World Wide Web 3 (2000) 43–52 

# V-Market: A framework for agent e-commerce systems

Pedro S. Ripper, Marcus F. Fontoura, Ayrton Maia Neto and Carlos José P. de Lucena

*Software Engineering Laboratory (LES), Computer Science Department, Pontifical Catholic University of Rio de Janeiro, Rua Marquês de São Vicente, 225, 22453–900 Rio de Janeiro, Brazil*

E-mail: {pedro, mafe, ayrton, lucena}@les.inf.puc-rio.br

Software agent technology is still an emerging technology, and as such, agent based software design is still in its infancy. Software agents have just started to be used in the e-commerce domain, and they are already beginning to create a series of new possibilities for this arena. Agents can be used to automate, as well as to enhance many stages of the traditional consumer-buying behavior process. This paper proposes a software engineering approach to the design of agent mediated e-commerce systems, through the definition of an object-oriented framework. The paper presents the underlying concepts, and the architecture of the environment, showing how it allows developers to customize virtual marketplaces, and to define transaction categories on demand, incorporating many possible products and services that can be traded online.

## 1. Introduction

Software agent technology is starting to create a series of new possibilities for the area of electronic commerce [Bradshaw 1997]. Agents can be used to automate, and enhance many stages of the traditional consumer-buying behavior process. Through the minimization of transaction costs, elimination of geographic barriers and time issues, many new markets, not viable before, are now being created, traditional markets are becoming more efficient, and the role of the middleman has been changing drastically [Guttman *et al.* 1997].

This paper proposes a software engineering approach to the design of agent mediated e-commerce systems, through the definition and implementation of an object oriented (OO) framework. This framework focuses mainly on e-commerce applications based on virtual marketplaces. A virtual marketplace is an Internet based system, in which software agents interact and negotiate, on behalf of their respective users, to buy, sell or find specific goods and services [Leebaert 1998]. In this type of system, all users can be potential buyers, sellers or both, depending on their specific interests.

The main goal of V-Market is to facilitate the creation of this type of applications, as well as to make them more robust and flexible. It is expected that this approach will greatly enhance the process of experimentation, and research on the new possibilities brought about by software agents to the e-commerce application domain.

V-Market gives developers the ability to customize virtual marketplaces, and define transaction categories on demand, incorporating many possible products and services that can be traded online. Its users can create new transaction types and items based on individual needs, defining customized software agents adapted to the new products and offered services. Software agents in V-Market proactively broker and negotiate with interested buyers and sellers represented by their respective agents. They can be created with any set of desired behaviors, thereby enabling the consumer to have a virtual presence in the marketplace to further his or her interest, while freeing the consumer from constant monitoring of market progress.

The next sections of this paper are organized as follows: section 2 defines software agents through the presentation of their properties; section 3 describes the concepts underlying agent-mediated e-commerce, and also describes related systems which have been studied for the definition of V-Market; section 4 presents the OO framework, highlighting its design and implementation issues; finally, section 5 presents our conclusions and future research directions.

## 2. Software agents

A software agent is neither a new concept nor a well-defined one. Although not clearly defined, agents have been used for quite some time in many different fields of computer science, and depending on the field of study, its definition may vary. For the purposes of this work, an agent is defined as a piece of software (not dependent on its implementation language) used to automate specific tasks [Bradshaw 1997]. This piece of software also needs to be proactive, to be capable of personalization, and to have a certain level of autonomy. Other features, such as mobility and collaboration, might be desirable for some applications, but are not considered as prerequisites for this definition.

- Personalization: The ability of the agent to be customized through information provided, explicitly or implicitly, by its user. Examples of such personalization are different parameters or domain restrictions for a search agent, or a maximum price and a negotiation strategy for a shopping agent.
- Proactive: Agents should not simply act in response to external events. An agent should have a main goal,

and should take the initiative, whenever necessary, to accomplish its goal.

- Autonomous or semi-autonomous: An autonomous agent has the ability to complete its tasks without the intervention of its users, which means that agents should have a considerable degree of control over its own actions.

Although concepts such as intelligence, adaptability, mobility and cooperation are, many times, closely related to agents, we do not consider them prerequisites for defining an agent, but orthogonal concepts of the agent definition. These concepts are going to be seen as desirable features depending upon the task to be performed by the agent.

- Intelligent/adaptive: An agent's intelligence is directly related to its developer's ability to define its behavior. Therefore, this property is quite subjective, and most of the researchers prefer to call them semi-intelligence. There are two schools of thought on the development of the so called "smart" or "semi-intelligent agents": One is in favor of the creation of a complex set of predefined rules that defines the agent's behavior (specialist systems like). The other favors a simpler set of rules in which agents are capable of analyzing their actions and their surrounding environment as a way of "learning" how to be more efficient. This is sometimes identified as the ability to be adaptive.
- Mobile: Agents can be classified as mobile or non-mobile (sometimes referred as anchored agents). Non-mobile agents reside either in a client or in a server machine, and do all their work on either one or the other. A mobile agent, on the other hand, has the ability to "navigate" through servers collecting information or performing other small actions in order to complete its task.
- Cooperative/interactive: Agents can act on their own, or can cooperate and interact with others agents to complete their tasks. Buying agents can interact with selling agents to try to close a deal (like in V-Market and Kasbah [Chavez and Maes 1996; Guttman *et al.* 1997, 1998]). Alternatively, agents can exchange information about their respective users in order to complete their tasks in recommendation systems (such as Firefly [Firefly Inc. 1996]). Although there is no standard language or interface for this type of agents interaction, some agent based languages and protocols have been proposed, such as KQML (Knowledge Query Manipulation Language) [Finin *et al.* 1997] and Ontolingua [Gruber 1992]. These languages have been designed to allow heterogeneous agents and systems to cooperate in their tasks.

The above properties provide a common ground for understanding what agents are like, and how they can automate a series of tasks, such as those for searching/filtering, buying and selling products over the Internet. However, there are some other factors that make them a compelling concept, especially for the e-commerce domain:

- Information overload: Too much information available is not useful for making decisions if this information cannot be filtered on time. The time to absorb all the information available when making decisions is very scarce. On the Internet, there is a scarcity of demand, and not of supply [Guttman *et al.* 1997].
- Information ignorance: Useful information relevant for making decisions is quite often not used, because the decision-maker is unaware of them.
- Closer to "perfect market": In the digital world, transactions can occur independently of the physical location of the participating parties. In addition, other factors that influence the market, such as transaction cost and duration of transactions, can be greatly reduced by automation.

## 3. AmEC: Agent mediated e-commerce

Given the exponential increase on information resources available on the Internet (World Wide Web), software agents have been given a lot of attention lately. Agents have the distinguishing ability to automate repetitive and time-consuming tasks, including searching, buying and selling products over the Internet. Most of the tasks involved in the consumer buying behavior process [Guttman and Maes 1998b; Guttman *et al.* 1998; Moukas *et al.* 1998] can be automated. Stages, such as identification of needs, product brokering, merchant brokering, and negotiation, can now be assisted or automated by many different agent-based systems. Merchants are currently struggling to explore new channels to negotiate their products, looking for opportunities to maximize their profits and, at the same time, to satisfy consumers. However, most of the electronic commerce stores available on the web today still take the form of static "catalogs" of products, in which customers select items manually, and purchase them online. So far, this model has fallen short on redefining the marketplace [Guttman *et al.* 1998; Terpsidis *et al.* 1997]. It is expected that software agents will turn existing markets into more efficient ones, and will change the role of the middleman, and make many small niche markets viable [Chislenko 1998].

### *3.1. Consumer buying behavior model*

The consumer-buying behavior model (CBB) [AmEC 1996] defines the decision processes which people undergo when purchasing a product. Many different models try to capture this behavior through the definition of a set of consecutive stages. They represent a simplification of a very complex behavior, in which the stages are not discrete entities. Normally, stages can overlap, and even be concurrent and iterative. However, even though limited and simplistic, these models provide an important tool to elicit under which circumstances agent mediated electronic commerce systems apply to the consumer shopping experience [Guttman *et al.* 1997].

Table 1
Online shopping systems vs. CBB model.

| | Persona | Firefly | Bargain | Excite's | Kasbah | Auction Bot | Auction Web | T@T |
|---|---|---|---|---|---|---|---|---|
| 1. Need identification | | | | | | | | |
| 2. Product brokering | X | X | | X | | | | X |
| 3. Merchant brokering | | | X | X | X | X | X | X |
| 4. Negotiation | | | | | X | X | X | X |
| 5. Payment and delivery | | | | | | | | |
| 6. Service and evaluation | | | | | | | | |

Each of these models has its own characteristic and peculiarities, but they normally agree on a set of fundamental stages to represent the consumer buying process. The stages defined by the CBB model are:

- Needs identification: Sometimes also called Problem Recognition, this stage represents the awareness of the consumer's need.
- Product brokering: In this stage, the consumer decides what to buy. He or she evaluates a series of different products, and tries to identify which one would satisfy his/hers needs.
- Merchant brokering: At this stage, the consumer already knows what he or she wants, and decides whom to buy the product from. The decision is based on a set of criteria, such as price warranty, availability, reputation, and so on.
- Negotiation: This stage determines how the transaction is going to occur. Many of the traditional models do not identify this stage explicitly, but the separation of this process into a new stage is very useful for determining agents' roles.
- Purchase and delivery: This stage can sometimes signal the end of the negotiation stage. Things, such as the payment process and delivery, occur here.
- Product services and evaluation: This stage includes product services, customer services, and an evaluation of the satisfaction with the product itself, and with the buying experience as a whole.

Most of the agent systems available so far concentrate on automating or assisting mainly three stages: Product brokering, Merchant brokering, and negotiation, as shown in table 1. These systems generally focus on only one of these stages, but it is becoming more common for systems to incorporate more than one stage (such as Kasbah [Chavez and Maes 1996], T@T [Tete-a-Tete 1997]). This second generation of agents systems tries to resemble more closely the real buying experience, in which the stage transitions are not completely discrete.

*3.1.1. Agent mediated product brokering*

As presented above, Product brokering is the stage in which the consumer decides what to buy. There are a variety of agent systems that assists consumers deciding how each product fits best his/hers needs, such as Personalogic [Personalogic 1997], Firefly [Firefly Inc. 1996], and Tete-a-Tete (T@T) [Tete-a-Tete 1997].

There are many possible approaches to accomplish this task. Personalogic [Personalogic 1997] is a tool that allows customers to narrow down on products that best fit their needs by guiding them through a large product feature space. The system then filters out unwanted products within a given domain by letting the customers specify constraints about the product's features. Currently, Personalogic is offered as a service from merchant to customers, but it can also be used to suggest products from different vendors.

The approach for Product brokering used by Firefly [Firefly Inc. 1996] consists in filtering a product based on "word of mouth", instead of by constraints about its features. It uses a recommendation engine called automated collaborative filtering (ACM). ACM works comparing customers' product ratings. The first step taken by ACM is to find users with similar tastes for a specific type of product. Then, Firefly recommends new items, based on ratings given by other users with similar tastes. This approach has been used to recommend commodities, such as books and music, in which there is a substantial subjective factor in their description.

*3.1.2. Agent mediated merchant brokering*

At this phase, the customer compares different merchant alternatives for a specific product (chosen in the previous stage). In this category, Andersen Consulting's BargainFinder was one of the first shopping agents to look for merchants on a price basis. Given a specific product, BargainFinder queries its price from several different merchants using the HTTP protocol.

Another system for merchant brokering is Jango [Chavez *et al.* 1997a]. It represents a more advanced version of BargainFinder that integrates both Product and Merchant brokering in a single stage. Users can specify a series of constrains for specific features of a product category. Jango returns, then, product offers from different merchants that match the established criteria.

One last example of merchant brokering is Media-Lab's Kasbah [1996]. Kasbah is a multi-agent system for consumer to consumer electronic commerce. Every user that wants to buy or sell an item creates an agent with some strategic directions and sends it to a centralized virtual marketplace. In this marketplace, agents interact with each other trying to find potential buyers and sellers for their respective products. The goal of the agent is to make the

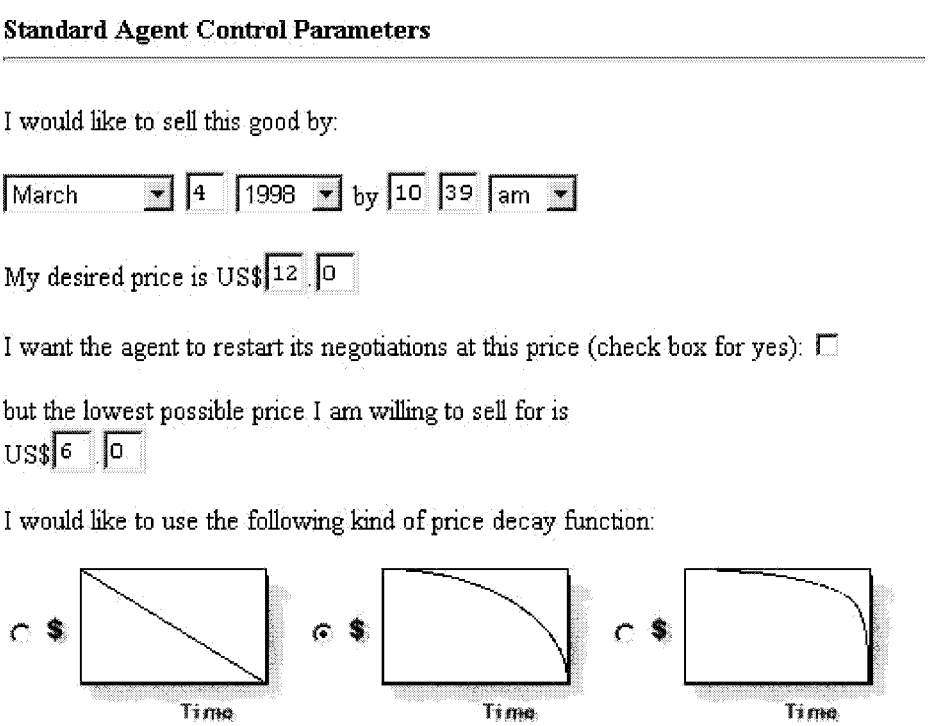


Figure 1. Kasbah's selling agents control parameters.

best possible deal in accordance to a set of user specified constraints.

*3.1.3. Agent mediated negotiation*

Negotiation is the process of determining the terms of a transaction (e.g., price). The main advantage of dynamically negotiating price for products or services is that the merchant is free from the burden of predefining the value of the products/services. Rather than that, this responsibility is transferred to the market itself. Therefore, limited resources tend to be allocated more fairly [AmEC 1996].

One of the main problems with negotiation is that it often requires all parties to share the same physical location, which is especially true for most of the traditional auctions. In addition, negotiation can be too complex or frustrating for the average consumer, since some negotiation protocols take place over an extended period, eliminating its usefulness to time-constrained users. Besides all that, transaction costs in negotiations may be too high for either the consumers or the merchants most of the time.

However, fortunately most of the issues and limitations described above disappear in the digital world. Things, such as geographical location, are not that important anymore in this new paradigm. Agents systems can assist the customers on the negotiation terms, and on the process of a transaction, making them fast, easy, and extremely cheap.

Both AuctionBot from University of Michigan and Ebay [Ebay Inc. 1995] are Internet classified auction servers. Users can create online auctions to sell a desired item by specifying some parameters. Examples of parameters normally used include auction start and end times, minimum bid, and reserved price. Once the auction is created, the seller's negotiation process is completely automated by the system, which uses the auction protocol and parameters chosen by its creator.

Kasbah [Chavez and Maes 1996] automates both the merchant brokering, and negotiation stages. The agents created to buy and sell items under the virtual marketplace are provided with different parameters, including negotiation strategies: Anxious, cool headed, and frugal (corresponding to a linear, quadratic, and exponential function for increasing/decreasing its bid for a product over time). Figure 1 illustrates these parameters.

*3.2. Virtual marketplaces*

Derek Leebaert defined virtual marketplaces as follows: "The marketplace is the place of exchange between buyers and sellers. Once one rode a mule to get there; now one rides the Internet. An electronic marketplace can span two rooms in the same building or two continents" [Leebaert 1998].

For the purposes of this work, this definition will be restricted a little further. From V-Market's perspective, virtual marketplaces are Internet based systems that allow software agents to interact and negotiate, on behalf of their respective users, to buy, sell, or find specific goods. However, these systems must have some additional features:

- Customer to customer: All users can be potential buyers, sellers or both, depending on their specific interests, which means it is a market for customers to customers, in which there are no predefined product offer or merchant entities.

- Centralized: It is a semantically centralized system, which means that although the systems can be internally distributed (run on more than one machine), to the outside user, it is a unique centralized marketplace, in which all participants meet to broker and negotiate their belongings.

This allows the use of a centralized homogeneous ontology for the items being traded on the marketplace, allowing more advanced brokering and negotiating strategies that would not be possible with a heterogeneous ontology. Also, centralizing the marketplace allows the existence of a single "owner/administrator" of the marketplace, who is responsible for specifying the rules that all users must follow under the marketplace. Things, such as the type of agents, negotiation strategies, and protocol, will be all standardized, greatly facilitating the integrity, and control of the marketplace. This allows the underlying structure of the marketplace to be more forgiven over integrity and security issues, and more flexible, allowing high levels of customizations.

Kasbah [Chavez and Maes 1996] is a perfect example of a system based on a virtual marketplace. Its users go to the marketplace (represented through a web site) to find, buy or sell their items. This is done through the creation of agents that can be seen as "smart ads" that will not only broker corresponding ads, but also negotiate the item.

V-Market is an object-oriented framework for the virtual marketplace domain that follows the above definitions. All systems instantiated by V-Market are semantically centralized, and there is no intrinsic distinction between buyers and sellers. One last restriction added is that all brokering and negotiation process happens asynchronously, which means that users create their agents with their respective control parameters and item description at a specific time and the brokering and negotiation process starts at that moment, but continues over time, completely independent of the users' presence, not requiring any type of real-time interaction.

## 4. V-Market: design and implementation

Based on a throughout study, and on an implementation of different agent mediated e-commerce systems, especially those previously described [AmEC 1996; Chavez and Maes 1996; Tete-a-Tete 1997], we have developed an object-oriented framework for the development of applications based on virtual marketplaces. It is intended to facilitate the development of such applications, enabling its users to focus on the higher level aspects of agent mediation. The framework has been developed using the modeling approach proposed in [Fontoura 1999], in which the variation points are explicitly represented in the design diagrams.

The framework proposed is greatly inspired on Media Lab's Kasbah [Chavez and Maes 1996] and concentrates mainly on the ability to create applications based on virtual marketplaces, in which buying and selling agents interact. Its implementation makes use of lower level agent services provided by some of the agent toolkits mentioned earlier, and builds upon these tools to create a higher level design solution for this specific application domain. The applications are created from V-Market through instantiation.

### 4.1. Design model

One of the most common problems in framework design is to find a balance between generality and ease of use. Normally, the more generic and flexible a framework is, the harder it is to use, and most of all, the harder it is to build. After participating on the development of some agent based e-commerce systems, such as Kasbah [Chavez and Maes 1996], Tete-a-tete [Tete-a-Tete 1997], and the AMEC infrastructure [AmEC 1996], it was possible to identify some specific points in the design of these applications that are critical to their flexibility. These specific variation points (or hot-spots [Fontoura 1999]) allow the instantiation of a wide range of different applications without requiring any fundamental change in the framework design.

V-Market allows the creation of marketplace applications, which are the framework instances. These instances are created through the adaptation of each framework hot-spot. The most important hot-spots in the V-Market architecture are summarized below.

*Multiple item support, and structured item ontology.* One of the main problems faced with the current implementation of Kasbah [Chavez and Maes 1996] is that it is completely tied to the two types of goods that it now supports (books and CDs). To add a new type of product, it is necessary to make a major change to the system's structure. The current implementation of the buying and selling agents and the persistence scheme are extremely tied to the specific description of each item.

V-Market addresses these issues by allowing for new types of goods to be easily added to the virtual marketplace. In order to make it possible, the goods' definitions should be generic enough to support not only commodity type goods, such as books and CDs, but also intangible type of goods, such as knowledge about a specific subject, skills, or services. Also, a standard item description/structure must be developed, so that agents do not need to be redesigned for every new item added to the marketplace.

Each item in the framework must be able to store and compare its attributes. Figure 2 illustrates this hot-spot using the UML-extended design language proposed in [Fontoura 1999]. The keywords {variable, dynamic} indicate hot-spot methods that may have implementation defined during runtime. In this case, it means that methods *write()* and *compare()* have to be defined by the user that adapts V-Market, and that this definition may take place during runtime.

*Multiple negotiation dimensions and strategies.* Kasbah [Chavez and Maes 1996] allows only one negotiation di-

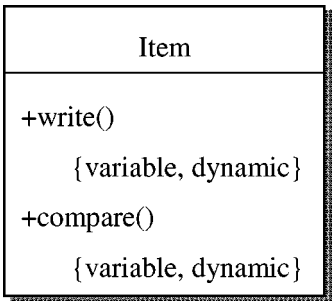


Figure 2. Item hot-spot.

mension (price), and three negotiation strategies for this specific dimension: Anxious, greedy/frugal, cool-headed. These negotiation strategies are basically price decay functions over time in the case of selling agent, or price increase functions in the case of buying agents (figure 1).

Although these negotiation strategies are quite simple, and do not necessarily obtain the best deals in a given virtual marketplace, they have been chosen because after some experiments, it has been identified that users normally do not fully trust their agents if they do not completely understand what their rationale is. Therefore, these strategies have been simplified for the sake of user understanding and trust.

Once these trust barriers are overcome, it will be important to experiment with different and more complex strategies. Some of them might even take market statistics to optimize each deal. In order to make it possible, the agent negotiation needs to be specified as a separate entity in the agent component, making it easier to "plug" different negotiation strategies, even at runtime. This type of approach also allows the creation of negotiation processes for systems that negotiate over more than one dimension[1], as well as to create many different strategies for them.

*Multiple communication protocols.* Kasbah [Chavez and Maes 1996] agent communication protocol is fairly simple. Both buying and selling agents can make buying or selling offers basically composed of a price offer for a specific item and the answer for this offer. The answer can be of two types: Positive, in which case the deal is closed, or negative, in which case the agents keep looking for other agents and adapting their prices over time. This type of protocol is well suited for simple negotiation, but for more than one dimension and more complex strategies this protocol would probably not work, or at least it would be very inefficient. It is desirable to support a scaleable communication protocol, in which agents could support more complex protocols, and in which more elaborated proposals and counter-proposals would be possible.

Figure 3 models the last two hot-spots, in which the *doThing(), createProposal(), sendProposal(), processProposal(),* and *processAnswer()* methods are used to define the communication protocol. Whenever creating a proposal, a negotiation strategy must be selected. All of them are marked as dynamic, indicating that they may be instantiated during runtime.

[1] An example of two-dimension negotiation may be a negotiation involving price and quality of a book, in a used book marketplace.

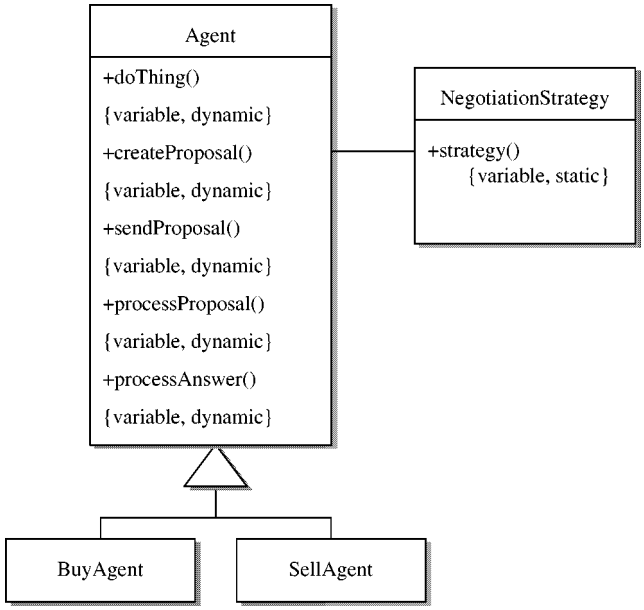


Figure 3. Negotiation strategies and communication protocols.

### *4.2. V-Market implementation*

Once the hot-spots have been precisely described in the design diagrams, they need to be implemented using current OO technology. This subsection discusses the V-Market implementation issues.

V-Market is composed of two main subsystems: a front-end and a back-end. The front-end is nor really a part of the core framework (belongs to the applications created from V-Market, and has to be defined by the framework user), but it is extremely necessary in order to allow the creation of a fully functional, complete instance. This front-end is basically responsible for the user interface, which in the case of V-Market's current implementation is composed of a set of server-side scripts that dynamically generate V-Market's web-site. This architecture is illustrated in figure 4.

The back-end subsystem is a one hundred percent Java program that can be either on the same or on a separate machine as the front-end. The back-end is basically the virtual marketplace, and everything within it. It uses two different database technologies for its persistence requirements (relational, to store the item definitions, and object-oriented, to store the agents).

The front-end communicates with the Java back-end through the use of a Java object *(Messager)* wrapped into a COM object [Rogers 1997]. This COM object serves as a unique interface to the back-end that can be addressed from within the server side script's code (JavaScript). All requests made through this *Messager* object are forwarded to a proxy object that dispatches the messages to the V-Market appropriate message handler.

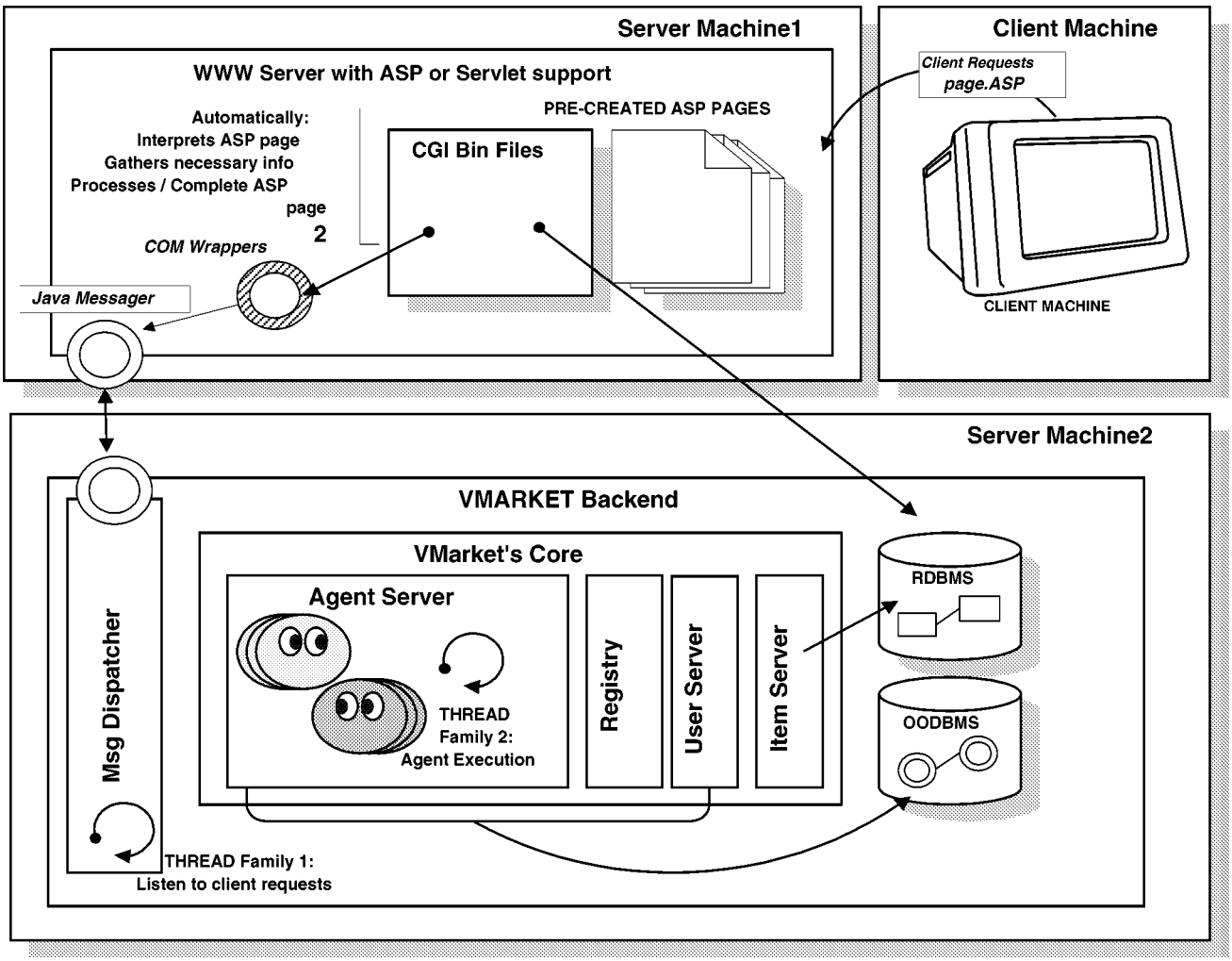


Figure 4. V-Market high-level architecture.

V-Market execution is highly concurrent, and it is composed of two families of execution threads. The first family is dedicated to executing clients' requests. These threads listen to messages sent by the front-end, and dispatch them to their respective message handlers that will then execute in individual threads. The other family of threads is responsible for the continuous execution of the marketplace itself. These threads are controlled by a concurrent agent scheduling process that assigns a fair amount of processing time to each agent.

The item hot-spot is the most important one, and should be instantiated easily. The solution adopted to implement it was based on meta-object protocols (MOPs) [Kiczales 1991]. MOPs allow meta-level concepts to be dynamically defined in terms of base-level ones. Thus, the use of MOPs is a good alternative for modeling variation points that require runtime instantiation.

In this example, a MOP was developed to allow the runtime definition of new items. This solution is shown in figure 5, in which one *Item* is defined as a list of *MetaItem* objects.

To enhance the instantiation of this hot-spot, an extra tool has been defined. It parses an XML description of the new instances and generates the HTML files that will interface with the end user and creates the new items using the MOP methods. Figure 6 illustrates the DTD used to instantiate the items.

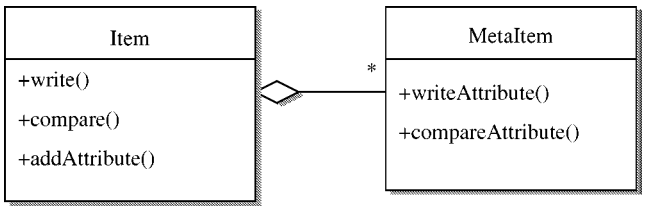


Figure 5. Item MOP.

The other two hot-spots have been modeled through the Strategy and State design patterns [Gamma *et al.* 1995], as shown in figure 7. New negotiation strategies and the protocol methods are defined in subclasses of S*trategy,* and *State*, respectively. The {incomplete, restricted} keywords, defined in [Fontoura 1999], indicate the places where new subclasses need to be added, in order complete the instantiation.

```
<!ELEMENT ITEM (NAME,DESCRIPTION, ATTRIBUTE+)>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>
<!ELEMENT VALUE (#PCDATA)>
<!ELEMENT LABLE (#PCDATA)>
<!ELEMENT DIRECTIONS (#PCDATA)>
<!ELEMENT ATTRIBUTE (NAME, LABEL, DIRECTIONS?, DESCRIPTION, PRESETS*) >
        <!ATTLIST ATTRIBUTE ATYPE (text|number) "text">
        <!ATTLIST ATTRIBUTE COMPARISON
        (equal|similar|no|numericalBigger|numericalSmaller) #REQUIRED>
        <!ATTLIST ATTRIBUTE INPUT_TYPE
        text|textarea|combobox|radio|checkbox)#REQUIRED>
        <!ATTLIST ATTRIBUTE BROWSEBLE  (yes|no)#REQUIRED>
        <!ATTLIST ATTRIBUTE REQUIRED  (yes|no)#REQUIRED>
        <!ATTLIST ATTRIBUTE SIZE CDATA "45">
<!ELEMENT PRESETS ((VALUE,LABLE)+|(VALUE)+)>
```

Figure 6. Item DTD.

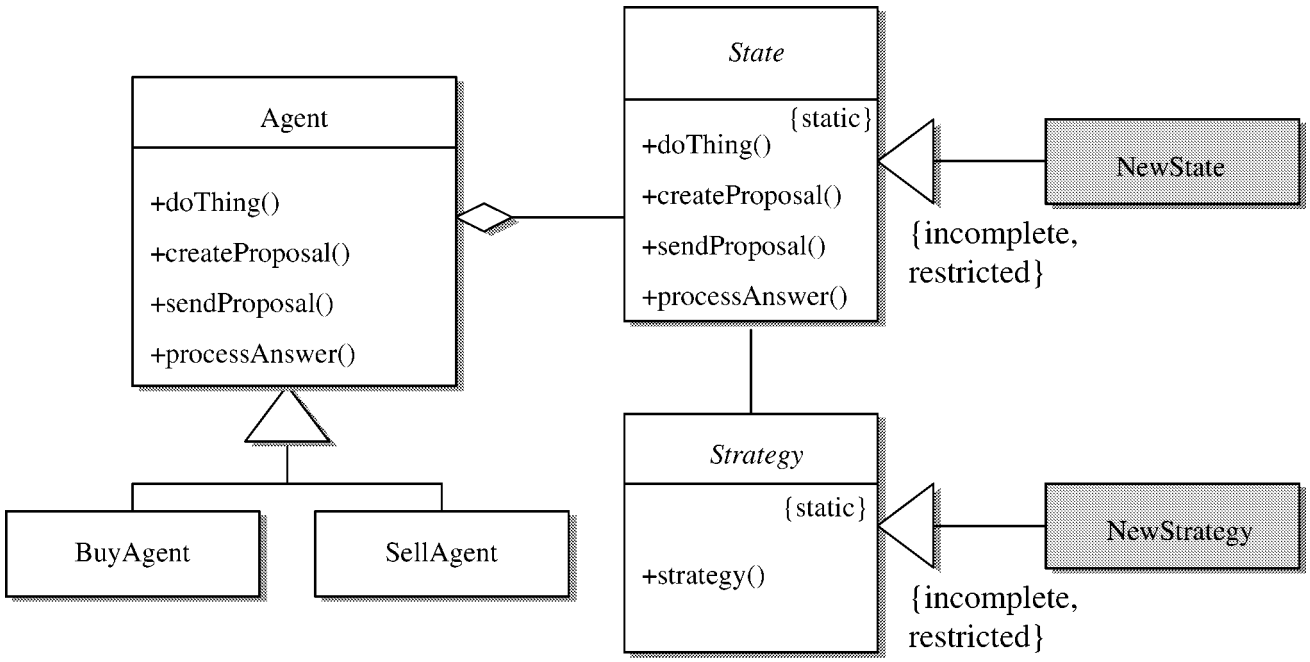


Figure 7. Using Strategy and State to implement hot-spots.

### *4.3. Other architectural issues*

Although the hot-spots described above are the most important points to be addressed by the framework, and are the essence of its flexibility, there are still some other issues that had to be addressed in the current V-Market implementation:

*Persistence and scalability.* The current implementation of Kasbah supports a very limited number of agents (no more than 1000). This limitation is due, mainly, to the fact that the system is completely centralized, and all agents interact with each other. Even agents that deal with distinct items interact with each other, making agent scheduling a process of complexity O($n^2$), where n is the number of agents in the marketplace, take, in this way, an enormous amount of time to process the entire marketplace. Also, the current database structure requires that, for each transaction, the whole database must be locked, creating a big bottleneck into the system. These are serious issues that were addressed by V-Market, by having a better structuring of the executing threads, and of the database schema used.

*Support for logging mechanisms.* One of the main purposes of this framework is to facilitate the process of creating application based on virtual marketplaces, as well as to make these applications more flexible. But the ultimate goal is to provide a tool that allows researchers to focus on the higher level aspects of agent mediation. Thus, it is fundamental that the applications created by this framework have its execution tracked very closely, so that a detailed analysis of execution can be made, and conclusions can be taken about different agent mediation roles. For this purpose, V-Market supports many types of data logging, such as the number of agents, at any given time, strategies used

by any given agent, and number of transactions closed by each agent.

*Support for reputation mechanisms.* Reputation is one of the biggest issues in e-commerce, and it becomes even a bigger issue when we are dealing with virtual marketplaces, in which every user is a potential seller or buyer of goods. There are many ways of implementing these mechanisms, and much research is going on right now on this subject. Most of the proposed solutions use some sort of reputation rating associated with each user of the virtual marketplace. These ratings are normally calculated based on his/her transaction history. Examples of these systems are Media Lab's Histos and Ebay proprietary reputation mechanism [Ebay Inc. 1995].

Hence, it is important for this framework to be flexible enough to be extended by such systems, once they become available. This was accomplished by leaving placeholders to extend user components with extra information about their reputation, and giving agents the ability to access part of this information, when making negotiation decisions.

*Support for user profiling.* One of the main strengths of software agents is their customization capability. Software agents should change their behavior depending on user preferences. In this way, many agent-based systems capture some user preference, and create user profiles based on them. Agents should use this information to customize themselves, and better serve their users. Although V-Market currently does not directly implement any user profiling facilities, the "user" component was designed with these capabilities in mind, allowing future extensions, or integration with third parties with user profiling systems, such as the Firefly's passport [Finin *et al.* 1997].

*User interface (UI) independence.* Both Kasbah [Chavez and Maes 1996] and T@T [Tete-a-Tete 1997] are divided into two main sub-systems: one back-end, in which the virtual marketplace resides and the agents execute, and one front-end, which is responsible for the user interface portion.

In Kasbah, the front-end is basically a set of CGI files responsible for dynamically generating HTML pages where most of the input and output of the system takes place. Unfortunately, there is still code responsible for the UI in the back-end, making the system's UI highly coupled with the back-end. Also, all the pages are completely hard-coded, and generated by C code, making it very hard to update any page look-and-feel. T@T, on the other hand, has an applet-based UI.

Analyzing these two systems, it became clear that our proposed V-Market framework should make a clear distinction between the UI components and the rest of the system. The UI portion should reside exclusively in a front-end subsystem (be it HTML, pager, email, or applet-based), leaving all the others virtual marketplace components on the back-end subsystem, which makes it easier to change

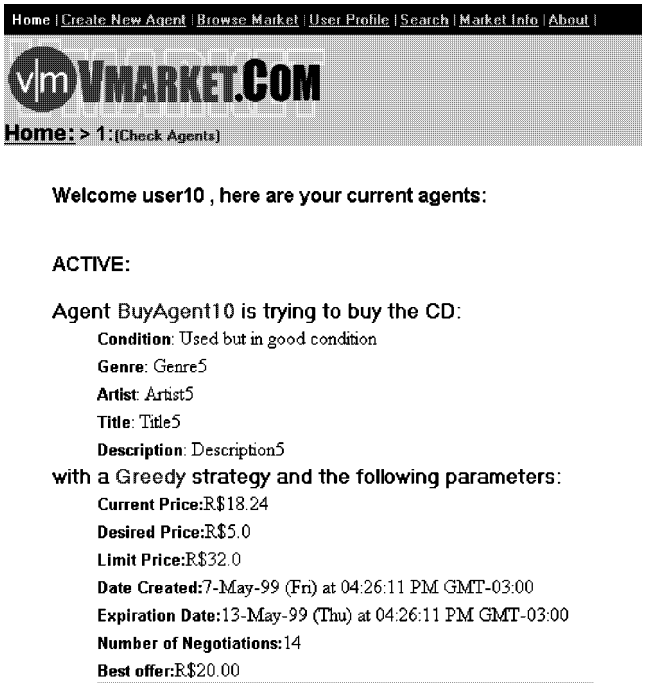


Figure 8. V-Market CD instance.

the UI components technology being used independently of the back-end subsystem. This type of flexibility allows the system to have more than one interface, such as a pager interface for monitoring agent performance, and a more detailed HTML interface to change agent control parameters.

The proposed solution for this problem was based, mainly, on two common design patterns normally used in these situations: Observer and Facade [Gamma *et al.* 1995], which were used to minimize the coupling between the front-end and back-end, define a strong interface for accessing the back-end functionality.

## 5. Conclusions and future work

This paper described the principles underlying V-Market, as well as the problems and flexibility requirements addressed by the V-Market framework. The architecture and implementation solutions adopted were also described. We have used the design solution proposed in [Fontoura 1999] to assist the explicit definition of the framework hot-spots. A more complete description of V-Market architecture, as well as the implementation approaches chosen, can be found in Ripper's [1999] M.Sc. Thesis.

Currently, we are evaluating V-Market by testing two framework instances: Books and CDs marketplaces (figure 8). Through this experience, we want to test the framework usability and performance restrictions. We are now planning to extend the system to allow mobile agents, and to allow the easy incorporation of new front-end architectures, such as pagers and smart cards. We believe that V-Market is a powerful experimentation and research tool, which allows the fast development of new robust marketplace applications in a fairly simple way.

## References


Bradshaw, J. (1997), *Software Agents*, The MIT Press, Cambridge, MA.

Chavez, A., D. Dreilinger, R. Guttman, and P. Maes (1997a), "A Real-Life Experiment in Creating an Agent Marketplace," In *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '97)*, London, UK.

Chavez, A., A. Moukas and P. Maes (1997b), "Challenger: A Multi-agent System for Distributed Resource Allocation: A Closed CPU-Time Market," In *Proceedings of the International Conference on Autonomous Agents*, Marina Del Ray, California, ACM Press.

Chavez, A. and P. Maes (1996), "Kasbah: An Agent Marketplace for Buying and Selling Goods." In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96)*, London, UK.

Chislenko, A. (1998), "Hypereconomy," Dow Jones' Markets Magazine, April.

Ebay Inc. (1995), "Personal Online Trading Community," `http://www.ebay.com.`

Finin, T., Y. Labrou, and J. Mayfield (1997), "KQML as an Agent Communication Language," In *Software Agents*, ed. J. Bradshaw, The MIT Press, Cambridge, MA.

Firefly Inc. (1996), "Firefly Network Inc.," `http://www.firefly.com.`

Fontoura, M. (1999), "A Systematic Approach for Framework Development," Ph.D. Dissertation, Computer Science Department, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), Brazil.

Gamma, E., R. Helm, R.E. Johnson, and J. Vlissides (1995), *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, MA.

Goldfarb, C. and P. Prescod (1998), *The XML Handbook*, Prentice-Hall.

Gruber, T. (1992), "Ontolingua: A Mechanism to Support Portable Ontologies," Technical Report KSL-91-66, Knowledge Systems Laboratory, Stanford University, CA.

Guttman, R. and P. Maes (1998a), "Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce," In *Proceedings of the Second International Workshop on Cooperative Information Agents (CIA '98)*, Paris, France.

Guttman, R. and P. Maes (1998b), "Agent-mediated Integrative Negotiation for Retail Electronic Commerce," In *Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET '98)*.

Guttman, R., P. Maes, A. Chavez, and D. Dreilinger (1997), "Results from a Multi-Agent Electronic Marketplace Experiment," In *Proceedings of Modeling Autonomous Agents in a Multi-Agent World (MAAMAW '97)*, Ronneby, Sweden.

Guttman, R., A. Moukas, and P. Maes (1998), "Agent-mediated Electronic Commerce: A Survey," Knowledge Engineering Review.

Kiczales, G., J. des Rivieres, and D. Bobrow (1991), *The Art of Meta-object Protocol*, The MIT Press, Cambridge, MA.

Leebaert, D. (1998), *The Future of the Electronic Marketplace*, The MIT Press, Cambridge, MA.

Media Lab Software Agents Group (1996), "AmEC Infrastructure Project," `http://ecommerce.media.mit.edu/Infrastructure/infra.html.`

Moukas, A., R. Guttman, and P. Maes (1998), "Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective," In *Proceedings of the First International Conference on Electronic Commerce* (ICEC '98), Seoul, Korea.

Personalogic (1997), "Personalogic," `http://www.personalogic.com.`

Pree, W. (1995), *Design Patterns for Object-Oriented Software Development*, Addison-Wesley, Reading, MA.

Ripper, P. (1999), "V-Market: A Framework for Agent Mediated E-Commerce Systems based on Virtual Marketplaces," MS Thesis, Computer Science Department, PUC-Rio, Brazil.

Rogers, D. (1997), *Inside COM: Microsoft's Component Object Model*, Microsoft Press.

Terpsidis, J., A. Moukas, B. Pergioudakis, G. Doukidis, and P. Maes (1997), "The Potential of Electronic Commerce in Re-engineering Consumer-Retailer Relationships," In *Proceedings of the European Conference on MM & E-Commerce*, Florence, Italy.

Tete-a-Tete (1997), "Fixing Online Shopping," `http://ecommerce.media.mit.edu/Tete-a-Tete/.`