SICS MarketSpace – An Agent-Based Market Infrastructure

Joakim Eriksson, Niclas Finne, and Sverker Janson
Email: {joakime, nfi, sverker}@sics.se

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

Abstract. We present a simple and uniform communication framework for an agent-based market infrastructure, the goal of which is to enable automation of consumer goods markets distributed over the Internet. The framework consists of an information model for participant interests and an interaction model that defines a basic vocabulary for advertising, searching, negotiating and settling deals. The information model is based on structured documents representing contracts and representations of constrained sets of contracts called interests. The interaction model is asynchronous message communication in a speech act based language, similar to, but simpler than, KQML [7] and FIPA ACL [8]. We also discuss integration of an agent-based market infrastructure with the web.

1 Introduction

The Internet has evolved from an information space to a market space with thousands, potentially millions, of electronic storefronts, auctions and other commercial services. This market space is not without problems. A major problem is the difficulty of finding relevant offers. Another problem is coping with the multitude of different styles of 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* helps customers and commercial sites find matching interests, and, when desired, negotiate and close deals. Each participant has an agent that acts in the interest of its owner. The infrastructure is entirely open and decentralized, just like the web itself, allowing anyone to enter the market. Interaction is entirely symmetrical. Any rôle on a market can be played by any participant. The benefits of automation, lowered transaction costs, are made available to all.

At the core of the infrastructure is a communication framework, consisting of an *information model*, for describing user interests, and an *interaction model*, defining a basic vocabulary for searching, negotiating and settling deals. The information model is based on structured documents representing *contracts* and representations of sets of contracts called *interests*. Interests are encoded in the *Market Interest Format* (*MIF*).

The interaction model is asynchronous message communication in a simple speech act based language, the *Market Interaction Language* (*MIL*). The rôles of MIF and MIL in the agent-based market correspond loosely to those of HTML and HTTP for the web. We hope to, by these examples, encourage further study into and development of *simple* yet useful standards that will pave the way to automated Internet markets, and avoid the unwanted generality and overwhelming complexity of most agent communication frameworks.

The proposed infrastructure is intended to complement, and work in close integration with, the web, email, and other human-oriented forms of communication. We will discuss how agents are integrated with the web, smoothly combining web browsing and agent-based automation.

In the remainder of this paper, we provide background and related work in the areas of electronic commerce and agent-based systems (Section 2), introduce the information model (Section 3) and interaction model (Section 4), discuss integration with the web (Section 5) and conclude with a discussion of future directions of this work (Section 6).

2 Background and Related Work

2.1 Electronic Commerce on the Internet

The Internet offers the hope and the promise of the global perfect market. In principle, the activities can be automated. But, in practice, since all development is driven by self-interest, it is entirely focused on producing advantages local to single or small groups of participants, not on producing a uniform platform for automation of commerce between participants that benefits all and none in particular.

A commerce model typically includes activities such as advertising, searching, negotiating, ordering, delivering, paying, using, and servicing (Fig.1, next page).

Storefronts and search engines offer Internet-wide search and negotiation, although not with much precision. Future developments of metadata, e.g., based on the W3C RDF/XML (Resource Description Framework mapped onto XML) [6], will increase precision. But to facilitate automation, and offer sufficient expressiveness, the metadata framework has to be based on uniform design principles.

Progress can be made without metadata. The web-based service (shopbot) Jango [1] (now part of Excite [13]) provides a simple interface for searching and ordering from a number of storefronts, thus serving as a kind of an integrating “meta-shop”. Its operators use tools that automate the creation of interfaces to the web based storefronts, to simplify this otherwise arduous task. If information and interaction were standardized, the creation of a Jango-like service would be a much simpler, almost trivial, task, and more attention could instead be placed on domain specific value-adding services.

Other services, e.g., auctions, also include negotiation mechanisms. If this trend is taken further, a single site could provide for all activities desired, a one site globally accessible marketplace, owned and controlled by a single participant. This is clearly strongly in conflict with the ideals of free markets.

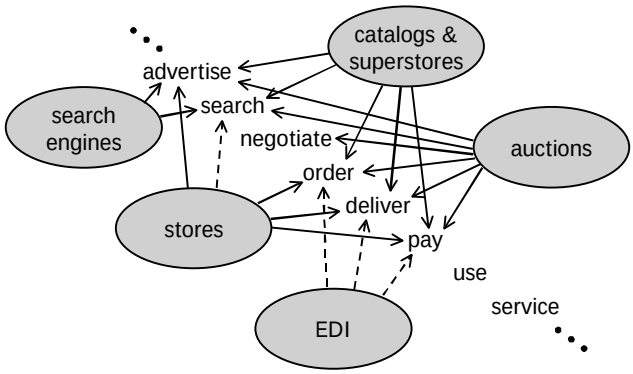

Fig. 1. Electronic commerce on the Internet

While EDI (Electronic Document Interchange [14]) offers standards for information and interaction between trading partners, current EDI standards are intended for use in static long-term relationships on the basis of detailed trading partner agreements, not for spontaneous commerce on the Internet.

A number of object-oriented platforms have been proposed for building distributed commerce applications integrated with the web. The most ambitious effort to date is perhaps the CommerceNet eCo System project [5], aiming 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. Another similarity with the work presented here is the integration of web clients and servers with distributed objects augmenting web-based interaction, in a manner analogous to our proposal for web-aware agents.

2.2 Agent-Assisted Commerce

We propose a framework based on the notion of agents, software components owned and controlled by the participants, which provide assistance to a range of market activities by interacting with the agents of other participants. Agents share a common language, a formalized subset of commerce communication, but are otherwise unrestricted in their behavior.

Anthony Chaves, Pattie Maes, et al, at MIT Media Lab have developed an agent-based market called Kasbah [4]. Users may assign the task of buying or selling a specified good to an agent, which then performs negotiation and settlement of deals, fully automatically, according to the users' choice of predefined strategy. The system has served a useful platform for experiments with groups of users [10], but was never intended to be a general market infrastructure, and hence does not offer distribution nor general information or interaction models.

The Stanford [11] and University of Michigan [12] Digital Library projects both employ agent-based architectures. These are strongly influenced by a top-down hier-

archical view of system design, a priori subdividing responsibilities into a number of components. Our goal is to provide the minimum possible glue to enable automation of a market of self-interested participants, who are free to assume the rôles of buyers, sellers, or various forms of mediators.

The agent communication language we propose is closely related to KQML [7] and FIPA ACL [8], but has a much smaller, task specific, set of message types. Similarly, our content language sacrifices the generality of languages such as KIF [15] for simplicity. Notably, KQML/KIF and FIPA ACL were developed primarily with a different class of applications in mind, distributed information/knowledge systems, with components interconnected through a network of facilitator agents, and with no concern for the self-interest of (owners or users of) component agents.

2.3 Motivating Applications

The SICS MarketSpace framework and infrastructure is being developed for and together with the following concrete applications (in close collaboration with Telia Research):

1) An agent-based marketplace for consumer goods and services, complementing, and emphasizing integration with, web-based commerce.
2) An agent-based workflow system for market-based organizations, which supports the dynamic generation of workflows from a market of available activities.

The examples in this paper pertain to the first of these domains.

3 The Information Model

3.1 Contracts and Interests

Our information model is based on the assumption that the goal of the activities of participants in a market is to close deals. As our basic information unit we choose the *contract*, for our purposes a structured document (see Fig.2). To assist the process of identifying which deals are possible, participants will, through their agents, exchange *interests*, which are (representations of) sets of contracts. Participants can use interests to advertise their true goals of buying or selling, or just reveal approximations of their true goals to enable an initial contact. Interests can be used both for communication and for the user models of user agents.

3.2 Example Interests

The following interests are easily expressed in terms of sets of contracts.

- Buy things cheaper than $1
- Buy pizza within an hour

- Sell these books
- Buy books on software agents

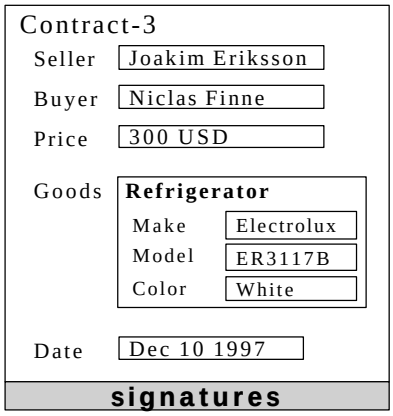

Fig. 2. A contract represented as a structured document

For example, the interest that I would like to buy something for a price of less than $1 is the set of contracts such that I am the buyer of a good (or service) and the price is less than $1. That I want something within an hour is a restriction on the date (a field in a contract type). And so on.

Requirements of the following more general kind

- Environmentally friendly
- What Joe (Jill) likes

may be captured as relations to other agents, which can be asked if they share a certain interest.

3.3 Concepts and Ontologies

Contracts are defined in terms of *concepts* (the record types that are the building blocks of the structured documents), which in their turn are defined in *ontologies* (collections/modules of concept definitions).

Concept identifiers are URLs, referring to concepts as local names in ontologies. Thus, a concept identifier “http://somesite.dom/basic.ont#contract-3” refers to a definition of “contract-3” in“http://somesite.dom/basic.ont”. Concepts only serve as building blocks of structured documents, no semantic information is attached to a definition other than the types of its components. They play the rôle of records in programming languages.

For certain tasks, agents will need to have knowledge of some concepts built in. For example, agents will typically need to know how to contact the agent of a person who has published a certain interest. More specialized agents will have knowledge of

product categories and know how to negotiate the price its user is willing to pay for a given quality. For other tasks, it is only necessary that users use the same concepts, or rely on services to bridge differences due to insufficient standardization, since they are only there to be matched against, not understood.

```
⟨def⟩ ::=
    (def ⟨name⟩ ⟨ref⟩
        (⟨name⟩ ⟨type⟩)*)
⟨type⟩ ::=
    integer | float | atom
    | string | date
    | (instance ⟨ref⟩)
    | (interval ⟨val⟩ ⟨val⟩)
    | (set ⟨type⟩)
    | (list ⟨type⟩)
    | (oneof ⟨val⟩*)
```

```
⟨expr⟩ ::=
    ⟨integer⟩ | ⟨float⟩ |
    | ⟨atom⟩ | ⟨string⟩
    | ⟨date⟩
    | (set ⟨expr⟩*)
    | (list ⟨expr ⟩*)
    | (instance ⟨ref⟩
        (⟨name⟩ ⟨expr⟩)*)
    | (or ⟨expr ⟩*)
    | (interval ⟨val⟩ ⟨val⟩)
    | (subset ⟨expr⟩*)
```

Fig. 3. The Market Interest Format (MIF)

The semantics and pragmatics of concepts will need to evolve both through formal and informal processes of standardization. They will only be reflected in the interpretation given to contracts by humans and in the behavior built into agents.

3.4 Encoding Interests

An interest is a set of contracts. By an *expression of interest* we mean a representation of an interest, in some language. Several possible languages could be used for encoding interests. In particular, the content languages KIF [15] and SL [8] from the KQML and FIPA ACL communities could be used. However, in their most general form KIF and SL are not computationally tractable, and well-defined proposals for useful subsets do not yet exist.

We propose initially to use simpler formats for describing structured documents. It is possible that W3C RDF/XML (Resource Description Framework mapped onto XML) [6] will become general enough to serve our purpose, but this is not yet available. To avoid working against a moving target, we are using a simple custom design language, the Market Interest Format (MIF).

3.5 The Market Interest Format (MIF)

The Market Interest Format is a simple frame language in Lisp syntax (Fig.3 above). As basic types, it offers numbers, symbols (atoms), strings and dates, and as composite types concepts (frames), sets, lists, and enumerations. When expressing an interest, basic types may be given as values or intervals (ranges), sets and lists may be given with any expression as elements, instances of concepts may be given with any subset

of attributes, alternatives may be given for any value, and subsets may be given for set types.

In Fig.4 (next page) are shown abbreviated examples of a MIF definition, which should be located in an appropriate ontology, and a MIF expression of interest, which could be an interest known to and stored in a user agent or the content of a message.

```
(def car "trade-object"
    (color (instance "pantone-color"))
    ...)
(instance "contract-3"
    (date (interval 1/1/98 6/30/98))
    (buyer (instance "person"
                (name "Joe Smith")
                (agent-address ...)))
    (goods (instance "car"
                (color (instance "red")))))
```

Fig. 4. A MIF definition and expression

The expression of interest says that Joe Smith is the buyer of a red car in a contract signed between 1/1/98 and 6/30/98. This could, for example, be used by Joe to advertise such an interest, or by the agent handling it to respond to incoming queries.

Note that we will assume that agent addresses are associated with the descriptions of the participants named in the contract. This eliminates the need for separate mechanisms for advertising interests and the addresses for agents.

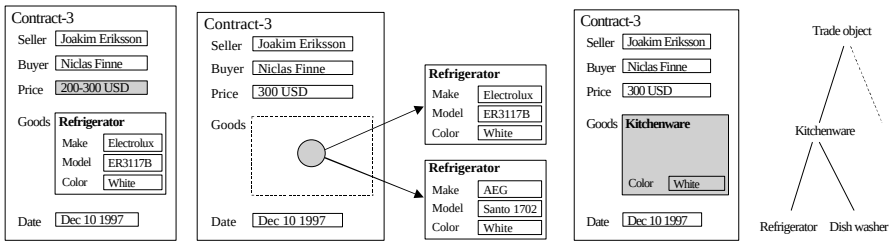

Fig. 5. Visualizations of MIF expressions. The leftmost illustrates intervals, the middle illustrates alternatives, and the rightmost illustrates generalization in a hierarchy of concepts.

MIF is designed with simple visual presentation of interests in mind (see Fig.5). Although users are not expected to enter their interests only in terms of structured documents (see Section 5.4), it is necessary that users are able to understand what they may reveal about themselves to other agents.

4 The Interaction Model

4.1 The Market Interaction Language (MIL)

The Market Interaction Language (MIL) is an agent communication language in the same family as KQML [7] and FIPA ACL [8], and shares with these its Common Lisp based serial syntax. (See Fig.6 next page for an example.)

```
(offer
   :from “map://onesite.dom/agent1”
   :to “map://othersite.dom/agent2”
   :in-reply-to i
   :reply-with j
   :language ”MIF 1.0”
   :content “<MIF expression>”
   )
```

Fig. 6. An example of a Market Interaction Language (MIL) message. The fields *from*, *to*, *in-reply-to*, *reply-with*, *language*, and *content* correspond to the fields with the same names in KQML. MIL expects references to ontologies to be part of the content language.

Below is an overview of message types (in an abstract syntax where *A* is the sender, *B* the receiver, and *eoi* an expression of interest which is the content of the message), The six message types fall into two groups, the noncommitting messages used for searching and advertising, and the committing messages used for negotiation.

The noncommitting messages follow with an informal description of their intended meaning.

- ask(*A*, *B*, *eoi*) - tell me an interest that matches *eoi*
- tell(*A*, *B*, *eoi*) - this *eoi* is an interest.
- negotiate(*A*, *B*, *eoi*) - give me an offer that matches *eoi*

The committing messages have a stronger meaning, in that they involve making legally binding agreements.

- offer(*A*, *B*, *eoi*) - this (signed) *eoi* is an offer
- accept(*A*, *B*, *eoi*) - this is an accepted (“positively” countersigned) offer
- decline(*A*, *B*, *eoi*) - this is a declined (“negatively” countersigned) offer

Note that these message types allow agents to forward offers, etc., made by other agents. This does not mean extending the offers to the new recipients. They serve as “for your information” messages, proving the existence of binding agreements.

4.2 Example

The following is a simple, but illustrational, conversation between A (an agent with the task to buy a refrigerator), D (a directory service), and B and C (refrigerator ven-

dors). For simplicity, interests are written in English with a brief interpretation as part of the explanation of each message.

- ask(A, D, “sell me a refrigerator”) – A asks the directory service D for interests matching an interest where A is the buyer and the good of type refrigerator
- tell(D, A, “B and C”) – the directory service replies with an interest where A is the buyer of a refrigerator and B and C are possible vendors
- negotiate(A, B, “sell me a refrigerator”) – A ask B for an offer matching an interest where A is the buyer of a refrigerator
- negotiate(A, C, “sell me a refrigerator”) – A asks the same of C
- offer(B, A, “Electrolux 3117B for $350”) – B gives A an offer (a signed interest) where B is the seller and A the buyer of a specific refrigerator at the price of $350
- offer(C, A, “Electrolux 3117B for $300”) – C gives A the same offer but for the price of $300
- offer(A, B, “C sells for $300”) – A prefers B as a vendor, and forwards C’s offer to A to B, suggesting to B to match or improve on the offer
- offer(B, A, “Electrolux 3117B for $300”) – B gives A an improved offer
- accept(A, B, “B’s last offer”) – A accepts B’s offer
- decline(A, C, “C’s offer”) – A declines C’s offer

By switching the rôles of buyer and seller in this example, we get an English auction style bidding procedure. The auctioneer sends negotiate messages to the participants to initiate bidding. Each bid is redistributed to the other participants. The auctioneer ends the auction by accepting the highest, and declining the rest. Similarly, and given that the parameters were known by the agents, we could emulate a simple form of Dutch auction by letting the auctioneer initiate bidding by a negotiate message, thereby starting the clock. Bids are required to be inversely proportional to the time when they are given. The auctioneer ends the auction by accepting the first valid bid. There are many other aspects to implementing auctions, but we believe that agent-agent communication in a wide range of auction types can be supported by MIL and MIF.

4.3 How to Talk to Whom?

By introducing several different interaction protocols, such as auctions, we introduce the problem of knowing and deciding how to talk to whom.

We have adopted the solution of associating the interaction protocol with the agent handling the interest. The expected protocol and its parameters is expressed in MIF and is associated with the agent address in interests. Several handler agents can be named that use different interaction protocols. This solution offers sufficient flexibility for our present purposes. The interaction protocol has to be known and the problem remains how to introduce new protocols into the agent-based market.

5 Integration with the Web

5.1 Agents and the Web

Seamless integration with the web is critical. The web is the way people access information and services on the Internet, now and in the forseeable future. Fig. 7 (next page) illustrates the basic setup for our integration of agents and the web.

The user has two browser windows, one (small) for interacting with the user agent, another for accessing the web interface of services. When the user accesses agent augmented services through the web, the user agent is informed (e.g., using JavaScript) and is given the opportunity to contact the service and assist the user.

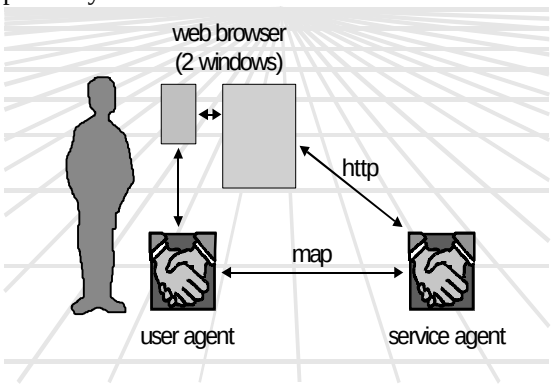

Fig. 7. Interaction between agents and the web. (MAP is the Market Agent Protocol for plain TCP/IP transmission of messages between agents.)

5.2 Personalized Services

The scheme for integration with the web also allows for personalization upon first contact. (1) The user accesses the service via the web browser. (2) The service sends a "redirect" web page containing service agent contact code whereafter it awaits a first contact from the user agent. (3) The web browser sends the contact code to the user agent, which contacts the service agent. (4) The service agent is notified about the user's current interest and generates a suitable personalized page.

5.3 The Agent Browser

The (small) agent browser window is the user interface to the user agent. It will typically reside on a contiuously running server to never be off-line, while the agent browser may run on any machine. The agent browser offers functions such as creating, editing, and viewing interests, manually sending and receiving messages, and assigning sub-agents (with unique addresses and possibly different protocols) to handle the interests. Interest handling agents have their own user interfaces for setting

user preferences, supervising progress, and for reporting results upon completion of the task.

5.4 Services that Generate Interests

The interest editor in the agent browser is a general editor for structured documents, and not all that intuitive for describing specialized interests. As a complement to this we envisage a plethora of agent augmented web-based services that specialize in offering tools for describing cars, houses, etc (see Fig. 8, next page). The completed interest is delivered to the user agent by agent communication at the request of the user.

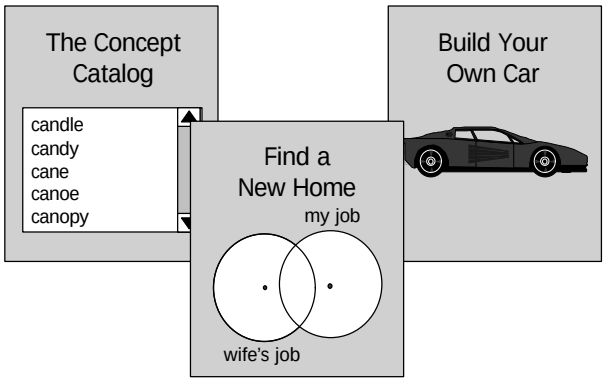

Fig. 8. Services on the web for creating interests. A simple service of the first form is available in the current prototype.

5.5 The SICS MarketSpace Prototype

The current SICS MarketSpace prototype was developed entirely in Java. (An earlier prototype was developed in Prolog [2,3,9].) It consists of:

- A personal assistant agent allowing the user to describe his/her interests in terms of structured documents, interact directly with other agents, and access agent augmented web services.
- Two agent augmented web shops, which generate personalized pages (show special offers relevant to the user's current interests).
- A directory service, which allows user and service agents to register interests and find agents with matching interests.
- An agent augmented web-based interest description service that allows users to describe their interests in terms of suitable MIF concepts. Completed interests are sent to and stored in the user agent.

The prototype is based on the SICS JavaBase library for implementing web- and internet-aware agent based systems. It consists of classes for:

- communication in MIL (and KQML)
- reading, writing, matching, typechecking MIF
- web clients and servers
- miscellaneous other Internet formats and protocols
- examples: web page objects, web file servers, cgi scripting, negotiating agents, interest directories, auctions, simple assistants

SICS JavaBase is available from the authors for non-commercial purposes, with the usual caveats for experimental research software.

6 Discussion and Future Work

We have presented very simple information and interaction models that could serve as a starting point for creating an agent based market infrastructure. The design strongly emphasizes simplicity and leaves room for extensions in a number of directions.

In this volume can be found several sophisticated proposals for market mechanisms, agent behaviour, and agent-user interaction. To the extent that these works deal with agent-agent interaction, we believe that most of them could be implemented in terms of MIF and MIL, and hence be supported by SICS MarketSpace.

But, not much work has been done in the direction of decentralized markets, and we expect higher requirements soon. To this end, we are currently exploring extensions in a number of directions, three of which are discussed below.

6.1 Interests with Preferences

MIF currently allows interests to be expressed as sets of contracts. No preferences in this set can be expressed. This is a concious design choice. Interests are not intended to reveal more information than necessary to enable participants with matching interests to find oneanother. Negotiation will locate the mutually preferred deals. A user's preferences are expressed to the agent that performs negotiation.

However, we may wish to delegate this task to a mediator, in which case we need to express our priorities. For this purpose, we are exploring representations supporting utility functions and/or uncertainty.

6.2 Openness to New Interaction Protocols

MIL provides a basic vocabulary for market interaction. Together with the convention that interaction protocols can be named and parameters given as parts of interests, considerably flexibility is achieved. But, the interaction protocols have to be known beforehand, and MIL cannot express every possible statement in market interaction.

We are exploring a new architecture that uses interaction protocol plugins, which are retrieved dynamically as needed. The problem is moved from agent-agent protocols to agent-plugin interfaces. If plugins can be trusted, these interfaces can be made very simple and generic for large classes of protocols.

6.3 Openness to New Information Formats

Once the ability to download trusted components is available, the interest format standard, MIF, can be replaced by an information interface standard, making the benefits of object-oriented programming available to agent based systems, but not in the form of distributed objects, nor in the form of mobile agents.

In this view, MIF is replaced by GOF (the Generic Object Format), the purpose of which is to provide a global namespace of record types, for which the record definition and its properties (the ontology) can be retrieved using the name as a URL, as in MIF, possibly the smallest possible foundation for agent based computing.

References

1. The Jango shopbot. See http://www.jango.com.
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 and Niclas Finne. MarketSpace: an open agent-based market infrastructure. Master's Thesis. Swedish Institute of Computer Science, 1997.
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. Jay M. Tenenbaum. eCo System: CommerceNet's Architectural Framework for Internet Commerce. White Paper. See http://www.commerce.net/eco/.
6. W3C Resource Description Framework. See http://www.w3.org/Metadata/RDF/.
7. Yannis Labrou and Tim Finin. *A proposal for a new KQML specification*. UMBC Technical Report, 1997. See http://www.csee.umbc.edu/~jklabrou/publications/tr9703.ps.
8. FIPA ACL. Foundation of Intelligent Physical Agents. Agent Communication Language. See http://drogo.cselt.stet.it/fipa/spec/fipa97.htm.
9. Joakim Eriksson, Niclas Finne, Sverker Janson, et al. An Internet software platform based on SICStus Prolog. In *WWW6 workshop "Logic Programming and the Web"*. See http://www.cs.vu.nl/~eliens/WWW6/papers/joakime/.

10. Chavez, D. Dreilinger, R. Guttman, and P. Maes. A Real-Life Experiment in Creating an Agent Marketplace. In *Proceedings of PAAM'97*. Practical Applications Company, 1997.
11. Stanford Digital Library Project. See http://www-diglib.stanford.edu/.
12. University of Michigan Digital Library (UMDL) Project. See http://http2.sils.umich.edu/UMDL/.
13. Excite – Search Engine. See http://www.excite.com.
14. Getting Started with EDI. See http://www.premenos.com/edi/edi.html.
15. Knowledge Interchange Format Specification. See http://logic.stanford.edu/kif/specification.html.