

To Each and Everyone an Agent: Augmenting Web-Based Commerce with Agents

Joakim Eriksson, Niclas Finne, and Sverker Janson

Intelligent Systems Laboratory
Swedish Institute of Computer Science
Box 1263, SE-164 29 Kista, Sweden

Email: {joakime, nfi, sverker}@sics.se

Abstract

Internet has evolved from an information space to a market space with thousands, potentially millions, of electronic storefronts, auctions and other commercial services. This creates great opportunities, but is not without problems. One major problem is the difficulty of finding relevant offers. Another problem is coping with the multitude of different styles of web-based user interfaces to different marketplaces. Yet another problem is how to automate routine tasks in such an environment.

We present one possible solution to these problems. An *agent-based market infrastructure*, in which agents support all users and services, helps customers and commercial sites find matching interests, and, if desired, negotiate and close deals. The infrastructure is entirely open and decentralized. Each participant has an agent that acts in the interest of its owner. Interaction is entirely symmetric. Any participant can play any role on a market.

In this paper we present an integration of such an infrastructure, *SICS MarketSpace*, with the web. Personal assistant agents help users in their interaction with services and are able to handle routine tasks off-line. Agent-enabled services are able to adapt to the interests of their users, even on their first visit, and are provided with a mechanism to take the first initiative (push) in a highly focused manner.

Keywords: agent-based markets, software agents, worldwide web, electronic commerce, personal assistants

1. Introduction

Internet has evolved from an information space to a market space where there are thousands of electronic storefronts, auctions and other commercial services. There are now a large number of different types of commerce sites on the net, from simple storefronts to advanced shopbots.

Today, a web-shopper looking for, let us say, a telescope, might want to start by searching for telescope information and stores that sell telescopes. Looking for “+telescope +buy” in Alta-Vista gives almost 6000 hits. After a good deal of surfing, the shopper will find a few stores that actually sell telescopes, and may then proceed by comparing prices, models, etc., all manually. This is a lot of work and good offers may be missed due to sheer exhaustion. (Another problem is that even the best search engines necessarily cover only parts of the net.)

Shopbots have made it a lot easier to find and compare offers from the catalogs of regular storefronts. But, the end result could be that a small number of players will control both par-

participation, through their one-sided choice of searched storefronts, and interaction, through their choice of search, evaluation and negotiation mechanisms in markets. This is strongly in conflict with the notion of a free market. Anyone should be able to participate and each participant should be able to have complete control of its interaction with others.

In a companion paper [1] (based on previous work in [2]), we introduce a framework for an agent-based market infrastructure, *SICS MarketSpace*, in which all the participants in the market are supported by agents. Such an infrastructure can be open and decentralized, just like the web itself, allowing anyone to enter, while enabling automation to the degree of, and even surpassing, shopbots. Each participant has an agent that acts in the interest of its owner. No single site can control the market. Certain sites will serve more central roles than others, such as indices (cf. search robots), but these roles are open. Any participant may perform them and other participants can trivially switch to using them. Interaction is entirely symmetric. Any participant can play any role on a market.

In this paper, we describe how agents are integrated with the web in *SICS MarketSpace*, smoothly combining web browsing and agent-based automation. Personal assistant agents help users in their interaction with services and are able to handle routine tasks off-line. Agent-enabled services are able to adapt to the interests of their users, even on their first visit, and are provided with a mechanism to take the first initiative (push) in a highly focused manner.

The remainder of this paper is organized as follows: In Section 2, we briefly summarize *SICS MarketSpace* and discuss related work. In Section 3, we present the basic integration of agents with the web. In Section 4, we describe the proposed basic functionality of a personal assistant. In Section 5, we present usage scenarios. In Section 6, we discuss implementation issues. In Section 7, we offer some final conclusions.

2. Background and related work

2.1 *SICS MarketSpace*

The *SICS MarketSpace* agent-based market infrastructure [1] is intended to complement the web, email, and other human-oriented forms of communication. At its core is an information model, for describing user interests, items, contracts, etc, and an interaction model, defining a basic vocabulary for searching, negotiating and settling deals. These are for the agent infrastructure what HTML and HTTP are for the web. A brief summary is provided here to make this exposition somewhat self-contained.

The *information model* is based on structured documents representing contracts and representations of sets of contracts called *interests*. The names of elements of contracts are URLs referring to their type definitions. Element types (*concepts*) may be arranged in an inheritance hierarchy. The language for interests offers an ability to give alternatives for any element, generalize in an inheritance hierarchy, and give ranges for values. Currently, a custom designed language, the *Market Interest Format (MIF)* is used to encode interests. It is possible that W3C RDF/XML (Resource Description Framework mapped onto XML) [7] will become general enough to serve the purpose of representation language.

The *interaction model* is asynchronous message communication in a speech act based language, the *Market Interaction Language (MIL)*, in the manner of KQML [8] and FIPA ACL [9], and shares with these its Common Lisp based serial syntax. Compared to other similar languages, its set of message types is small and focused: ASK, TELL, NEGOTIATE, OFFER, ACCEPT, and REJECT. They all take a single interest as argument. Sending messages of types OFFER, ACCEPT, and DECLINE means making legally binding commitments, and hence requires support by digital signatures or other authentication mechanisms.

2.2 The SICS AgentBase software platform for agents

Our experimental environment is a (research prototype) software platform for agents developed in SICStus Prolog, *SICS AgentBase for SICStus* [4], consisting of libraries implementing basic agent facilities, several standard Internet formats and protocols, KQML/KIF (which are not used in MarketSpace), and MIL/MIF.

Several agents may co-exist in an OS process. In the default message transfer mode, messages are sent and received asynchronously over TCP/IP sockets. An agent address is in this case a URL of the type *map://domain:port/agentname*, where MAP stands for the Market Agent Protocol. Agent messages may also be sent through SMTP, FTP, and HTTP.

In a manner analogous to agent communication in MIL, agents can send and receive HTTP requests, making them web clients and servers (or individual pages). The web address of an agent defaults to *http://domain:port/agentname*. Parsing and generation of HTML to/from an internal representation can be performed automatically.

2.3 Other agent-based and otherwise related approaches

A *shopbot* is a program or service that automates shopping in web storefronts. For example, the web-based service Jango ([12], now part of Excite [13]) provides a simple interface for searching and buying from a number of storefronts. Its operators use tools that automate the creation of interfaces to the web based storefronts, to simplify this otherwise arduous task. Since both information and interaction are standardized in SICS MarketSpace, the creation of a Jango-like shopbot is a very simple task. More attention can instead be placed on domain specific value-adding services.

Personalization of services may be based on collaborative filtering techniques [16,17]. FireFly [17] provides a mechanism for automatically sharing profiles, based on the Open Profiling Standard (OPS) [18]. Our mechanism for communicating interests has some similarities with OPS, in that information serving as a profile can be automatically communicated between a user and a service, and can be used to personalize the service. In SICS MarketSpace, the user only provides information for a specific purpose and is also provided with an incentive for keeping this information up-to-date.

Anthony Chaves, Pattie Maes, et al, at MIT Media Lab, have developed an agent-based market called Kasbah [4,5]. Users may assign the task of buying or selling specified goods to an agent, which then performs negotiation and settlement of deals, fully automatically. The system was not intended to serve as a general market infrastructure and does not offer distribution or general information or interaction models.

The Stanford [10] and University of Michigan [11] Digital Library projects both employ agent-based architectures. These are, like most agent-based systems in the KQML spirit, influenced by a top-down hierarchical view of system design, a priori subdividing responsibilities into a number of components. With SICS MarketSpace we attempt to provide the minimum possible glue to enable automation of a market of self-interested participants. No special components are needed.

A number of object-oriented platforms have been proposed for building distributed commerce applications integrated with the web. The very ambitious CommerceNet eCo System project [6] aims to develop an architectural framework compatible with all major Internet commerce platforms. Interaction between agents in a Common Business Language (CBL) is suggested, but insufficient information is available to make a comparison with the framework presented here. One strong similarity is the notion of an ObjectWeb, the integration of web clients and servers with objects augmenting web-based interaction, in a manner analogous to the web-aware agents of SICS MarketSpace.

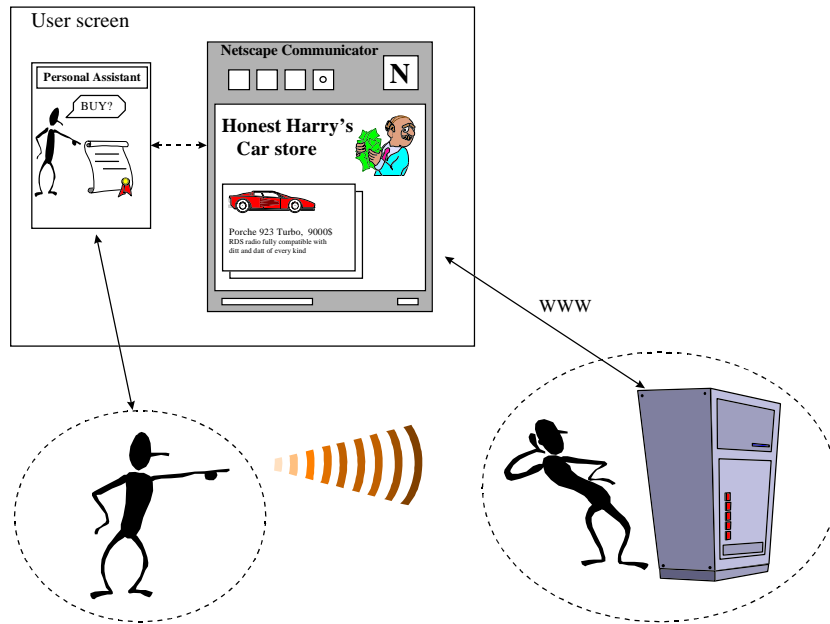


Figure 1: Integration of user and service agents with the web

3. Integration of user and service agents with the web

Figure 1 illustrates the basic integration of user and service agents with the web. The user will interact with various services as usual through one or more web windows (to the right). While the user is surfing, the user's *personal assistant* agent will be able to augment the web-based interaction by interacting (in MIL/MIF) with the agents of services. The user can interact with, and control, the personal assistant through a dedicated window (to the left, also see Section 4).

When the user visits an agent-augmented service on the web, the assistant will detect the presence of an agent "behind" the web page (see Section 6 for details) and establish a connection to that agent. The service will be able to query the assistant about its user's interests, or the assistant may take the initiative by telling the service about certain specific interests. This enables the service to personalize interaction, e.g., display offers relevant to the user's current interests.

MarketSpace is based on a notion of interests, which are explicitly provided by users and services as input to their respective agents. While the format based on structured documents is quite straightforward, it can be tedious to create interests in this form. The integration with the web enables web-based services to become an essential part of MarketSpace by providing user-friendly ways of entering interests. This could include interactively designing a car at a car dealer, creating a general "sailing" interest at the yacht club, or selecting a CD record from a catalog. When the description of an interest is completed, it is sent to the assistant.

The main goal of MarketSpace is to enable a high degree of automation of the Internet market. At any time, a user may delegate tasks to a handler agent of the assistant, to be handled off-line. Such tasks can be to look for good offers matching a certain interest, or to sell goods (described in another interest) in an auction. When off-line, the user can be notified about events through email or GSM/SMS (text messages in cellular phones).

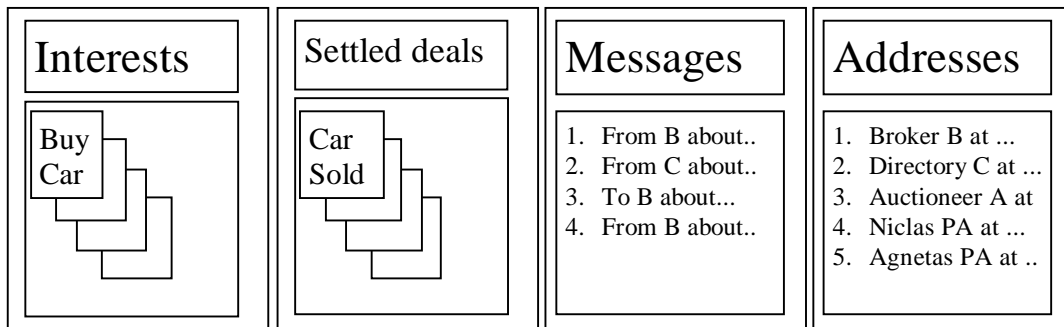


Figure 2: Types of information used in the Personal Assistant agent

4. The personal assistant

The personal assistant needs information to handle automation of commerce for its owner. It needs information about its owner's interests (including information about the owner himself), settled deals, messages received and sent, etc.

The following information is stored in the Personal Assistant (see Figure 2):

1. Interests - both interests of the assistant's owner and of other market participants. Interests are the key knowledge that the personal assistant has about its user, enabling it to search for other participants with matching interests, negotiate about prices etc. Interests are represented as structured data, based on templates that are shared by all agents in the market (which ensures that all agents know that they are taking about the same things). Interests are public or private. Public interests may be sent to any agent (cf. profiling information [18]).
2. Settled deals - information about deals that the assistant has settled (automatically or guided by its owner) when representing the owner.
3. Messages - these are either messages from other agents in the market or from the assistant agent to the user. Messages from other agents could for example contain offers from other market participants. Messages from the assistant could be that it needs more information on the red car that you are selling.
4. Addresses - these are links (or bookmarks) to other agents that the user can interact with. This information corresponds to the mail address book or the browser's bookmarks.

The behavior of the assistant is given by its current range of handlers. Handlers are plug-ins of the assistant. One handler can advertise an interest to a number of brokers. Another can attempt to find a good offer through a bidding process. A third can sell goods acting as an auctioneer with a given auction format. A user may give an interest to a handler, interact with running handlers, and inspect their results upon completion

5. Scenarios

This section presents a couple of simple scenarios illustrating how the personal assistant and agent infrastructure can be used. The first scenario illustrates the use of a personal assistant to search for and buy a product.

Assume that Alex has an Internet service provider that supplies him with a personal assistant for electronic commerce. Alex decides to buy the CD "Violator" by Depeche Mode. He surfs to his service provider and starts his Personal Assistant (PA).

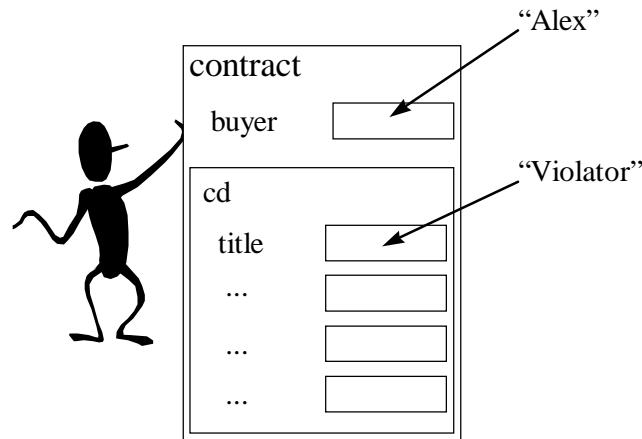


Figure 3 Alex describes an interest for the Personal Assistant

1. Using the PA, Alex describes an interest by choosing a contract and specifying himself as buyer. He then describes the CD he is searching for by picking a CD template and entering information in some of fields (see Figure 3). The interest is stored as public in the PA.
2. Alex wants to find someone who can sell him the CD so he surfs to a well-known directory service. His PA immediately detects that the directory service is agent-enabled and informs it what Alex is searching for. The directory service generates a new web page with a summary of some deals that might be interesting for Alex. At the same time as it sends the same information directly to Alex's PA.
3. There are several dealers interested in selling the CD and, since Alex does not care to see them all, he tells his PA to find the best deals without actually buying anything.
4. The PA investigates the summary received from the directory service and starts to negotiate with the sellers in order to find better deals. It finally generates a small summary with the most promising offers.
5. In the summary, Alex finds that a certain CD store offers the best price and surfs to it. The store is agent-enabled and Alex's PA informs it about the CD interest. The store produces a personalized web page with full description, pictures and music clips. Alex decides it looks promising but is not sure that the store is reliable. He examines the other offers before he finally decides that the store was the best choice and he tells his PA to settle the deal.
6. The PA settles the deal and gives the seller additional information such the address to deliver the CD.

In this scenario Alex manually investigated the merchants to see the offers but he could just as well have told his PA to find and buy the CD automatically. In that case the PA would have asked the directory service for sellers, negotiated with them and automatically settled a deal at the best price. The next scenario illustrates a merchant using an automated service to sell CDs.

Assume that Mindy has a small business selling CDs and that she has an agent-enabled storefront. She has just received a large shipment and needs to sell some of her old CDs quickly.

1. Mindy contacts her storefront through a PA and looks at the statistics view to see how the sales have gone. She then describes an interest to sell the old CDs at a substantially lower price than normal and instructs the PA to sell them.
2. The PA starts by advertising the CDs to some directory services. It also asks the directory services for parties interesting in buying and informs the found parties about the CDs. The PA then waits for interested parties to contact it.

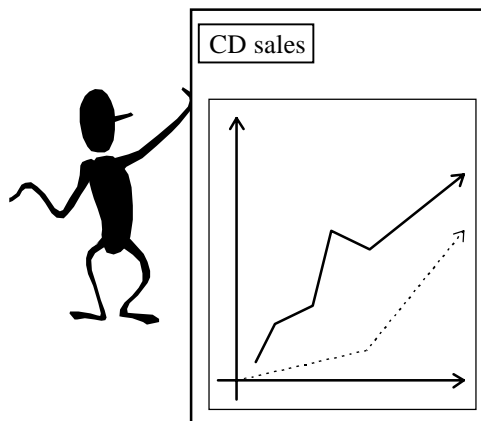


Figure 4 Mindy's Personal Assistant statistics view

3. After some time an agent contacts the PA and asks for matching interests. The PA investigates the received interest, finds that the interests match and replies with the matching interest.
4. The agent initiates a negotiation with the PA and after a time of negotiation, they agree on a price and a CD is sold.
5. When all CDs described by the interest have been sold, the PA marks the interest as satisfied. It also contacts the directory services and informs them that the interest no longer is valid. A success report is generated and sent to Mindy by email.

7. Implementation

We have developed a prototype of an agent system that was used to implement the Personal Assistant (PA) consisting of a core and a GUI. The core is an agent in the agent system that handles all functionality like searching, negotiation, etc. The GUI, which is implemented as a Java applet, is the interface between the user and the PA. In this section, we will focus on the PA GUI, implemented using Java and Netscape Navigator (from now on called Navigator).

When a user first accesses the PA using Navigator, a small JavaScript program will open a new Navigator window with a specific name and load the PA GUI applet into it. The applet immediately opens a TCP/IP connection to the PA and this connection is kept open as long as the GUI is running. All further communication between the user and the PA goes through the PA GUI.

The PA GUI can control the Navigator window that caused its window to open using JavaScript (a Navigator window may access its opener) and for example make it load another web page.

Commerce sites with agent support add a certain piece of JavaScript code to their web pages. This code opens a window with the same name as the PA GUI window and gets a reference back. If the window already is opened, a reference to that window is returned and this reference is used to call a public method in the PA GUI applet with information about the service agent. (A Navigator window can access the content in windows to which it has a reference.)

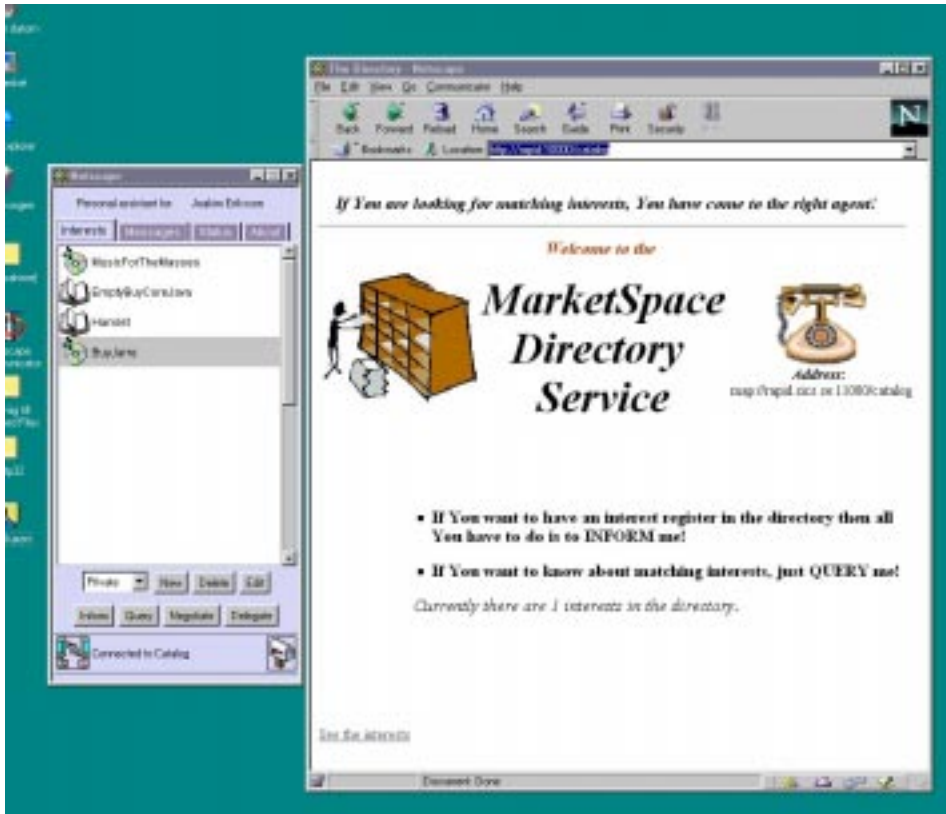


Figure 5: Screen-shot showing the personal assistant (to the left) and a browser window accessing a directory service.

The following code will give information about a Broker agent location to the PA.

```
<!-- Broker contacting the PA GUI -->
<SCRIPT LANGUAGE="JavaScript">
  var attributes = "toolbar=no, location=no,
    directories=no, status=no, menubar=no,
    scrollbars=no, resizable=no, copyhistory=yes";

  function declare_active() {
    AgentBrowser = window.open("", "agentbrowser",
      attributes);
    AgentBrowser.document.AppletLink.register_ab("Broker",
      "mink.sics.se", "25000", "auto");
  }
</SCRIPT>
<BODY onLoad="declare_active()">
```

When a user with a personal assistant surfs to a commerce site with the above code, the assistant will receive the address of the commerce site's agent ("mink.sics.se:25000"), through its register_ab method.

This simple implementation has some problems. First, it is not certain that Netscape will continue to support this kind of interaction between Navigator windows (for security reasons). Second, if some user does not have the PA GUI running when accessing an agent-enabled site a new window will open. Further, the PA GUI can only control the opener window and does not know anything about the other windows.

We are currently looking at other solutions to the problems. The last problem could probably be solved if the service provider can give a reference to the active window when informing the PA GUI about the agent behind the web page. Signed applets could be used to give the PA GUI higher access to the browser. Instead of having the web pages contact the PA GUI, it could inspect the other Navigator windows to check if some agent-enabled page has been

loaded. It is also possible to write client software (at least under Windows 95/NT) that more or less controls the browser. Another alternative is to use a proxy to scan the web pages for agent information at each access. The last two alternatives are somewhat less attractive since they require software to be installed on the client computer.

8. Discussion

Clearly, complementing the web with an agent-based market infrastructure could greatly improve the efficiency of Internet commerce. Whether the market can adopt such a solution remains to be seen. The problem is that it benefits each and everyone, but none in particular. The benefits appear only in large groups of users.

We believe that it is possible to create self-contained markets with moderate size groups of users, e.g., used book markets at universities, to whom both the technology as such and the added benefit will appeal. When such a group of users grows large enough, it will become an interesting target for other commercial actors, who will then have an incentive to adopt the technology, even though it also puts them under the pressure of automatic price comparison shopping. This could attract more users, hence more services, and lead to sustained growth and eventual complete adoption.

Another question is whether the niche of complementing the web will be filled by an “EDI for the web age” similar to ours or by a quite general distributed object framework. In our belief, it is essential that software acting on our behalf does so in “human terms”. Any interaction with others generated by my software, should in a strong sense be regarded as performed knowingly and willingly by myself. Even though I will trust my software to carry out routine tasks, I must always be able to predict what will happen, and be able to review it upon completion. This would have to be a restriction also on distributed object frameworks.

Acknowledgements

The research presented here was funded by KFB, the Swedish Transport and Communications Research Board, and carried out in collaboration with Telia Research AB.

References

- [1] Joakim Eriksson, Niclas Finne, and Sverker Janson. SICS MarketSpace – an agent-based market infrastructure. SICS Technical Report. Swedish Institute of Computer Science, February 1998. (Submitted to the Agent Mediated Electronic Trading Workshop of the 2nd International Conference on Autonomous Agents, 1998.)
- [2] Joakim Eriksson, Niclas Finne, and Sverker Janson. Information and interaction in MarketSpace. In *2nd USENIX Workshop on Electronic Commerce*. USENIX Press, 1997.
- [3] Joakim Eriksson, Niclas Finne, Sverker Janson, et al. An Internet software platform based on SICStus Prolog. Presented at *Logic Programming and the Web, WWW6*, San Jose, 1997. See <http://www.cs.vu.nl/~eliens/WWW6/papers/joakime/>.
- [4] Anthony Chavez and Pattie Maes. Kasbah: An Agent Marketplace for Buying and Selling Goods. In *Proceedings of PAAM'96*. Practical Applications Company, 1996.
- [5] Anthony Chavez, D. Dreilinger, Robert Guttman, and Pattie Maes. A Real-Life Experiment in Creating an Agent Marketplace. In *Proceedings of PAAM'97*. Practical Applications Company, 1997.

- [6] Jay M. Tenenbaum. *eCo System: CommerceNet's Architectural Framework for Internet Commerce*. White Paper. See <http://www.commerce.net/eco/>.
- [7] W3C Resource Description Framework. See <http://www.w3.org/Metadata/RDF/>.
- [8] Yannis Labrou and Tim Finin. *A proposal for a new KQML specification*. UMBC Technical Report 9703, 1997. See <http://www.csee.umbc.edu/~jklabrou/publications/tr9703.ps>.
- [9] FIPA ACL. Foundation of Intelligent Physical Agents. Agent Communication Language. See <http://drogo.csel.stet.it/fipa/spec/fipa97.htm>.
- [10] Stanford Digital Library Project. See <http://www-diglib.stanford.edu/>.
- [11] University of Michigan Digital Library (UMDL) Project. See <http://http2.sils.umich.edu/UMDL/>.
- [12] The Jango shopbot. See <http://www.jango.com>.
- [13] Excite. See <http://www.excite.com>.
- [14] EDI. See, e.g., <http://www.premenos.com/edi/edi.html>.
- [15] KIF. See <http://logic.stanford.edu/kif/kif.html>.
- [16] Net Perceptions, Inc. See <http://www.netperceptions.com/>.
- [17] FireFly Network, Inc. See <http://www.firefly.net/>.
- [18] Netscape on OPS. See <http://developer.netscape.com/ops/ops.html>.