), its characteristics (topic, audience, timeliness), its terms (redistribute, read-only), or to whom it is sold (individual, library, group). Different kinds of auctions will have advantages and disadvantages for different types of information goods and services.

Although evaluating tradeoffs between desirable properties of different auction types is important, the role of the UMDL is not to require that a particular auction be used in a given circumstance, but rather to provide an open framework whereby the use of these market facilitator agents is supported. To automate the creation and management of auctions as much as possible, we use the Auction Manager Agent (AMA).

4.1 The Auction Manager Agent

The purpose of the AMA is both to generate and track auctions as well as to serve as a tool for exploring alternative market configurations. Currently the AMA is used to automate the process of creating a well-specified market. Agents request auctions for the services they sell/buy from the AMA. Using the SCA, the AMA determines which auctions sell these services. If one or more such auctions exist, then the agent is notified of their agent-ids. Otherwise, the AMA creates an auction and then informs the requesting agent of its location. Since the AMA uses the SCA to generalize service descriptions, the description sent by the agent does not have to exactly match the auction's service description.

5 SIMPLE MARKET SCENARIO— PRICE TAKERS

In this section, we discuss two simple scenarios implemented in UMDL and the design process and mechanisms used. The scenarios use the same agents

and services as shown in Figure 1. We make the assumption that the agents have been designed within UMDL to fulfill certain system-level objectives. In particular, we assume the QPAs have been designed to behave competitively. Competitive agents take prices as given—ignoring any market power they might possess, thereby eliminating the need to reason strategically about other agents’ bidding behavior and simplifying the QPA design.

5.1 Load Balancing

This scenario shows how, under certain conditions, a price mechanism can be used to distribute service load between multiple providers of the same service. Specifically, consider the case where we want to balance the service load between several high-school science QPAs running on separate machines. Each QPA provides the same service, in this case, *exactly* the same service since each one is just a spawned copy of the same code. We simplified this scenario in a number of ways, assuming that all sites provide free access, and all queries are the same.

In designing the QPAs, we assume they are competitive, and bid their marginal costs, namely what it would cost them to provide another unit of their product. This pricing policy directly reflects the cost to the system of using the input resources to provide this service. QPAs use computational and network resources to produce query planning service. Since these resources get congested as the load on a machine or network gets heavier, the marginal cost of adding another query will increase with the number of queries currently being processed. We modeled this technology in the QPAs using a quadratic cost function:

$$\begin{aligned}\text{Cost}(\text{query}) &= A * \text{load}^2 + B * \text{load} \\ \text{Marginal Cost}(\text{query}) &= 2 * A * \text{load} + B\end{aligned}$$

Each additional query the QPA processes is priced at its marginal cost. Note that this cost function does not represent any real computer load model, but it does capture the basic notion of using a congested resource. The parameters A and B can be set to reflect relative differences between QPA technologies. For example, a QPA with $A, B = 1$ could be thought of as running on a faster machine than a QPA with $A, B = 2$.

All sellers (QPAs) make offers based on their current marginal cost. Buyers (UIAs) are assumed to want to purchase the query planning service immediately upon bidding. Query planning services are traded in an auction which matches the current lowest price seller with an incoming buyer, as long as the buyer’s bid is above that price. Note that because of the way the agents have been designed, we know that this auction is incentive compatible for buyers, i.e., their best bidding strategy is to bid their true value for query planning services. This is because the QPAs, who would normally have an incentive to try and strategize about bid amounts, have been engineered to be competitive⁷. Once

⁷ If the QPAs had not been engineered, a better solution would have been to use a second price auction.

a transaction occurs between a given seller and buyer, both offers are removed from the standing offer list. The QPA recomputes its marginal cost based on having an additional query to process and submits a new, higher, sell offer to the auction.

To see how this results in query load balancing, let's consider what happens where there are two QPAs, (QPA-1 and QPA-2). Given the same initial query load, the sell offers for each agent will be the same, and the first buyer will be matched up arbitrarily with one of them, say QPA-1. At this point, the query load for QPA-1 will rise, causing its marginal cost to rise, and it will submit a new offer at a higher price than before. The next incoming buyer will be matched with the current lowest price seller, QPA-2. QPA-2 now makes its new higher sell offer to reflect its increased load. Assuming that QPA-1 and QPA-2's previous queries are still being processed, both agents are again offering query planning service at the same, although higher, price. If one of QPA-1's queries finishes, then it sends in a new lower offer, and will be matched up to the next incoming buyer. Given a steady load of user queries, this system appears to settle down to an equilibrium price, although we have not done any formal analysis of these results. Notice that the dynamic addition or deletion of an agent does not affect the long-term running of the load balancing mechanism. For example, if any one of the agents were to die, it would only affect the queries that they were processing at that time. Future query planning requests would be matched with the remaining QPAs still participating in the auction. Similarly, spawning a new QPA means that it can start making offers to the query planning auction immediately.

Similar situations where this mechanism would be useful are in the provision of basic library services, which outside agents may not be interested in supplying. By making the agents competitive, UMDL can assure that the system costs are accounted for, that users get a low price, and agent designs are kept simple. Another example is distributing the access load to collections for which the library has a site license. Even though the site license may allow unlimited access, there are still associated network and compute costs, and for popular collections we may want to create mirror sites and distribute the load across them. A consistently high price of access may indicate when a given collection should be mirrored.

The point in the preceding is that the SMS economic foundation provides a flexible means for making decentralized resource allocation decisions. When we consider specifically the problem of load balancing, there are clearly many other load-balancing algorithms that have been developed in fields such as distributed computing and operations research. For many problems, such as when the available resources are fixed, the task needs well-defined, the performance criteria are globally determined, and the control centralized, alternative algorithms can provide optimal or near-optimal (within well-defined bounds) allocations. What we have illustrated in our work is that the economic principles of supply and demand provides an alternative load balancing mechanism which appears to be particularly useful in domains where resources come and go, tasks may have un-

certain resource requirements, the participants can have different performance criteria (the costs and values they ascribe to activities and outcomes can differ), and the control is decentralized. More work remains in determining exactly the circumstances in which an approach like ours will outperform more traditional “command and control” methods.

5.2 Price Monitor Agent

In a large-scale, dynamic system, information about the current status of the system can serve as feedback to different kinds of control mechanisms. In this section, we describe a simple monitor agent, called a Price Monitor Agent (PMA), which collects information about an auction’s prices over time and uses that information to decide whether to spawn more service providers.

In our scenario, the PMA monitors the price of query planning services. The price in this auction reflects the load on the QPAs— a consistently high price means that the QPAs are heavily loaded. One way to reduce the load is to replicate QPAs (assuming that there are other less heavily loaded machines on the network). When the load is reduced, the price should come down.

The PMA is initially set with a given minimum and maximum price bounds, although they can be changed later. When the price exceeds the upper bound, i.e., the load on each QPA is getting high, it spawns additional QPAs. This effectively distributes the load from future queries.⁸ When the price goes below the lower bound, i.e., the QPAs load has fallen, the PMA has a choice of either removing or just inactivating one of the extra QPA, depending on whether it expects the load to increase again in the near term. The effect of this behavior is to keep the load across QPAs (and therefore the response time to users) within specified bounds even though the user demand is dynamic.

Figure 4 shows an overview of how the PMA operates. Below is a description of what is occurring at each step:

1. A heavily loaded QPA sells its services for \$0.13.
2. The PMA checks the current price at the Auction.
3. Since the price is above it’s upper bound (\$0.10), the PMA spawns another QPA.
4. The new QPA sells its services for \$0.08.
5. The user is matched with the lowest priced (least loaded) QPA.

⁸ In fact, there is nothing to prevent a QPA from itself attempting to find another QPA to service a query. That is, an overloaded QPA, which has been awarded a query because it is the least overloaded of all the QPAs, might periodically check the auction and attempt to pass off one or more of its assigned queries. Thus, new QPAs can in principle reduce the current load, although in our current implementation the QPAs do not do this.

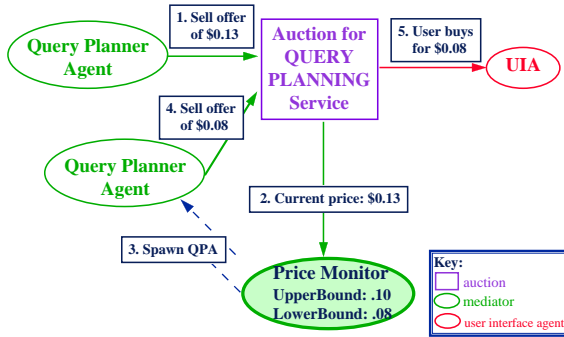


Fig. 4. Price Monitoring Scenario

6. (not shown) When/if the price goes below \$0.08, the PMA inactivates one or more of its spawned QPAs.

In our experiments, the PMA, given reasonable bounds for the prices and activity of the agents, was successfully able to keep the price within the given price bounds. However when the price bounds were too tight or the activity on the system was too dynamic, the PMA would oscillate. For example, if too many agents get added, the price may fall too low, causing too many agents to get removed, which causes the price to go too high, and so forth. Providing the monitoring agent with knowledge about usage patterns, or having it collect statistics about usage, is one way to address this problem.

Finally, let us briefly address the question about where to set the upper price bound. Since price reflects load, which in turn reflects response time, we might determine the maximum desired response time and from that compute the maximum price. However, if this is done for all potentially replicable services, then we risk overtaxing the available computational and network resources with too many copies of too many agents. Ultimately, any policy for replication needs to determine whether the resources that will be devoted to a new agent could be more usefully applied elsewhere. While right now the PMA simply uses user-specified bounds (and indeed, only replicates QPAs so that there isn't any competition between requests for spawning different agent types), ultimately the resources available for replicated agents should themselves be auctioned off, such that PMAs for different classes of agents will compete for the resources and those resources will be allocated based on the same economic principles as other UMDL services.

5.3 Conclusions from simple market scenarios

We showed how UMDL agents which use system resources to provide query planning services to library users can be designed to distribute system load efficiently and to respond to dynamic changes in the number and frequency of users' requests. A less obvious benefit to designing the seller agents to be competitive is that the software design is straightforward and modular and requires no information about the behavior of the other agents in the system.

Also, the task of choosing an auction mechanism is vastly simplified when dealing with competitive sellers, especially for issues such as how to achieve incentive compatibility (having bidders bid their true evaluation, so that resources can be allocated efficiently) or deciding what kinds of information the auction should release (clearing price, bid quotes, bids) and how that affects agent behavior and system properties. In this case, sellers are designed to bid what it costs the system to provide the service. The auction used was a type of double auction, where each buyer's bid is cleared immediately and is priced at the lowest seller's price. Thus, library users can bid their true value for the service and be assured that, as long as their value for the service is greater than the cost to the system to provide it, they will acquire the service at system cost. Neither buyer nor seller agents need any information when bidding about other agents or even, in this simplified scenario, auction prices.

Additionally, if library users *were* to bid strategically, i.e., not their true value for the service, all that would happen is that they might bid too low and not acquire a service they would be willing to pay for, or they might bid too high and acquire one for more than they value it, both inefficient system outcomes. Of course, there are still strategic opportunities for buyers. For example, they might try to strategically time when to make a bid, e.g., if prices are lower at midnight than at 9am, then a user's agent might want to store up non-rush queries to be processed later. In this scenario, we assumed that these were live users, actively waiting for queries to be processed and, additionally, that the price for query processing was small enough not to make it worthwhile to try and exploit small jumps in prices. The Price Monitor was developed to support this immediate processing of on-line queries by using the information that price conveys about system costs to smooth the load across the system at a particular time.

This section has focused on the case where all services are offered by the library itself. We showed a mechanism that provides for the efficient allocation of library services between users. If the services are instead commercial services offered by third parties, then strategic behavior on the parts of the buyers and sellers and careful consideration of the properties of the auction mechanism will become important.

6 STRATEGIC AGENTS

A real economic system provides monetary incentives to its participants and dynamically allocates the available system resources in part because the human

beings and corporations that take part in it are smart. They can recognize when they are charging too little or too much, or when they are not getting the quality they expected. They also do not spend all their time thinking about the economy, but only do as much strategic thinking as is needed.

If we want our agent economy to be as robust as the real economy, we will need to have at least marginally intelligent agents. These agents will need to know what to bid— both when price has reached an equilibrium (which is easy), and when the price is fluctuating. For example, if an agent has knowledge about an expected increase in the number of buyers/sellers, it should be able to use this knowledge to its advantage. Also, if an agent is the only seller of a service then it should be able to take advantage of its monopoly, while if the buyers find that a seller’s prices are too high for the service it sells, they should be able to avoid buying from him. Agents should, in effect, be strategic. One drawback of this, however, is the possibility that agents will spend all their time thinking strategically rather than carrying out their domain tasks. Fortunately, as we will show, our economy discourages agents from engaging in ever-increasing amounts of strategic thinking.

6.1 P-STRATEGY AGENTS

In this section, we relax the assumption that agents behave competitively, and investigate how strategic agents affect the UMDL in terms of market and allocation efficiency.

A Strategy based on stochastic modeling (p-strategy) We have developed an agent’s bidding strategy based on stochastic modeling (called p-strategy) for the UMDL SMS, the details of which are given in [7]. The main idea behind the p-strategy is to capture the factors which influence the expected utility for the agent, using Markov chains. For instance, a seller is likely to raise its offer price when there are many buyers or when it expects more buyers to come. The number of buyers and sellers at the auction, the arrival rates of future buyers and sellers, and the distribution of buy and sell prices are among the identified factors. The p-strategy is able to take those factors into account in its stochastic model.

In our previous research [7], we have shown that an agent possessing the p-strategy has an advantage over agents that do not possess it when they compete in multi-agent auctions. Given that the p-strategy is effective in the UMDL auction, nothing prohibits any self-interested agent from adopting the p-strategy. We expect many p-strategy agents to coexist in the UMDL, and thus are interested in the collective behavior of such agents. In what follows, we briefly summarize our observations which are more completely detailed elsewhere [8].

Experimental results In our experiments, the p-strategy agents have models of how the auctions in the SMS evolve as buyers and sellers enter and are (sometimes) matched. Thus, these experiments should be considered illustrative

Session	Description	Bidding strategy						
		Seller1	Seller2	Seller3	Seller4	Seller5	Seller6	Seller7
(1)	All competitive	C	C	C	C	C	C	C
(2)	1 p-strategy	C	C	C	C	C	C	P
(3)	2 p-strategy	C	C	C	C	C	P	P
(4)	3 p-strategy	C	C	C	C	P	P	P
(5)	5 p-strategy	C	C	P	P	P	P	P
(6)	All p-strategy	P	P	P	P	P	P	P

Fig. 5. Experiment setting. C is Competitive. P is for p-strategy agents.

of performance for the current SMS economic framework; under different assumptions about, for example, auction parameterizations, different observations might hold. But for the SMS, we are interested in the effects of p-strategy agents on the efficiency. We measure the efficiency of the system in two ways. First, we measure the efficiency of allocation, by comparing a p-strategy agent's absolute and relative performance. Second, we measure the efficiency of the market, using the total profit generated.

Figure 5 shows the six experimental settings with 7 buyers and 7 sellers. The buyers bid their true valuations, while the sellers bid their sell prices depending on their strategies. In Session 1, all seven sellers are type C (Competitive) and bid their true costs. From Session 2 through Session 6, we introduce more type P (p-strategy) agents into the auction.

A p-strategy agent has an upper hand over other-strategy agents, but this may not hold in the presence of other p-strategy agents. To test this we compare the profits of Seller 7 (p-strategy agent) across Sessions 2 to 6. As shown in Figure 6, the marginal profit of the p-strategy (smart) agent decreases as the number of p-strategy agents increases.

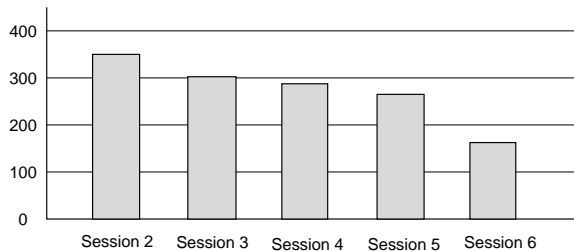


Fig. 6. The profit of the p-strategy agent (seller 7)

By replacing Seller 1 with a fixed-markup agent (who bids its cost plus some fixed markup), we also measure the relative performance of the fixed-markup agent (Seller 1) and the p-strategy agent (Seller 7). In Figure 7, we find that the simpler strategy agent (fixed markup agent) generally gets less profit than the p-strategy agent, but the difference decreases with the increase of p-strategy agents. That is, the disadvantage of being less smart decreases as the number of smart agents increases.

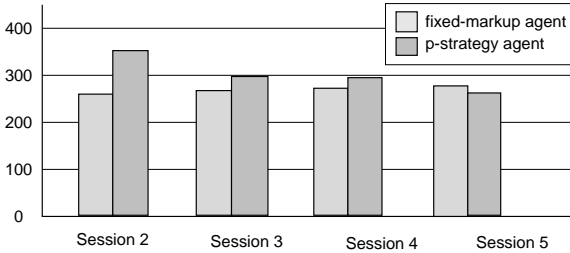


Fig. 7. The profit comparison between the fixed-markup agent and the p-strategy agent.

This result indicates that an agent may want to switch between using p-strategy and using a simpler strategy depending on what the other agents are doing. By dynamically switching to a simpler strategy, an agent can achieve a similar profit (to that of using the p-strategy) while exerting less effort on computing bids.⁹

We measure the market efficiency using the total profit generated from the buyers and sellers (see Figure 8). The total profit eventually decreases with more p-strategy agents, as the market becomes inefficient due to strategic misrepresentation of p-strategy agents (and therefore missed opportunities of matches). Note that the total profit does not decrease as sharply as one might expect (it in fact increases slightly up to Session 4) due to the efficiency of the UMDL auction mechanism.

We conjecture that having strategic sellers poses interesting tradeoffs between strategic inefficiency and surplus extraction. By misrepresenting their true costs, the p-strategy agents miss out on possible transactions. By anticipating the future arrival of buyers, on the other hand, they are able to seize more surplus.

⁹ While the chances are that the computational effort in computing bids is small compared to the costs of providing the service, it is still important to consider this cost. Specifically, because the UMDL auctions match buyers and sellers as bids are received, higher-cost bidding strategies can delay the submission of bids such that an agent consistently arrives at the auction “too late.” The overhead is worthwhile, however, if the agent makes up for missed opportunities by extracting more profit in

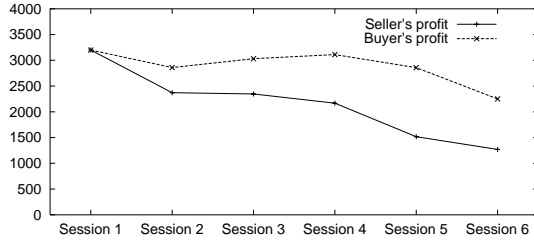


Fig. 8. The efficiency measured by the total profit.

Conclusions from experiments with strategic agents Although a self-interested agent in the UMDL has the capability of strategic reasoning, our experiments show that strategic thinking is not always beneficial. As previously shown, the advantage of being smart decreases with the arrival of equally smart agents.

We expect the UMDL is likely to evolve to a point where some agents are being strategic some of the time. An agent may want to switch between being strategic and not depending on the behavior of other agents: if enough other agents behave strategically, and agent can achieve additional profit even if it stays non-strategic.

Finally, the market efficiency of the UMDL will not decrease as sharply as one might expect. The profit-seeking behavior of self-interested agents will keep the UMDL agent population mixed with some strategic and some non-strategic agents. Thus, even though the market efficiency decreases with the increase in the number of strategic agents, the UMDL will not suffer the market inefficiency in the worst case.

6.2 LEARNING AGENTS

In this section, we use the scenario from Figure 1, but now we give sellers the freedom to return a service of any quality. The only thing that will prevent them from doing this is that we also give the buyers the ability to learn, which they use for determining which seller to buy from and at what price. To increase the information available for learning, we also allow the auctions to reveal all information (i.e. bids) they have to all interested agents. The agents are free to ignore any of this added information. By learning from this information, agents can avoid miscreant agents, and the performance of the information economy thus will not be compromised (or at least not for long) by such agents.

Learning in a market system While the UMDL ontology provides a way for agents to characterize the services they sell, there is no guarantee that all the

the cases where it does succeed in making a match.

goods sold at an auction for service x are indeed instances of service x . Agents could intentionally misrepresent their services. More frequently, agents might not entirely agree on precisely what constitutes service x . Indeed, subjective preferences might mean that some agents are quite satisfied with service x from a particular provider, while other agents are dissatisfied with that same provider's service. For example, while all agents might agree that QPA1 does sell service x , one agent might think that QPA1's service is faster, better, or more thorough than the same service x as provided by QPA2. This being the agent's subjective opinion, it is unlikely to be in agreement with all the other agents, but the agent might still be willing to pay more for service x from QPA1 than from QPA2.

Learning provides a means for agents to discriminate between services (or providers of services) when they have found sufficient grounds to make the discrimination useful. By learning, an agent can avoid being disappointed in its interactions by learning which agents to not interact with. If agents can learn from others' interactions (by observing others' experiences or sharing their experiences), they can as a society quickly ostracize rogue agents. Indeed, some agents might provide "recommendation services" by sampling and rating agents, and sharing (perhaps for a price) what they have learned with other agents. When agents learn, therefore, they increase the robustness of the system by partitioning away (what they see as) faulty agents.

In essence, learning provides a way for agents to develop expectations about others, and exploit these expectations to their mutual benefit. We can consider this as a rudimentary form of *trust* among agents. For a market-system to work well, an agent needs to be able to trust that its partners' view of good x is the same as its view of x . Similarly, an agent that uses recommender agents needs to trust their recommendations. This trust can be acquired by repeated iterations with the agents in question. Once the trust is acquired the learning is no longer needed, that is, until the trust is broken. This is why we argue that agents need the *ability* to learn, even if this ability is not always exercised.

Lastly, we propose that learning agents are not only useful, they are inevitable. In a society of selfish agents, we can expect that the designers will use every technology available to enhance the profits of their agents. Learning is one such technique. By implementing learning agents ourselves, we can determine how much of an advantage they will have and how they will affect the system.

Experimental Results To demonstrate the viability of the SMS with learning agents and under real-world heavy usage conditions, we ran several tests on the scenario shown in Figure 1. We implemented UIAs that periodically (every 16 seconds) buy a query from some QPA using the protocol described earlier. We also used one Auction, AMA, and SCA, along with several UIAs and QPAs. All the agents were deployed in machines all over our network. The UIAs kept track of how long it took for the QPA's reply to arrive and used this value in their learning. In general, the UIAs preferred fast and cheap service, and they were willing to pay more for faster service. A QPA's only preference was to increase its immediate profit, i.e. place its bid in order to maximize its expected profit

(remember that failing to get a sale means the QPA gets zero profit).

We gave the agents different learning abilities (see [11]). 0-level agents used reinforcement learning on the prices/values received. 1-level agents actually tried to model the other agents as 0-level agents. That is, they remembered what other agents had bid in the past and made probabilistic predictions based on the assumption that they will behave in the future as they behaved in the recent past. The 1-level agents then took actions based on these predictions.

0-level agents For our first test we used 0-level UIAs and QPAs. They all began with no knowledge about what prices to bid or accept. UIAs quickly learned that they can expect to get higher value if they pick lower prices, while QPAs learned what price they can charge that will maximize their expected profit. As predicted by economic theory, this price was their marginal cost, i.e. the lowest price they can charge without losing money. This equilibrium was reached even while all the agents acted purely selfishly.

However, this equilibrium is not completely stable. Network and machine delays, along with the agents' occasional explorative actions¹⁰, add noise to the system preventing the price from staying fixed at the marginal cost.

Even with only 0-level agents (i.e. no 1-level agents) we can see how the system behaves in a robust manner. Figure 9 shows the clearing price for an auction which starts with only one seller QPA. This QPA quickly "realizes" that it has a monopoly and starts to raise its prices. A second QPA is then added (as marked by the first vertical line), and we can see how its addition makes the price drop, but eventually it gets overloaded and the price rises again. A third QPA is added (at the second vertical line), affecting the price, and more QPAs are added at regular intervals (one at each vertical line).¹¹ The equilibrium price gets fairly close to the QPAs' marginal cost of 0. Towards the end the QPAs start to leave the system so the price starts to rise accordingly. This type of experiment gives us confidence that the agents will behave in a reasonable way even under boundary conditions, e.g. when a lot of the agents die, or when new services are added. If the agents were to determine their bid prices based on a fixed utility function, we might expect to find periodic or chaotic behavior [3], which we wish to avoid.

1-level Agents 1-level QPAs take advantage of price fluctuations by keeping models of the QPAs and UIAs and using these to make better predictions as to what they should bid. The 1-level models, while computationally expensive, allow QPAs to track the individual agents more closely, thereby identifying when a UIA is willing to pay more than the going rate. Previous research has shown that the advantages of 1-level models can be correlated to the price volatility (see [11]).

¹⁰ The agents were set to take a random action with probability of .05. This keeps them from converging on a local maxima.

¹¹ Note that these QPAs are *not* added because of some price bounds as with the Price Monitor Agent (Section 5 but rather are simply added at fixed intervals.

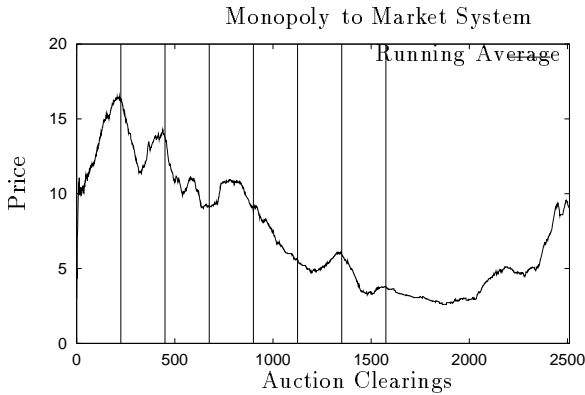


Fig. 9. Clearing price for successive auctions. Initially there is only one seller QPA, and an additional one is added at each of the vertical lines. The QPAs' marginal cost is zero.

However, this strategic thinking is only successful against 0-level UIAs. When we tested the 1-level QPAs against the 1-level UIAs, as seen in Figure 10, we found the QPA's performance on par with other similar 0-level QPAs. In fact, the sellers that made more were the ones that could answer the query faster, not the 1-level seller. In other experiments we also found that 1-level sellers' extra profit is reduced as other 0-level sellers become 1-level. In both cases, the 1-level sellers incentive to be 1-level (instead of 0-level) disappears with increased competition.

Conclusions from Learning Experiments The results on learning deeper agent models show that the UMDL SMS benefits from the existence of agents that have the *ability* to keep deeper models and look out for their best interests, even if they *do not always use this ability*. Agents are encouraged to model others because this can bring them higher profit. However, there is a computational cost associated with modeling, and a decreasing return for the agent as other agents also start to build models. This means that that the SMS will likely evolve to a point where some agents build models, some of the time, in the same manner as we expect some agents to use p-strategy some of the time. This is a great scenario because it gives us a very robust system (i.e. one that can not be sabotaged by deviant agents), while using few resources for this purpose and distributing the resources it uses among the agents.

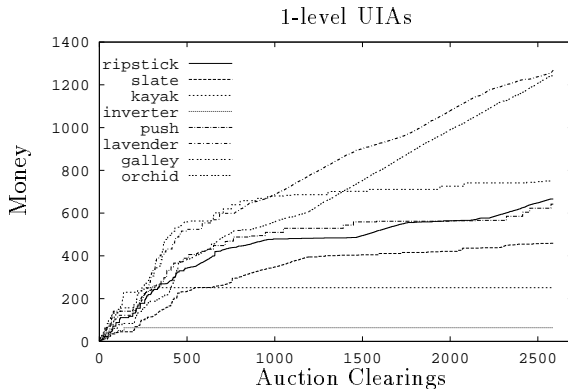


Fig. 10. Total revenue accrued by seller QPAs. Ripstick is the 1-level seller, the rest are 0-level. As time goes on the faster sellers accrue more wealth, not the 1-level seller. Because the different machines have different loads (other users), over time some of the machines are avoided because they respond too slowly, and their accumulated profits level off.

7 SUMMARY

We have given a condensed overview of the UMDL SMS and the agents, protocols, and languages that it encompasses. Through this description, we hope that we have conveyed how our SMS infrastructure supports our desiderata:

- **Flexibility:** In our SMS work, we have already explored several policies for designing an information economy, ranging from a more “public” flavor where agents that are assumed truthful and interested in providing services for the lowest possible price, to a more commercial flavor where agents might misrepresent themselves and will try to maximize their individual profits.
- **Extensibility:** We have described how agents with new services (as described by composing new descriptions in the shared ontology) can join the UMDL, and why they would have incentive to join.
- **Scalability:** We have described how decentralized resource allocation decisions are made through the market mechanisms coupled with agent reasoning methods for computing bids, and how services in high demand can be replicated. Note that, typically, new agents bring their own resources into the system with them; thus, when agent developers have incentive to introduce their agents, they are also introducing resources for managing the larger system (at a minimum, their agents have to decide for themselves on bids to make).

- **Robustness:** We have illustrated how the system can still perform well despite the potentially conflicting desires of the participating agents and the possibility of misrepresentation in the system.

Collectively, the mechanisms we have described can automate much of the administration of a digital library, including organizing information and services using ontological relationships, selecting, evaluating, and remembering useful services using machine learning, and deciding how to allocate (finite) resources to meet the evolving demands of a user community.

These results were achieved via the merging of different technologies which include: ontology design, market oriented design, and nested agent modeling and learning. The agents and protocols that we have described have all been implemented and integrated into the UMDL SMS. Some of these, such as the auctions and the Price Monitor Agent, are also used by agents in the deployed UMDL system. For example, these agents can automatically spawn more QPAs during high demand periods by the students using UMDL simultaneously from many schools. Thus, our SMS “information economy” model is already being used by UMDL patrons; while they are unaware of the economic foundations of the system, those foundations are already being used in a rudimentary way to improve the system performance for the users.

Of course, given the specific needs of the current UMDL user community, and the rights we currently have over the UMDL content, other possible means for controlling resources and providing services would work for the UMDL. Specifically, we could develop management algorithms tailored specifically to the UMDL. While it could be the case that, for the short term, such an alternative might be more effective, we are not convinced of its long-term viability when the system is truly open, dynamic, and large. While much remains to be done within the SMS strategy for addressing these long-term issues (many open issues have been raised throughout this paper), we believe that the SMS has already demonstrated promise in being able to meet the needs of the open, evolving, information economy of the future.

Acknowledgments The rest of the members of the UMDL SMS group all contributed to its design and development. They are: Anil Arora, Bill Birmingham, Eric Glover, Dan Kiskis, Anisoara Nica, and Bill Walsh.

References

1. Edmund H. Durfee, Daniel L. Kiskis, and William P. Birmingham. The agent architecture of the University of Michigan Digital Library. *IEE Proceedings on Software Engineering*, 144(1):61–71, 1997.
2. James Frew, Michael Freeston, Randall B. Kemp, Jason Simpson, Terence Smith, Alex Wells, and Qi Zheng. The Alexandria digital library testbed. *D-Lib Magazine*, July/August 1996.
3. Tad Hogg and Bernardo A. Huberman. Controlling chaos in distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1325–1332, December 1991. (Special Issue on Distributed AI).

4. Steven P. Ketchpel, Hector Garcia-Molina, and Andreas Paepcke. Shopping models: A flexible architecture for information commerce. In *Proceedings of the ACM Digital Libraries Conference*, 1997.
5. Robert M. MacGregor. The evolving technology of classification-based knowledge representation systems. In John. F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann.
6. Tracy Mullen and Michael P. Wellman. Market-based negotiation for digital library services. In *Second USENIX Workshop on Electronic Commerce*, 1996.
7. Sunju Park, Edmund H. Durfee, and William P. Birmingham. Advantages of Strategic Thinking in Multiagent Contracts (A Mechanism and Analysis). In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, pages 259–266, 1996.
8. Sunju Park, Edmund H. Durfee, and William P. Birmingham. Emergent properties of a market-based digital library with strategic agents. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS98)*, 1998.
9. Marting Röscheisen, Michelle Baldonado, Kevin Chang, Luis Gravano, Steven Ketchpel, and Andreas Paepcke. The Stanford InfoBus and its service layers. In *MeDoc Dagstuhl Workshop: Electronic Publishing and Digital Libraries in Computer Science*. To appear.
10. José M. Vidal and Edmund H. Durfee. Task planning agents in the UMDL. In *Proceedings of the Fourth International Conference on Information and Knowledge Management (CIKM) Workshop on Intelligent Information Agents.*, 1995.
11. José M. Vidal and Edmund H. Durfee. The impact of nested agent models in an information economy. In *Proceedings of the Second International Conference on Multi-Agent Systems*, pages 377–384, 1996.
12. Peter Weinstein and Gene Alloway. Seed ontologies: growing digital libraries as distributed, intelligent systems. In *Proceedings of the Second ACM International Conference on Digital Libraries*, June 1997.