


7

Kasbah@MIT: A Real-World Agent Mediated Electronic Marketplace

by

Keith D. Smith

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

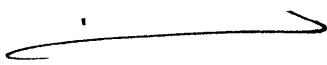

May 18, 1998


Copyright 1998 Keith D. Smith. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

7

Author _____
Department of Electrical Engineering and Computer Science
May 18, 1998

Certified by _____

Pattie Maes
Associate Professor, MIT Media Laboratory
Thesis Supervisor

Accepted by _____

Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Eng.

**Kasbah@MIT:
A Real-World Agent-Mediated Electronic Marketplace**

by

Keith D. Smith

May 18, 1998

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The Kasbah@MIT system provides a robust, secure, online medium for buyers and sellers to transact goods using customizable software agents. As a multi-agent electronic marketplace, Kasbah@MIT allows buying and selling agents to negotiate and close deals on behalf of their owners. After being configured, a buying or selling agent searches the marketplace for other agents selling or buying the item it was created to buy or sell, respectively. An agent notifies its creator of its progress via personalized accounts and electronic mail and can be reconfigured at any time. The Kasbah system also provides users with real-time market information to assist them in customizing the behavior of their agents. Kasbah@MIT automates the entire process of locating, buying, and selling of goods among many individuals thereby reducing their respective workloads.

Thesis Supervisor: Pattie Maes
Title: Associate Professor, MIT Media Laboratory

Acknowledgments

I dedicate this thesis to my dear mother, Lesia Smith, whom I never had the opportunity to thank for her love and nurturing. She taught me about sacrifice and diligence without ever mumbling a word. I would also like to thank my wonderful family whose love and support have been beacons throughout all my endeavors; especially my older brother, Kevin Smith, who is my biggest fan and role model.

I am eternally indebted to the Brothers of the Rho Nu Chapter of Alpha Phi Alpha Fraternity, Inc. for my unconquerable soul.

I would like to thank the gentlemen of Chocolate City at MIT for being my surrogate family these past five tumultuous years and giving me a true home at MIT.

I also would like to thank my girlfriend of three years, Latosha Byrd, for always believing in me even when I had doubts. I did not think I would ever finish and you assured me otherwise. As usual, you were right.

Finally, I would like to thank my associates and fellow researchers in the Software Agents Group at the MIT Media Laboratory. Your guidance and expertise were invaluable to my growth and development.

Contents

Introduction	7
Electronic Commerce Today	7
The Role of Software Agents	8
Motivation for this Thesis	10
Overview of this Thesis	12
Functional Overview	14
Using Agents to Transact Goods	14
Creating an Account	15
Creating an Agent	18
Item Description	18
Configuring the Agent	23
Agent Strategy	28
Modifying an Agent	33
User Reputation	34
Agent-to-User Communication	38
Find Agents	46
Related Work	49
Overview	49
BargainFinder	50
FireFly	51
Fido	51
United Computer Exchange	52
ONSALE	53
Yahoo! Classifieds	54
Classifieds2000	55
Ebay	55
Previous Kasbah Experiments	56
Summary	57
System Architecture	59
Overview	59
Web Server	61
Marketplace Server	64
Mail Server	70
RootOperations	71

<u>Discussion of Experiment</u>	73
Overview	73
Quantitative Analysis	74
Qualitative Analysis	76
<u>Future Work and Conclusions</u>	79
Future Work	79
Conclusion	82
<u>References</u>	84

List of Figures

Figure 1. A User's Home Section	17
Figure 2. Item Description (Seller)	20
Figure 3. Item Description (Buyer)	21
Figure 4. Agent Configuration	26
Figure 5. Price Negotiation Equations	31
Figure 6. (a) Anxious Buyer, (b) Moderately-Anxious Buyer, (c) Frugal Buyer	32
Figure 7. Agent Modification	34
Figure 8. Reputation Equation	36
Figure 9. Prospective Buyer Found Message	41
Figure 10. Agent Status Report	42
Figure 11. Deal Made Message	44
Figure 12. Rate Other Party	45
Figure 13. Item Found Message	47
Figure 14. System Interconnections	60
Figure 15. Marketplace Server Decomposition	65
Figure 16. RootOperations Applet Interface	72
Figure 17. RootOperations Command Line Interface	72
Figure 18. Book Agent Statistics	75
Figure 19. Music Agent Statistics	75
Figure 20. Transaction Statistics	75

Introduction

Electronic Commerce Today

The Internet has long been regarded as a medium for communication and the transmission of information. In addition to providing us with almost instant access to volumes of information, the Internet facilitates searching for resources and products of interest. As computers have continued to play more important roles in our business lives, we have come to rely on them to augment our personal lives as well through electronic mail, newsgroups, and browsing. Computers are no longer viewed merely as tools for increasing our productivity at work but as tools for increasing our productivity period.

The concept of using computers to assist users in making shopping-related decisions is relatively new but has recently received a tremendous amount of attention. Electronic commerce is an effort to supplement the traditional methods used to transact goods with the speed and ubiquity of the Internet. More precisely, Electronic commerce is geared toward the buying and selling of goods and services via the Internet.

Traditionally, people employ a number of conventional methods to transact goods directly—without an intervening third party (e.g., a

commercial store)—among each other. These methods include posting flyers to attract prospective buyers, advertising in the classifieds section of the local newspaper, having garage sales, etc. The success of the Internet, specifically the World Wide Web, has provided a new medium for people to transact goods. Today there are many online sites that aid people in buying and selling a plethora of merchandise. These range from online malls and concierges to mail order and classifieds sites.

By the year 2000 it is estimated that Internet-based sales will reach three hundred billion dollars [Alba96]. As a result, many companies have begun to divert resources toward establishing a presence on the World Wide Web which normally consists of erecting massive Web sites and advertising their Web addresses on television, billboards, radio, etc. Our focus, in this thesis, is on consumer-to-consumer transactions however vendor-to-vendor and vendor-to-consumer models are easily extracted.

The Role of Software Agents

The intelligent agent is a concept that has been around for nearly twenty-five years. Descriptions of software agents range from macros that a user can manipulate by adjusting its parameters to

intelligent agents that have the ability to learn and have some degree of artificial intelligence [Shneiderman97]. Today, most software agents fall closer to the former description than the latter. All agents have some similar characteristics in that they are all software entities with some degree of autonomy, designed to carry out operations on behalf of the user and, in this process, partially fulfill the user's goals. Agents are a needed entity due to the exponential growth in information available on the Internet. With the constant addition of information to the Internet, it is impossible to process the data manually causing users to suffer information overload.

Software agents help people with time-consuming or repetitive activities [Maes94]. Today, there are agents that save users time by performing laborious tasks such as gathering and posting electronic mail and checking newsgroups. The user does not need to directly manipulate the agents and can compose electronic mail and newsgroup postings normally. These types of agents are relatively simple and are essentially scripts that adhere to strict rules. Agents that search for information on behalf of a user allow true multitasking by freeing the user to do other tasks as they retrieve the requested information.

Currently, to locate information on the Web, one uses a search engine such as *Yahoo!*, *Infoseek*, or *Alta Vista*. One could expend invaluable minutes, hours, and sometimes days wading through the results of such searches for the desired information. With the help of an intelligent software agent, this wasted time could be put to better use. After configuring the agent with the appropriate parameters, one could continue working on other projects or complete more interesting tasks.

Motivation for this Thesis

Making informed buying and selling decisions are seemingly insurmountable tasks given the magnitude of information available to the consumer. Intelligent software agents capable of making decisions on behalf of their owners could be used as the first line of defense against information overload. The primary motivation for our work is the realization that consumers squander an inordinate amount of time transacting goods directly with other consumers. The following is a list of the six stages of consumer buying behavior [Guttman97]:

1. Need Identification
2. Product Identification and Comparison
3. Merchant Identification and Comparison
4. Negotiation
5. Payment and Delivery

6. Product Usage and Service

In the “Need Identification” stage, consumers become aware of an unmet need and begin to search for solutions. The “Product Identification and Comparison” stage consists of gathering information to help distinguish between similar products that the consumer believes will meet her needs. Similarly, the “Merchant Identification and Comparison” stage is the point at which a consumer compares and contrasts the vendors carrying the product using consumer-selected criteria (e.g., price, warranty options, support, etc.) The “Negotiation” stage characterizes the consumer and merchant finalizing the terms of the transaction and has market-dependent complexity and duration. The “Payment and Delivery” stage marks the termination of negotiation but could also influence and be influenced by previous stages. For example, the consumer may draw different conclusions in the “Merchant Identification and Comparison” stage if the only payment option for a particular merchant is cash. Finally, the “Product Usage and Service” stage is when the consumer evaluates her satisfaction with the product and merchant.

Each stage adds overhead to the buying process since they all require intervention on the part of the buyer or seller or both.

Intelligent software agents can increase the productivity of both buyers and sellers by automating some of the stages outlined above. The concept of using software agents to automate stages of consumer buying behavior is not new however the systems having this functionality tend to only automate the “Merchant Identification and Comparison” stage with price as the sole comparison criterion. The Kasbah@MIT system automates the “Negotiation” stage in addition to the “Merchant Identification and Comparison” stage.

The Kasbah@MIT system has many predecessors but none of them were used in a real-world experiment over a substantial period of time. As a result, the previous systems did not have many security safeguards expected in online systems today. Similar to its precursors, the Kasbah@MIT system consists of an agent marketplace with a Web front-end. We conducted an experiment with the entire MIT community using the Kasbah@MIT system that has spanned several months. The results of which are presented herein.

Overview of this Thesis

This thesis is organized as follows. In Chapter 2 we introduce the concept of using software agents to assist in good transaction and outlines the individual processes involved with using the

Kasbah@MIT. Chapter 3 discusses related work focusing on existing online systems. In Chapter 4 we highlight the architecture and implementation of the Kasbah@MIT system. In Chapter 5 we present the results of an experiment involving the Kasbah@MIT which spanned several months. In Chapter 6 we propose future extensions for the Kasbah@MIT system and provide a brief conclusion.

Functional Overview

The Kasbah@MIT system is accessed via the World Wide Web. Members are able to use the system to buy and sell books and music. The system was designed to focus on consumer-to-consumer transactions but could easily be used to support vendor-to-consumer transactions as well. We consider consumer-to-consumer transactions those that do not rely on a third party or broker to act on behalf of either the buyer or seller. This is similar to buying or selling a home without hiring the services of real estate agents. This chapter highlights the key areas of functionality of the Kasbah@MIT system.

Using Agents to Transact Goods

Users of the Kasbah@MIT system buy and sell goods by creating software agents. These agents could be considered autonomous since, after creation, they are able to function with very little user intervention. Agents relieve users of many of the laborious tasks associated with good transactions. These include locating potential buyers (or sellers) of the offered (or desired) good, applying user-specified rules to ascertain the ability of the other party to meet the user's needs, and negotiating on behalf of the user. Thus the agents

are personal assistants that collaborate with the user rather than embellished scripts that require direct manipulation [Maes94]. In order to fulfill these requirements the agents must be configured with “smarts” or data that allow them to procedurally make decisions and behave according to the user’s wishes.

There is a one-to-many mapping of users to agents and a one-to-one mapping of a single agent to goods. That is users can own multiple agents but an agent transacts a single good—multiple agents may transact the same good. The agents act on behalf of their creators and each attempts to maximize its utility on behalf of its owner. In order for the agents to interact, they must share some common location or method of locating each other. This location can be considered, in the abstract, a marketplace. Consequently, Kasbah@MIT is an online, multi-agent marketplace where agents can interact on behalf of their owners.

Creating an Account

Before one can access the Kasbah@MIT system he must become a member. The only membership requirement is that the person have access to a valid electronic mail account. To join, a prospective user need only visit the site and click the appropriate link. The user

completes an online form requesting basic optional information (e.g., name, location, etc.) and the user's electronic mail address. Once this information is submitted, an account is created for that user and a temporary password is assigned. To prevent users from maliciously signing up unsuspecting individuals or supplying fictitious electronic mail addresses the temporary password is sent to the electronic mail address provided by the user. Once the user receives the introductory message containing her temporary password, she can log into the system.

Once logged in, the user can complete a variety of tasks. We suggest users first change their temporary password. Each user is brought to her specific home section which contains important messages from the Kasbah@MIT system or her active agents. Every feature of the system is accessible from the home section. **Figure 1** shows a particular user's home section. The following is a list of tasks available to a user in the home section:

- Create a new agent
- Reconfigure/Terminate an existing agent
- Browse merchandise currently offered in the marketplace
- Search for a specific item in the marketplace
- View statistical data on the marketplace
- Change user information (e.g., password, name, location)
- Send feedback to the developers

- View reputation profile
- Get help on navigating the site
- Terminate current session

Each of these tasks will be explained in greater detail in following sections.

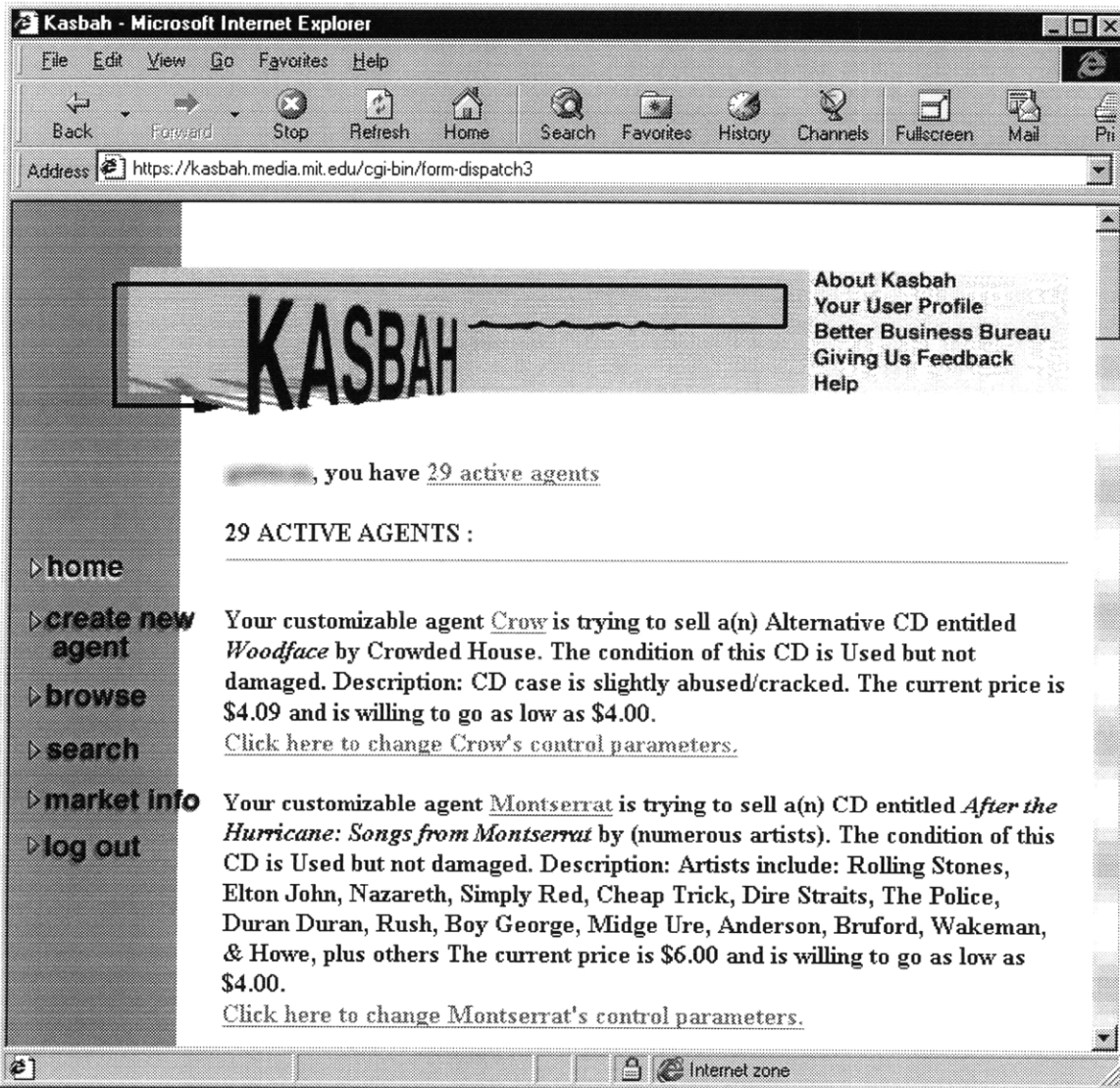


Figure 1. A User's Home Section

Creating an Agent

Users can create agents to buy or sell a particular good. Additionally, if a user desires to be notified when a good becomes available for sale or is desired for purchase in the marketplace, he can create a Find Agent. In this section we describe the agent creation process for a user wishing to buy or sell an item. The functionality of Find Agents will be highlighted later. The role of the agent is determined by whether the user wants to buy or sell an item. Since buying and selling are symmetrical the agent creation processes for the two types are fundamentally the same.

The agent creation process consists of two stages. The first stage is a description of the good one desires to buy or sell. The second is the configuration of the agent to enable it to act on the user's behalf by supplying the necessary parameters and behaviors.

Item Description

Describing the good one wishes to buy or sell is probably the most important stage of the agent creation process. This description is used to locate potential buyers or sellers of the particular good the agent will be selling or buying.

The Kasbah@MIT system is limited to transacting books and music to avoid the problems of naming and classifying that would result from attempting to handle arbitrary items spanning disparate categories. Since we rely on the computer's ability to match buyers and sellers, descriptions must be explicit and accurate. Inadequate product description rules could lead to the erroneous pairing of agents that have different goods or failure to pair agents with the same good but differing descriptions of the good.

Figure 2 is a typical screen seen by a user wishing to sell a particular book. The information the user must provide includes:

- The type of book (e.g., hardback or paperback). This field is required and defaults to "Paperback."
- The book's genre (e.g., fiction, health, history, poetry, etc.) This is a required field but is "Don't Know/Unspecified" by default.
- The book's title. This field is required and must be provided by the user.
- The name of the author. This field is required and must be provided by the user.
- The edition of the book. This field is optional.
- The number of the MIT course which uses this book. This field is optional and is useful for selling textbooks.
- The condition of the book. This field is required and defaults to "Used but not damaged."
- An area for additional comments is provided for users to include more elaborate or thorough descriptions of the item being sold. For instance: "This is one of ten copies still in existence."

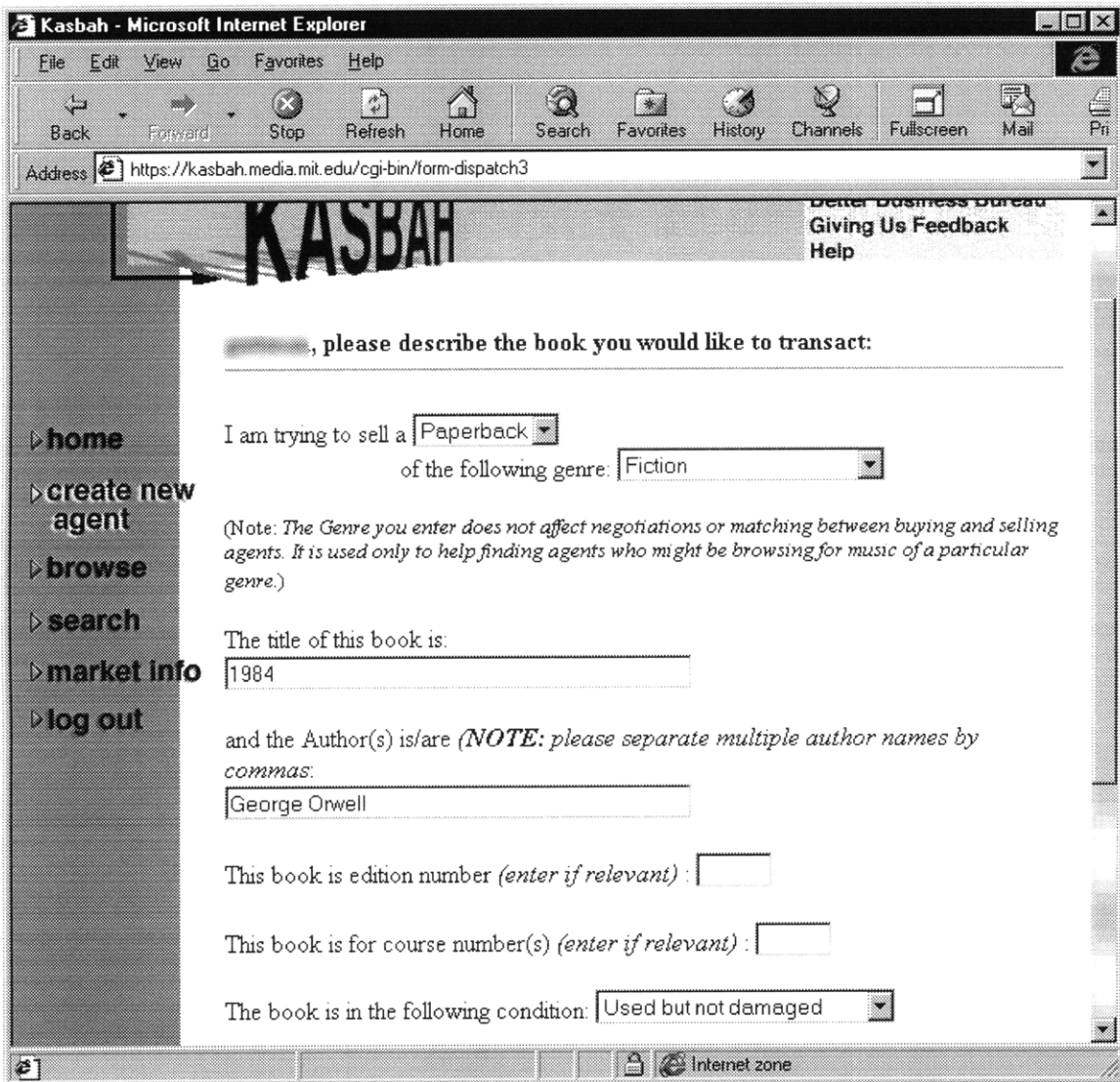


Figure 2. Item Description (Seller)

For example, if a user has an extra copy of George Orwell's *1984* they want to sell, she would simply enter the information as shown in **Figure 2**. Another user wishes to purchase *1984* for a friend. This

user creates a buying agent by completing the item descriptions as shown in **Figure 3**.

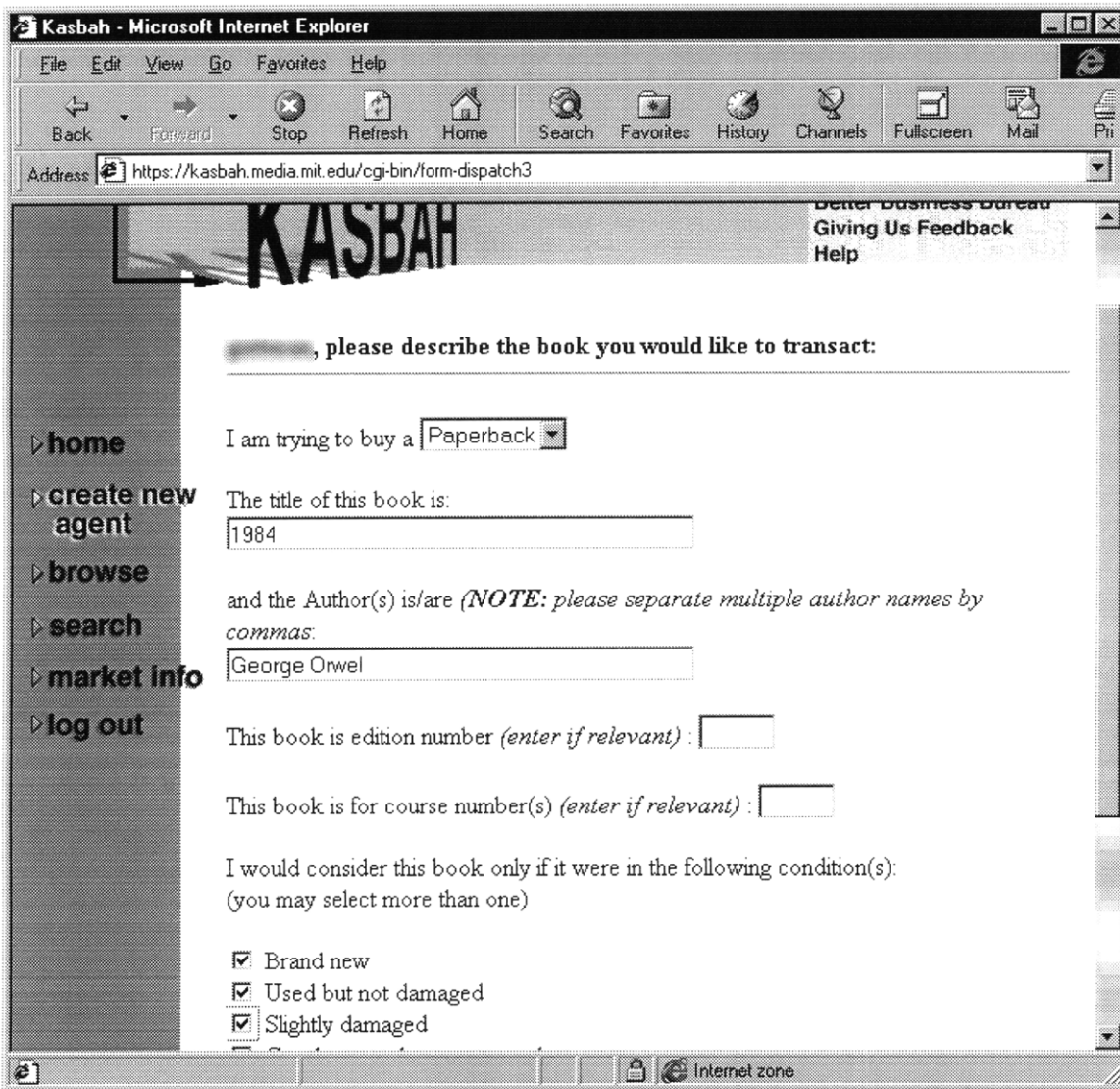


Figure 3. Item Description (Buyer)

Although the author's name is slightly misspelled by the prospective buyer, it is evident that the user wishes to purchase the book being

advertised by the seller. To accommodate typographical mistakes of this sort, the Kasbah@MIT system uses a longest-common-string syntactic pattern-matching algorithm which provides some leniency for misspelled or misarranged names and titles. As humans we take such decision-making ability for granted. Consequently, it is imperative that agents that are to act on our behalf be given similar capabilities. As long as two strings are above a sixty-six percent matching threshold, the system will assume they are the same. This number was experimentally determined to provide the best results. We did not want to increase the frequency of false positives (i.e., two strings are said to match although they do not) at the expense of eliminating false negatives (i.e., two strings are said to not match although they do.) Admittedly, this is not an elegant solution however formally solving this problem would force one to delve into the field of artificial intelligence more than is desired in this system.

The descriptions for music items are similar to their book counterparts. The information a user desiring to sell a music item must provide includes:

- The recording medium (e.g., compact disc, DAT, tape, etc.). This field is required and defaults to “CD.”
- The genre of the music (e.g., hip-hop, blues, classical, etc.) This is a required field but is “Don’t Know/Unspecified” by default. A

genre of “Don’t Know/Unspecified” will behave as a wildcard such that it will match with any other genre.

- The title of the recording. This field is required and must be provided by the user.
- The name of the artist. This field is required and must be provided by the user.
- The condition of the recording. This field is required and defaults to “Used but not damaged.”

An area for additional comments is provided for users to include more elaborate or thorough descriptions of the book or music item being sold. For instance: “Includes duets with Barbra Streisand and Pattie LaBelle.”

Configuring the Agent

After describing the good he is interested in buying or selling, a user enters the second stage of agent creation—agent configuration. This is when the user determines the type of agent that will best meet his needs and provides the agent with the necessary parameters and data for the agent to have some semblance of autonomy. Computers can only do what they are told and this stage is where the user makes his requirements known to his agent. We are assuming that agents know what it means to negotiate but, in fact, this too has been specified procedurally as described below. Given this negotiation ability, agents need only be configured with a few additional

parameters that provide their negotiation procedures with the data required to actually begin negotiating.

Before a user can begin to configure an agent she must decide what type of agent will best meet her needs. A *simple* agent is one that maintains a fixed asking/offering price over its lifetime. A *customizable* agent is one that will change its price over its lifetime—in a user-instructed manner—to entice buyers. Simple agents are nothing more than automated classified advertisements that respond to interested parties on behalf of their owners. However, customizable agents are aggressive buyers/sellers with dynamic behavior and the ability to accommodate changing marketplace conditions. If there are no buyers for an item being sold, customizable agents attempt to attract buyers by reducing the asking price.

The Kasbah@MIT system views agents as self-contained objects with capabilities, strategies, and utility functions that are not visible from the outside world directly. These objects can be queried and manipulated through a standard interface. The user-provided data are the basis of the agent's capabilities and strategy. The agent's interface is what allows it to interact with other agents in the marketplace and with the marketplace as well. The following is a list of parameters a

user is able to provide to a selling agent (buying agent parameters are complementary):

- **Name.** Agents are given names to make it easy to distinguish between them in later messages and interactions.
- **Deadline.** The date this agent should stop attempting to make a transaction. The agent will stop looking for possible buyers on this date.
- **Desired Price.** The amount the user wants for the item being sold. The agent will attempt to find buyers willing to pay this price for the item.
- **Lowest Price.** The lowest price the user will accept for the item. The agent will not sell the item for less than this amount.
- **The negotiation strategy.** These determine whether the agent is anxious, moderately anxious, or patient in adjusting the price to attract potential buyers (as described below.)
- **Reputation cutoff.** This specifies the lowest possible reputation of any agent (actually the agent's owner) the agent should negotiate with. For instance, if the agent should only negotiate with agents of users with reputations of average or better, the cutoff should be set to "Average." Reputations are described in more detail below.

As shown in **Figure 4**, the user simply completes an online form to configure her agent.

Kasbah - Microsoft Internet Explorer

File Edit View Go Favorites Help

Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print

Address <https://kasbah.media.mit.edu/cgi-bin/form-dispatch3>

Customizable Agent Control Parameters

▶ **home**

▶ **create new agent**

▶ **browse**

▶ **search**

▶ **market info**

▶ **log out**

What would you like to name your agent? *(remember to make it different from your other agents' names)*

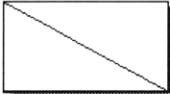
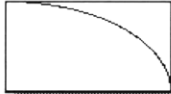
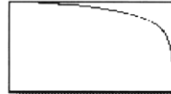
I would like to sell this good by:

April 9 1998 by 1 53 am

My desired price is US\$ 00

but the lowest possible price I am willing to sell for is US\$ 00

I would like to use the following kind of price decay function:

\$  Time
 \$  Time
 \$  Time

I would like my agent to send me e-mail notifications in addition to posting messages for me in the Kasbah site:

Yes No

I would like my agent to give me a status report of its activity:

Whenever it feels something important has happened

Every hours

I would like my agent to negotiate within MIT only

I would like my agent to negotiate with agents who have a reputation rating greater than Average

Internet zone

Figure 4. Agent Configuration

This high-level interface relieves users of the responsibility of tinkering with the low-level details and code that drive their agents.

To make the system attractive to users with varying levels of computer experience, it is imperative that the interface for controlling their agents be simple yet powerful. The interface we present shields the novice user by abstracting away implementation details and being absent of technical rhetoric. We also provide graphical depictions of the different options whenever possible. However, we do not attempt to “dumb-down” the interface with useless embellishments or an overabundance of wizard-like behavior. Doing so would be to the chagrin of veteran users. It was our intention to have agent configuration proceed like a conversation with a friend. Imagine discussing your plans to purchase a used Al Green compact disc with your friend, Greg. It could go as follows:

You: “Greg, I’m considering buying this Al Green CD entitled “Al Green’s Greatest Hits.”

Greg: “Really? How much are you willing to pay?”

You: “Well, I can only afford to pay \$7 but, since it’s a classic, I’m willing to pay as much as \$9.”

Greg: “You’re right. That CD is a classic. The only way you will be able to buy it for \$7 is if it’s in poor condition.”

You: “That’s true. I really want to have it before we leave for Spring Break in three weeks. I hope one becomes available.”

Greg: “Good luck.”

Decomposing this conversation into its important parts we would gather the following agent configuration data:

Deadline: A maximum of three weeks from today.

Desired Price: \$7
Highest Price: \$9
Negotiation Strategy: Anxious

It should be noted that the condition of the CD is part of the item description and would be used as part of the item description stage.

Agent Strategy

Customizable agents, as mentioned before, aggressively seek to complete a transaction on behalf of their owners. Whether buying or selling, customizable agents adhere to their agendas until they consummate deals or reach their deadlines, whichever comes first. In this section, we describe the customizable agent negotiation strategy from a buying agent's perspective. Obviously, the good type (i.e., book or music) and agent role (i.e., buying or selling) are inconsequential and strategies of each type should be easily extracted.

Upon creation, buying agents comb the marketplace for sellers of the item they desire to buy. In the best case, a buying agent will immediately find a selling agent offering the good it wishes to purchase at the price—or a lower price—than it wants to pay. In this scenario, the agent would notify its owner of the details of the potential deal and waits to receive permission to make the deal. A selling agent will always accept a price greater than or equal to its asking price.

Conversely, a buying agent will always agree to pay a price lower than or equal to its offer price. Once its owner gives the agent permission, the deal, if it is still available, is made. The decision to have the agent wait to receive the owner's permission before making a deal was a design decision for handling the possibility of inexact item descriptions. To humans, the words "physics" and "psychics" are completely different but computers view them as syntactically similar. If the agent finds an agent selling the good but at a price greater than its upper bound, it will incrementally raise its offer price over its lifetime up to its upper bound. If the seller's price is still too high by the buying agent's deadline, the buying agent will report its inability to make a deal to its owner and become inactive. We decided to have agents become inactive rather than terminating them in case their owner's wish to give them more time thus saving the time needed to create new agents to sell the same good. In the worst case, there would be no agent selling the item the buying agent wishes to purchase. The buying agent will have no choice but to wait until the item becomes available or it reaches its deadline. In every case, agents are always informed whenever a new agent is added to the marketplace. Therefore, if the buying agent is currently negotiating

with a selling agent whose price is too high when another agent, selling the same item at a price within the buying agent's range, becomes active, the buying agent will make the deal with the new agent—just as a human would.

As mentioned before, agents incrementally increase their asking price, in the case of buyers, or decrease their asking price, in the case of sellers according to a user-specified strategy. Agents are not optimal in that they are not guaranteed to get the best price for a particular transaction. Instead, agents are opportunistic in that once they recognize a deal matching their criteria they will make it. In this regard, agents simply mimic human buyers and sellers. For instance, if a person places an advertisement in the local newspaper to sell a car for \$1,906.00 he intends to sell it to the first person willing to pay that amount. He would not turn away prospective buyers willing meet his price with hopes of someone calling later willing to pay \$1,989.00.

One may immediately assume that this puts sellers at risk since potential buyers could simply wait until the agent reduces its price before making an offer. However, this situation is unlikely since buyers are not aware of the agent's deadline, lowest price, and, in most cases, desired price (since the agent could have been created sometime

before a buyer became interested in the item being sold.) Furthermore, both buying and selling agents are oblivious to the type of negotiation strategy being used by other agents.

Figure 5 shows the equations governing the negotiation strategies used by buying and selling agents:

Selling Agents : $P(t) = P_D - k(P_D - P_F)$	Buying Agents : $P(t) = P_D + k(P_F - P_D)$
---	--

where:

$P(t)$	The current asking/ offering price of the selling/buying agent as a function of time.
P_D	The desired price
P_F	The lowest/highest price the selling/buying agent is willing to offer
$k = \left(\frac{t}{t_f - t_0} \right)^\alpha$	The scaling factor t The current time t_f The deadline t_0 The start time α 1 = anxious, 2 = moderately anxious, 3 = patient

Figure 5. Price Negotiation Equations

These equations dictate the dynamic nature of the agents' pricing strategies. Of most significance is the scaling factor, k , which is composed of the current time, start time, deadline, and a degree. The scaling factor plays a more and more significant role in the current

price as the deadline nears—the numerator grows larger as the denominator stays the same. The degree, α , encapsulates the type of strategy being used. Anxious buyers and sellers will increase/decrease the offer price by fixed amounts in fixed intervals of time. This describes a linear negotiation function. A more moderate approach would be to gradually increase/decrease the price near the beginning, but as the end gets nearer, to increase/decrease the price more substantially. This describes a quadratic negotiation function. Finally, a more patient buyer or seller would wait until the absolute latest moment to start raising/slashing its price to maximize its potential of getting a hefty savings/profit early in the negotiation process. A cubic negotiation function has this behavior. **Figure 6(a)-(c)** gives a pictorial example of a selling agent's offer price versus time with various values of α . Notice P_D is the leftmost price and P_F is the rightmost price in each graph.

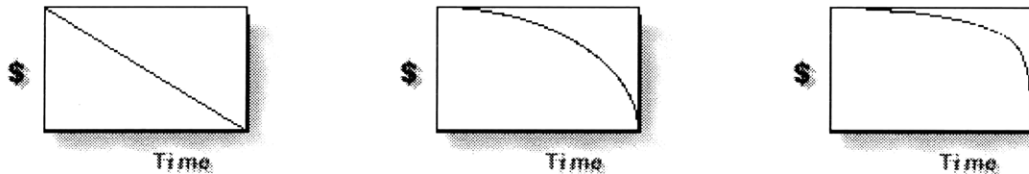


Figure 6. (a) Anxious Buyer, (b) Moderately-Anxious Buyer, (c) Frugal Buyer

Modifying an Agent

An agent's owner can, at anytime, modify the agent's parameters or terminate the agent. The latter feature is useful if the item was bought or sold using a different method (e.g., word-of-mouth.) The former feature is geared more toward power users who desire more complex behavior or negotiation strategies for their agents. For instance, a user who wants to have the number of other agents in the marketplace buying/selling the same item factor into his agents negotiation strategy currently has no way of expressing this to the agent. The user would need to manually monitor the marketplace and reconfigure his agent(s) as he deems necessary. This type of behavior removes a great deal of autonomy from the agents themselves. We intentionally did not include more complex agent negotiation techniques so users would be able to interpret and forecast their agents' actions completely. Predictability of agent behavior is proving critical for real-world success and user acceptability of agents [Shneiderman97]. To modify an agent's parameters, a user simply clicks the agent's name in the home section. **Figure 7** shows the screen resulting from this action. Notice the concise summary of the agent's current parameters.

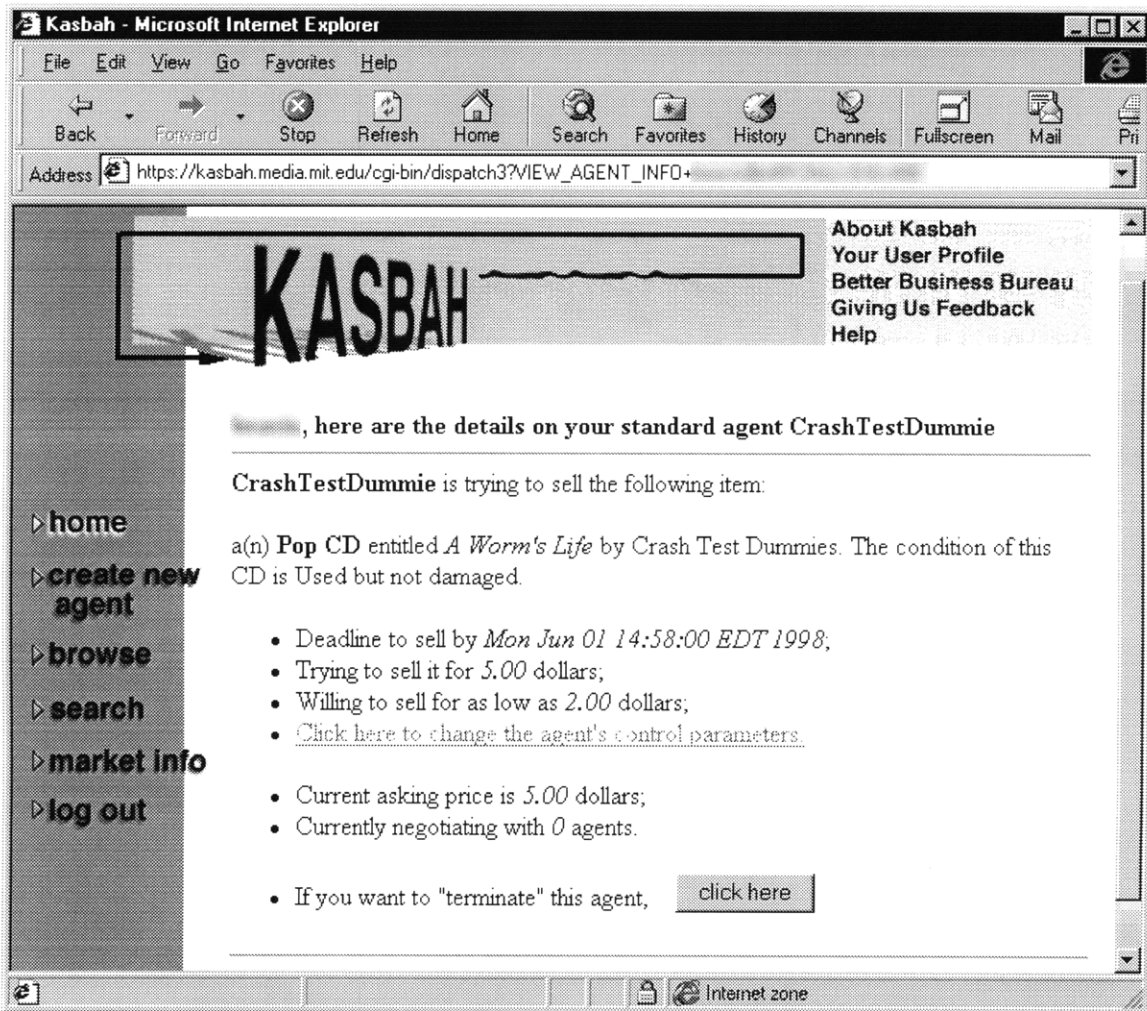


Figure 7. Agent Modification

User Reputation

Trust, security, and privacy are all key issues in electronic commerce. If a system's security and privacy mechanisms are inadequate, many users will not use it. The Kasbah@MIT system provides security and privacy by using Secure Sockets Layer

encryption, a protocol developed by Netscape Corporation for transmitting private documents via the Internet, for all transactions between the user's Web browser and the Kasbah@MIT servers. No user information is transmitted in plain text and all user information and agent data are kept on a secure server in a secure location.

Humans develop trust over periods of time. There is no reason for a person to trust a system or another person to which she has had little or no prior exposure. Instead, people rely on time-proven honesty and integrity to separate those who are trustworthy and those who are not.

The Kasbah@MIT system takes a similar approach to developing trust among its users. Similar to its real-world namesake, the Kasbah@MIT system's Better Business Bureau is responsible for storing and reporting individuals' ratings to others considering doing business with them. A user's rating or reputation is part of her profile and cannot be modified directly by the user.

Every user starts with a rating of average. At the end of every transaction, both parties are given the opportunity to rate the other on a scale ranging from "horrible" to "great." These ratings are purely subjective and can be affected by a number of factors including:

- Did the seller describe the item correctly (e.g., was the item used but described as being in mint condition)?
- Was the other party prompt in making the exchange of the good for payment?
- Did the other party try to change the agreed upon price/offer during the actual exchange?

Ratings range from poor to great and have numerical values of zero to five, respectively—an average rating has a numerical value of three. As illustrated in **Figure 8**, a person's reputation is the running average of all previous ratings given to him by other users following transactions.

$$R(u) = \frac{\sum_{i=1}^n R'(u, v_i, i)}{n}$$

where :

$R(u)$ = The reputation of user, u

$R'(x, y, z)$ = The rating of user x by user y in transaction z

v_i = The user involved in transaction i with u

n = The number of transactions involving user u

Figure 8. Reputation Equation

Currently there is nothing to prevent a user with a poor rating from creating a new account (presumably with a different username.) However, the more transactions a person makes that result in positive ratings the more apt other individuals will be to have their agents negotiate with that person. In a sense, one's reputation becomes his

online moniker that could attract or repel potential buyers and sellers. Another potential weakness of the current Better Business Bureau is a person receives the same reputation as both a buyer and a seller. As a result, a malicious user could be an excellent buyer but a horrible seller and still retain an average rating. Due to the low cost of the goods transacted via the Kasbah@MIT system we believe the current incarnation of the Better Business Bureau to be sufficient. If more valuable goods were transacted, the current implementation would not adequately safeguard users and could have devastating results.

A user can view her current rating by clicking the “Better Business Bureau” link in her home section. However, there is no method for a user to determine the sources of each individual ratings. Agents inherit the reputations of their owners. As mentioned before, part of every agent’s configuration is the reputation cutoff. This number represents the lowest acceptable rating an agent must have for this agent to consider negotiating with it. As an example, if agent “A” wishes to negotiate with agent “B” but agent “A’s” reputation falls below agent “B’s” cutoff, agent “B” will reject agent “A” and add it to the list of agents it will not negotiate with in the future. Therefore, it behooves the user to maintain a positive rating. If not, he may be

unable to buy from or sell to other users. A more sophisticated reputations facility is currently being implemented which overcomes many of the limitations in the current system [Zacharia98].

Agent-to-User Communication

A major design goal for the Kasbah@MIT system was to have agents have almost exclusive autonomous behavior when acting on behalf of their owners. However, since an agent's behavior results in some action involving the transaction of money, most users would desire to be completely informed of their agents' dealings. Agents communicate with their owners via two mediums—electronic mail and online mailboxes. Once a user creates an account, a mailbox is created which can be accessed whenever the user logs into his account. Agents send messages to the user by posting them to his mailbox. When a user creates an agent he specifies if the agent should contact him via electronic mail as well. In this regard, electronic mail is optional for each specific agent. Users who do not desire the extra clutter electronic mail from their agents may add to their electronic mail systems would settle for messages posted to their online mailboxes only. However, with the popularity of pagers and cellular phones that can receive electronic mail, the option to have one's agents send

electronic mail could be beneficial to many mobile users and allows users to act more quickly.

There are many different types of messages an agent sends to its owner including periodic status reports, prospective match found messages, and deal completed notifications. Each of these messages requires a different action on the part of the user. Some messages are simply updates while others require user intervention for the agent to proceed. Currently, a user must log into the system in order to respond to those messages requiring a response. There are a number of external factors that influence the contents of an agent's message to its owner. The most important being the number of active agents in the marketplace buying or selling the good the agent has been instructed to sell or buy, the current status of negotiations the agent is involved in, and the periodicity of status reports from the agent (as specified by the owner.)

Since the pattern matching used by agents to locate other interested agents is imperfect, agents will send their owners a message whenever they think they have located an agent buying/selling the item they wish to sell/buy; even if the item descriptions match exactly. This "prospective buyer/seller found" message gives the owner an

opportunity to review the details of the item being sold and ensures that the item the agent has found is the item the owner wants. Agents notify their owners in the case of exact title/artist matches in case multiple sellers are selling the good in varying conditions. A user may want to purchase a mint condition copy instead of a used or slightly damaged copy if given a choice. **Figure 9** shows a typical “prospective buyer/seller found” message from an agent to its owner.

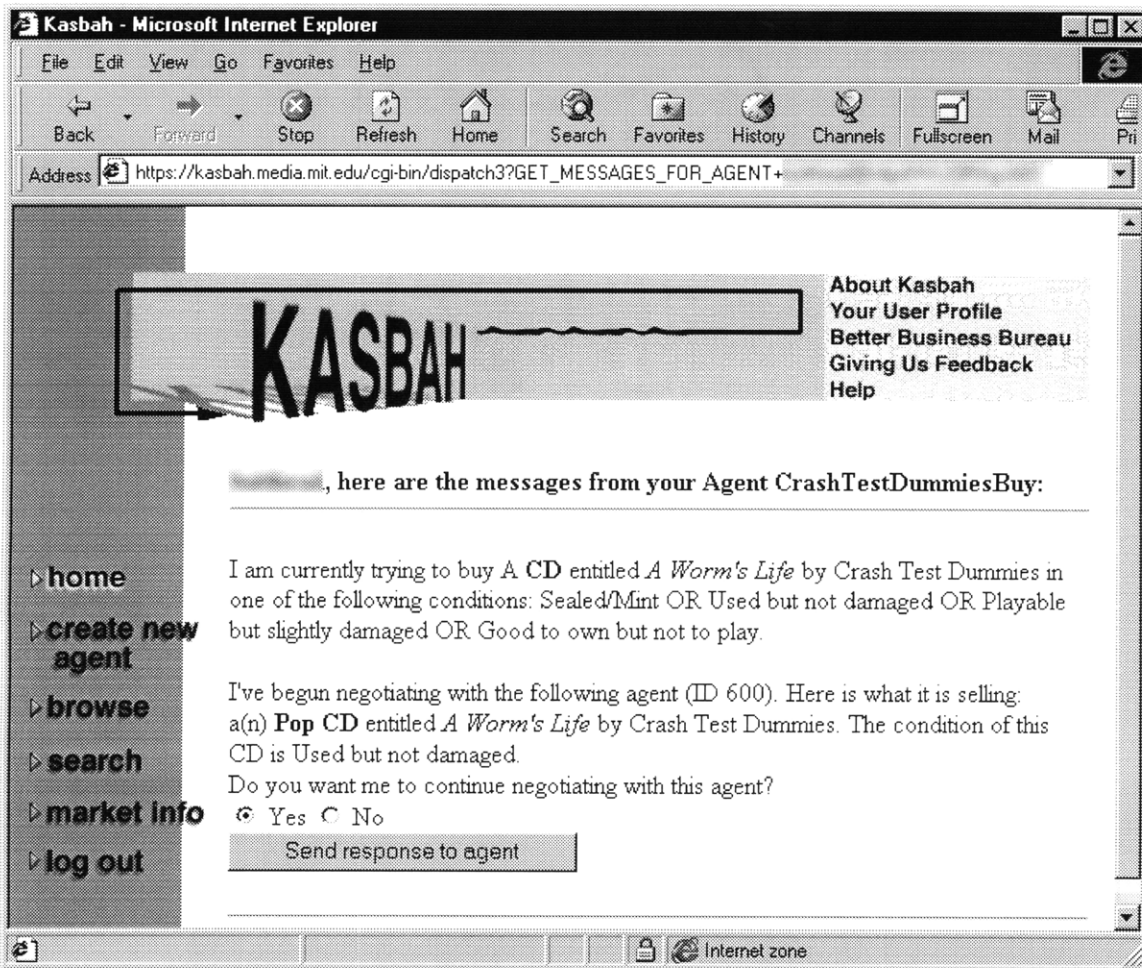


Figure 9. Prospective Buyer Found Message

This message simply informs the owner of the item the other agent wishes to buy or sell and asks the owner if her agent should begin to negotiate with that agent. If instructed to continue to negotiate, an agent will do just that. If the owner instructs the agent to not negotiate with that agent, the owner's agent will add the prospective agent to the list of agents it should ignore and will never consider negotiating with that agent again. Consequently, the answer to the agent's inquiry is final and there is no way to make an agent begin negotiating with an agent it is told not to negotiate with—to do so the user must create a new agent with the same item description. Conversely, there is no way to make an agent stop negotiating with an agent it is told to negotiate with save terminating the agent. This last point is very crucial since terminating the agent will obviously severe its negotiations with other agents as well. The pitfalls of inexact matching affect buyers more than sellers since sellers are not concerned with who buys the item they are selling. Buyers, on the other hand, would be very dissatisfied if their buying agents were to blindly purchase items the agent owners did not want.

Agents keep their owners abreast of their activities by posting/sending status reports. Depending on the settings provided by the user during agent creation, the messages will be sent only when some important event occurs or periodically (e.g., once per hour, per day, per week, etc.) An important event is generally any event that impacts the agent's ability to close a deal (e.g., a change in the other agent's offer/asking price.) **Figure 10** shows a typical status report sent via electronic mail from an agent to a user.

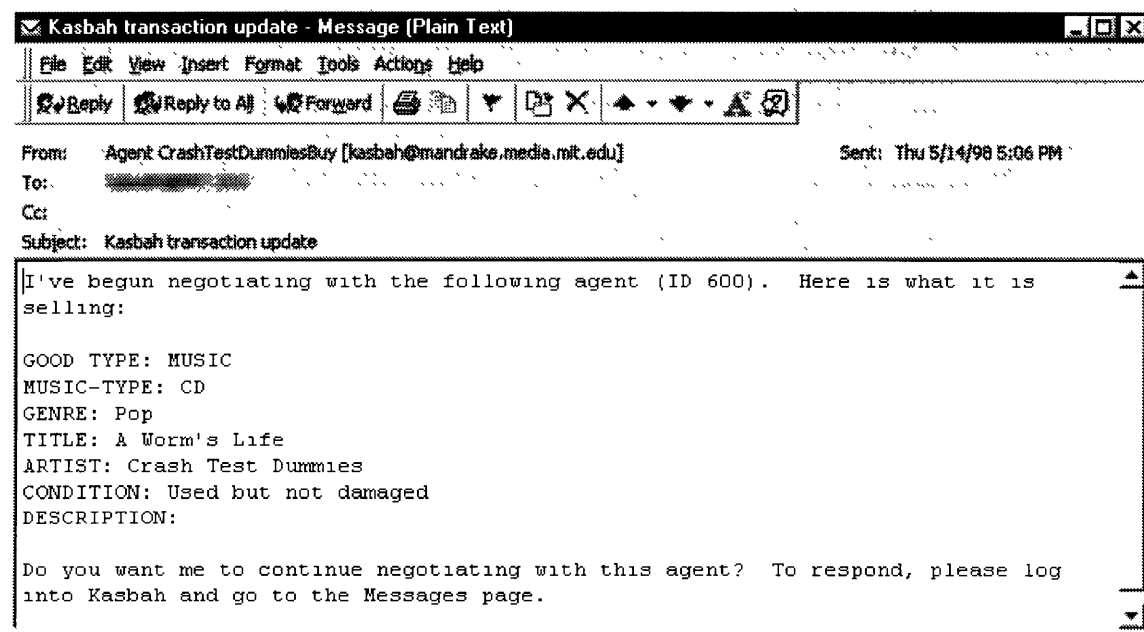


Figure 10. Agent Status Report

The status reports are designed to keep the user informed of his agents' activities. Users could opt to reconfigure their agents or

elaborate more in their item descriptions according to the information they receive via the status reports. It should be noted that status messages require no response from the user. They are purely informative.

The last type of message sent by an agent to its owner is the “deal completed” message. This type of message is sent when an agent successfully completes a deal and is meant to inform the user that it is time for her to contact the other party to make the physical transaction. In addition to informing the user of the price paid/received for the item bought/sold, the message includes the electronic mail address of the other party. Since we do not require users of the system to provide their real names, electronic mail is the only guaranteed method of identifying and contacting the other party to arrange a meeting to consummate the deal made between their agents. **Figure 11** shows a “deal made” message from an agent to its owner.

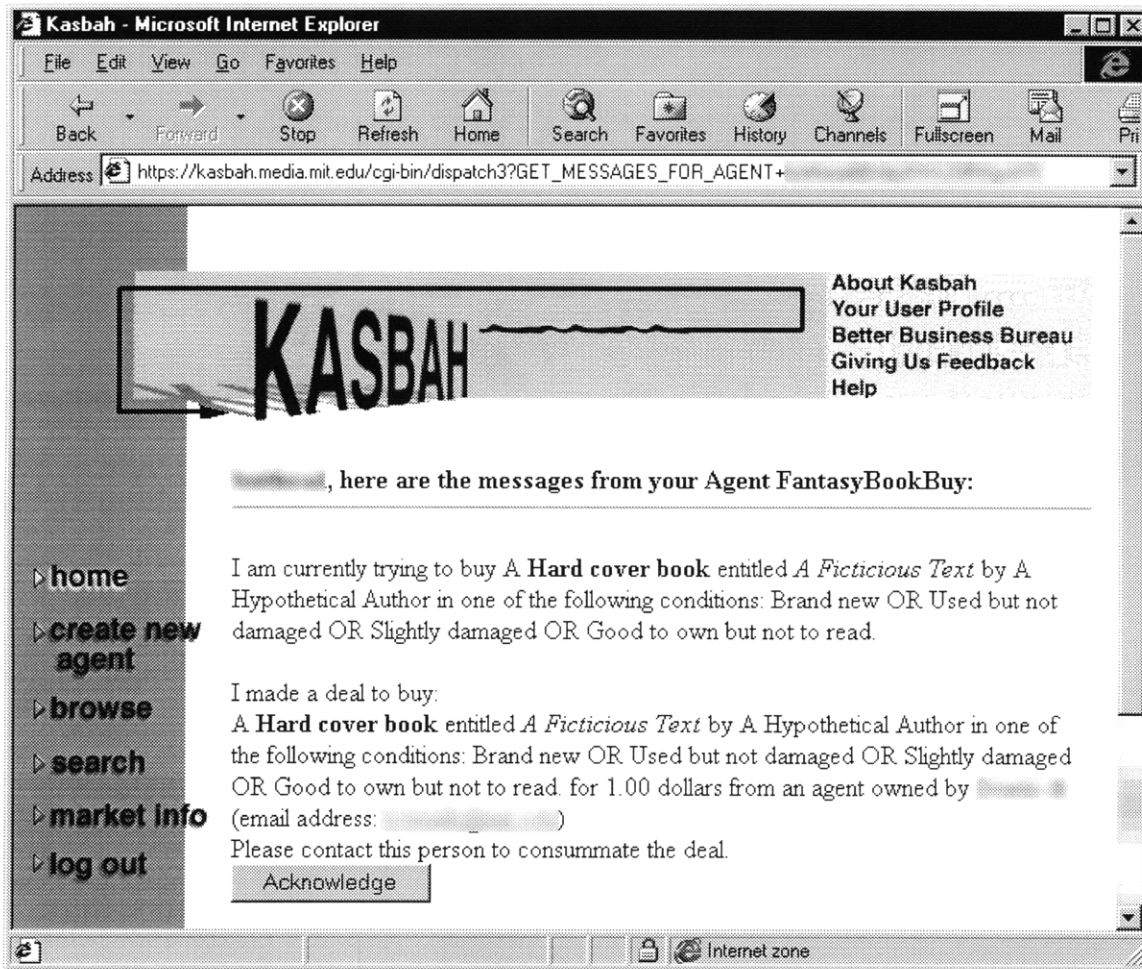


Figure 11. Deal Made Message

Unlike the other messages sent by agents, the “deal made” message is not automatically removed from the user’s mailbox when its read. This is because the user is given the opportunity to rate the other party following the physical swap of the payment and the good. The “deal made” message contains a link requesting its owner to rate

the other party. **Figure 12** shows the screen that is shown when a user decides to rate the other party.

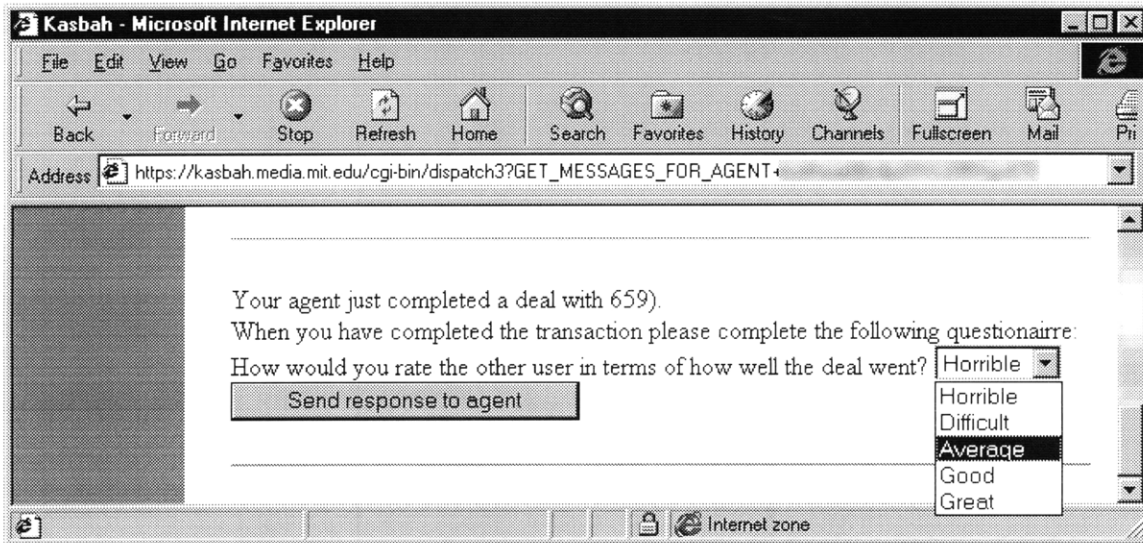


Figure 12. Rate Other Party

Users are not required to rate the other party. After rating, or choosing not to rate, the other party, the user deletes the message from her mailbox by terminating the agent.

There is a fourth type of message agents send to their owners but only a specific type of agent—not yet introduced—is capable of sending it. These agents and their messages are the topic of the next section.

Find Agents

It would be nice if users desiring to purchase or sell a book or music recording not currently available in the marketplace could be informed when the item did become available or a desire to purchase the item was listed. That is the premise behind Find agents. Find agents are similar in behavior to buying/selling agents sans the ability to negotiate. Since all agents are notified whenever a new agent becomes active in the marketplace find agents are functionally simple. A find agent scans the marketplace for a seller/buyer of the item it has been configured to monitor the marketplace for and, when one becomes available, the find agent notifies its owner. Find agents do not have infinite lifetimes and, in this regard, are configured similarly as buying and selling agents. **Figure 13** shows a typical “Item Found” message from a find agent to its owner.

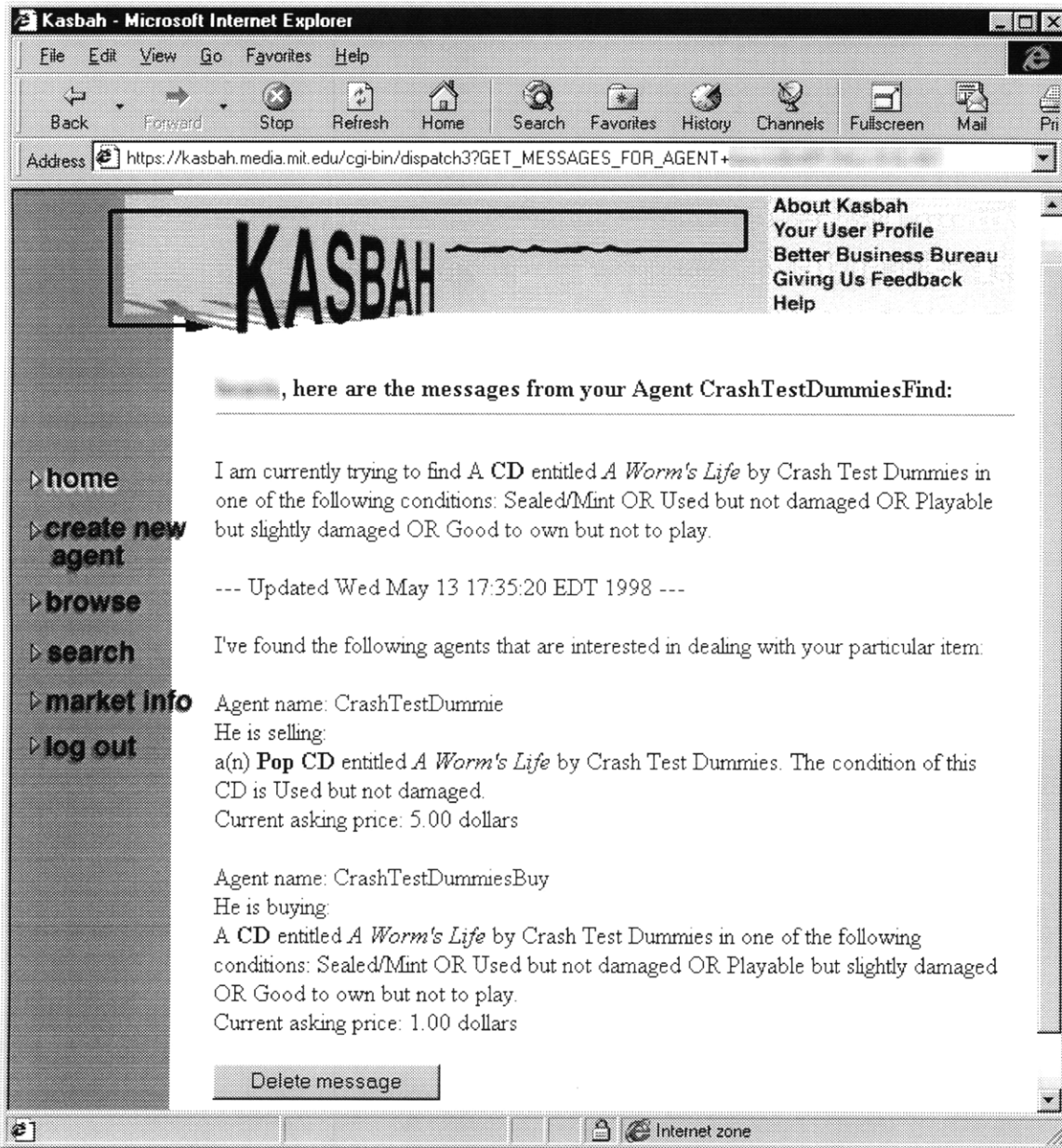


Figure 13. Item Found Message

Two particular issues that have been conveniently neglected until this point are the legal ramifications surrounding the use of the system to mislead or do financial harm and users' accountability in

deals made by their agents. Neither the Kasbah@MIT system nor its administrators provide a clearinghouse for the physical transaction of goods or any guarantees or protection from dishonest users. A lengthy disclaimer explicitly detailing the risks involved with using the Kasbah@MIT system to transact goods is presented to every user following account creation. Mechanisms such as the Better Business Bureau, Secure Sockets Layer cryptography, and passwords attempt to curtail dishonest use of the system and to discourage users with bad intentions from becoming members. Users are ultimately responsible for deals gone bad and are encouraged to seek any outside forms of legal recourse they deem necessary.

Related Work

Overview

As mentioned previously, consumer buying behavior can be broken into six primary stages [Guttman97]:

1. Need Identification
2. Product Identification
3. Product Comparison
4. Merchant Comparison
5. Payment Settlement & Delivery
6. Product Usage & Service

Product identification, product comparison, and merchant comparison define the process of matching the consumer with a vendor. Once this process is completed, the consumer and vendor agree on a price for the good or prior to delivery. If brokering is quantified as the process of matching buyers and sellers and negotiation as the process of finalizing a deal, numerous online systems exist which assist users in various aspects of brokering but few currently support negotiation. What follows are descriptions of existing online systems including their purposes, strengths, and shortcomings.

BargainFinder

BargainFinder was released in conjunction with Smart Store® Virtual, a Web site developed by Andersen Consulting to explore developments that affect companies engaging in electronic commerce, and to report on the best examples of virtual retailing [BF]. BargainFinder is an intelligent agent which shops for the lowest price of a particular music compact disc by artist and title. Although BargainFinder illustrates a useful application of software agents it has a few drawbacks. One is that it is very slow since it parses information received through queries it makes to online vendors' sites. Another is that it is stateless causing two identical requests to result in exactly the same time-consuming, brute-force, query-and-parse process rather than making use of cached results from previous queries. Another problem is that few companies want to be judged solely on price leading them to block BargainFinder agents from their site. BargainFinder assists users in merchant comparison only. Furthermore, price is the only distinguishing factor between merchants which is not always the best metric since there is no way to assess the merchants' ability to meet the user's needs. Moreover, a lower price does not guarantee a better deal especially if the user must

wait longer for the item to be delivered or is forced to pay a restocking fee if he desires to return the merchandise. Tete-a-Tete is a system currently under development that attempts to address some of these limitations by enabling agents to negotiate over more dimensions than price [Guttman98b].

FireFly

Firefly also helps the user find music, but it is more sophisticated in that it learns the user's musical preferences [Firefly]. Firefly asks users to rate a number of different artists, then it suggests other types of music that the user might like. The suggestion is based on correlations with what other people say they enjoy listening to, rather than artificial intelligence. Firefly helps the user with a different sub-problem in the overall process of shopping, namely locating goods one may be interested in. In contrast, the other systems described here help with the purchase of a good and assume the user has already decided what good they want to buy.

Fido

Fido the Shopping Doggie is a service provided by Continuum Software, Inc. to aid consumers in locating products advertised on the

Internet [Fido]. Fido aims to save users the time and frustration attributed to using generic search engines by maintaining a centralized database of vendor products and prices. The service is free to users but vendors pay a one-time fee to have their products listed in the database as well as additional fees each time the vendor wishes to update their pricing and product information. Fido parses product and pricing information from a vendor-provided URL and requires that all the pages meet certain formatting criteria. Once a shopper enters a search request, Fido provides pricing information and, in some cases, direct links to the vendors' Web page describing the item in more detail. In this capacity, Fido helps with merchant comparison only.

United Computer Exchange

United Computer Exchange Corporation's OnLine Exchange is an interactive trading and auction system on the Web for buyers and sellers of new and used computer equipment [Online]. Sellers can list computer equipment they wish to sell paying a commission to United Computer Exchange after a transaction has been completed. Buyers are able to browse listings and bid on any piece of equipment they are interested in. The OnLine Exchange system notifies the seller of a bid via electronic mail at which point the seller can choose to accept or

ignore the bid. If the seller accepts the bid, the buyer is notified via electronic mail and United Computer Exchange processes the transaction. One drawback of the OnLine Exchange is the exorbitant commissions charged for selling items. United Computer Exchange currently charges a fifteen-percent commission on items less than \$1500, twelve percent for items between \$1500 and \$3000 and ten percent for items over \$3000. There is a minimum commission of \$60 on any transaction. The system does not automate price negotiation but does assist in delivery and payment.

ONSALE

ONSALE is a Web-based interactive auction house specializing in computers, peripherals, consumer electronics, and sporting goods [Onsale]. ONSALE hosts live, online auctions of vendors' merchandise in addition to providing advertising space on their Web site. The limited quantity of each item up for bid combined with the fast-paced bidding makes ONSALE popular for shopping as well as entertainment. ONSALE locates merchants who agree to offer bargains on refurbished and discontinued merchandise, and once an order has been placed, the participating merchants are responsible for processing and collecting payments. The system provides no

information to assist bidders in price negotiation aside from whether a better bid has been offered. However, the system does include a feature termed “Bid Agent” that will automatically increase the bid amount—up to a user-specified maximum—whenever a user is outbid.

Yahoo! Classifieds

Yahoo! Classifieds is an online service that mimics conventional printed newspaper classifieds [Yahoo]. Sellers post advertisements of goods they wish to sell and buyers are able to search given categories for items they are interested in purchasing. The use of the service is free to both buyers and sellers. To facilitate communication between both parties, Yahoo! Classifieds provides online, confidential personal mailboxes. Unlike previous systems discussed, Yahoo! Classifieds is not strictly limited to merchandise transaction. Job openings, services, and personal advertisements are a few of the non-product advertisements that are posted. Yahoo! Classifieds assists in matching buyers with sellers or advertisers with targeted consumers. Good and service exchange is the responsibility of the parties involved and Yahoo! Classifieds does not guarantee the accuracy, integrity, or quality of any of the information posted. Additionally, there is no facility for rating or viewing the credentials of the other party.

Classifieds2000

Classifieds2000 provides a service similar to Yahoo! Classifieds [Classifieds]. In addition to the ability to post or respond to an online advertisement, Classifieds2000 provides the “Cool Notify” feature which notifies users via electronic mail if an item matching their search criteria is later advertised similar to Find agents in the Kasbah@MIT system. Classifieds2000 is a free service and gives users the ability to include a photograph with their advertisement. Classifieds2000, like Yahoo! Classifieds, exists primarily to match buyers with sellers. There exists no facility for assisting in price negotiation or merchandise exchange or rating other users following an exchange.

Ebay

Ebay is an online auction site similar in functionality to ONSALE [Ebay]. Ebay is primarily devoted to consumer-to-consumer auctions—although vendors can also post merchandise for sale. ONSALE relegates its consumer-to-consumer auctions to a smaller sub-site called the “ONSALE Exchange” [Onsale]. Ebay was the first site to include the “Bid Agent” functionality also provided by ONSALE. Ebay offers no assistance in merchandise exchange or price negotiation

but does provide the capacity for users to rate other users following a transaction. Rather than averaging all ratings to compute the aggregate, Ebay adds the number of positive ratings then subtracts from this sum the number of negative ratings. This score is used to award stars. A user receives one star for every point in her score modulo one hundred. One disadvantage with this approach is it takes one hundred transactions to get a better rating than the one everyone starts with. Also, a malicious user could cheat fifty percent of the time and still maintain an average rating. To discourage those with inferior ratings from assuming new identities, Ebay informs users if either the buyer or the seller has assumed a new identity in the past thirty days according to information provided when the account is created.

Previous Kasbah Experiments

As mentioned previously, there have been many predecessors to the Kasbah@MIT system used in a variety of experiments [Chavez96] [Chavez97a]. The most notable experiment occurred on October 30, 1996 at the MIT Media Laboratory's "Digital Life" symposium. The symposium attracted more than 200 attendees with varying backgrounds. The experiment involved a limited set of goods the attendees could transact over the one-day symposium. Since the

experiment terminated at the end of the symposium, agents were not configured with deadlines—they were coded to stop negotiating at 5pm. Other significant features of the experiment are it did not involve real money and the organizers, not the guests, provided the goods that were transacted. Overall, in spite of a few minor hardware failures and glitches, the experiment was a huge success and is the basis of our work involving real goods with real money in the real world.

Summary

One major hindrance to electronic commerce today is the absence of a standard data description language or syntax. All aforementioned sites use brute-force methods to parse Web pages for interesting content or require proprietary data formats and query syntaxes for them to operate correctly. Recently, there has been a big initiative to ameliorate this situation by supplementing the data formatting capabilities of Hypertext Markup Language (HTML) with a data description language, the Extensible Markup Language (XML) [W3C]. The World Wide Web Consortium has released version 1.0 of the XML specification geared toward enabling computers to extract meaning from the data embedded in Web pages. This will make

software agents, especially those operating on the Web, more powerful by enabling them to know, for example, what “price” is and how to interpret it in various contexts.

Another initiative currently under development is the formation of a standard agent communication language and protocol. This initiative is aimed primarily at multi-agent systems that require agents to pass messages back-and-forth, like the Kasbah@MIT system. The most popular attempt at forming a general agent communication specification is KQML [Finin93][Labrou94]. More recently, ObjectSpace, Inc. released Voyager which implements a number of features useful to agent work including a messaging protocol and support for the creation of heterogeneous, mobile agents [Voyager].

Just as the popularity of the Web fuels its growth as a medium for commerce, with time, developments and innovations in software agents will make agents work more viable, efficient, and manageable.

System Architecture

This chapter describes the implementation details of the Kasbah@MIT system. The purpose is to present a detailed tour of the system, its interconnections, and dependencies. As previously mentioned the Kasbah@MIT system has many predecessors—each targeted toward a specific audience or experiment and designed accordingly. Although all of these systems share the same basic infrastructure, there have been numerous modifications and additions in forming the Kasbah@MIT system.

Overview

The Kasbah@MIT system consists of many interconnected servers communicating via TCP/IP sockets. The decision to split the responsibilities of the system across multiple servers was one based on security and performance. As illustrated in **Figure 14**, the Simple Mail Transfer Protocol (SMTP) server, Web server, and marketplace server each run on separate machines. Users access the Web server via Internet browsers supporting Secure Sockets Layer. This maintains user privacy in all communications with the Kasbah@MIT Web interface.

The Web server opens socket connections to the marketplace server and dispatches commands via CGI scripts. The marketplace server hosts the actual agents and is where negotiation occurs and user records are stored. The SMTP server provides an electronic mail interface for the Kasbah@MIT system allowing agents to communicate with their owner's without the owner logging into the Kasbah@MIT system.

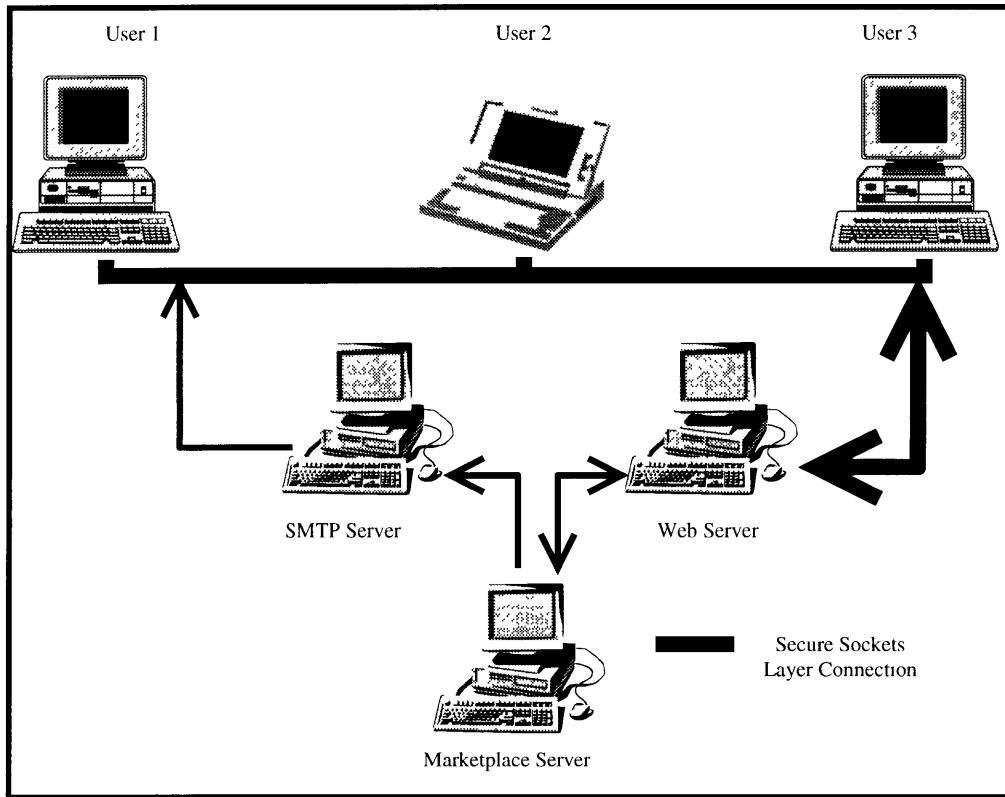


Figure 14. System Interconnections

Over the course of implementation and debugging, the modularity of this architecture has been one of its most redeeming

features. This flexibility made it easy to modify one part of the system without affecting others and to make substitutions of different versions of subsystems to ascertain the most useful and functional of the alternatives.

Another invaluable administrative tool is the Root Operations toolkit. Root Operations allow administrators to access the system in supervisor mode. This privileged mode is very helpful when users forget their passwords or the system gets congested with inactive agents. Root Operations is implemented as a distributed process making it impossible for a user to gain privileged access without simultaneously compromising both the Web and marketplace servers.

Web Server

Users access the Kasbah@MIT system via any Secure Sockets Layer (SSL) enabled Web browser (e.g., Microsoft Internet Explorer or Netscape Navigator.) The Web server is actually a hybrid consisting of the typical Hypertext Transfer Protocol (HTTP) server and a battery of Common Gateway Interface (CGI) scripts. The CGI scripts give the Web server the ability to respond to user requests with the dynamic data characteristic of systems like Kasbah@MIT.

There were basically two options in deciding how to construct the user interface for the system. The first was to use the Web. The second was to build a custom interface from the bottom up. Although the latter option would have provided greater control and flexibility when designing the interface, it would have required a tremendous amount of time and effort. Also, and more importantly, custom interfaces generally are not portable across different architectures and operating systems. Therefore, had we taken the custom interface approach, we would have spent the majority of our time debugging cross-platform issues or fielding complaints from users whose systems failed to adequately support our interface.

The ubiquity of the World Wide Web made it an ideal choice for the foundation of the user interface. Although the different implementations between browsers and across versions diminish the plausibility of viewing the Web as a standard interface as well, there is enough overlap to justify its use. Furthermore, Web page design is very easy and many powerful tools exist that make designing pages with a lot of functionality a cinch.

As shown before, most user interaction with the Web server occurs through simple HTML forms. The user completes fields and

makes appropriate selections then clicks a button to submit the data to the Web server. There are many routes processing can take at this point. In most cases, a user's request will require the Web server to communicate with the marketplace server to gather the necessary data or to request changes to be made on the user's behalf. In some cases, however, the Web server is capable of receiving, interpreting, and completing the user's request without intervention by the marketplace server. An example of this last case is when a user requests help on using the Kasbah@MIT system. The following list splits every possible Web server request/query into two categories—those requiring a connection to the marketplace server and those that do not:

Require a marketplace connection	Do Not require a Marketplace connection
Login Show Active Agents Browse the marketplace Search the marketplace View marketplace statistics Modify user profile View reputation/rating	Get help on using the system Learn about the experiment Submit feedback Logout

The Web server is responsible for converting users' requests into the proper syntax to be sent to the marketplace server—if necessary. For those requests not requiring a connection, the CGI scripts dynamically

generate the HTML response pages and forward them to users' Web browsers for display.

The request-response messaging protocol used between the Web and marketplace servers is string-based and consists of commands with varying data fields. For example, when a user requests to log into her account by submitting her username and password, the Web server extracts the username and password data from the form. Next, the CGI scripts form a proper login message to be sent to the marketplace server. This message would be a concatenation of the command, "LOGIN_REQUEST", and the username/password pair. The Web server CGI scripts would open a socket connection to the marketplace server and post the message. The marketplace server would fulfill the request to login the user by computing the usercode associated with the specified account and generating the user's home section populated with summaries of all the user's agents and unread messages.

Marketplace Server

The marketplace server is the heart of the Kasbah@MIT system. It is here that agents exist and operate and also where the data used to generate users' Web pages reside. The marketplace server is written

completely in Java 1.1. The marketplace server can be decomposed into its several constituent parts as shown in **Figure 15**.

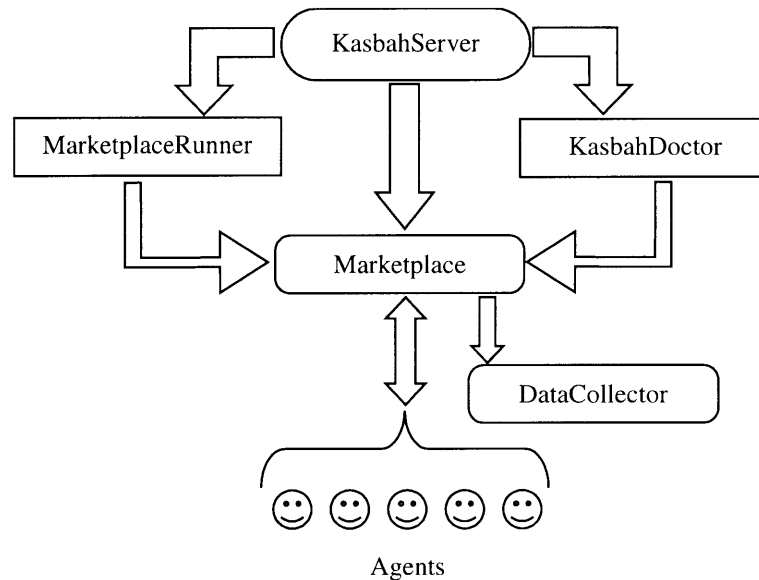


Figure 15. Marketplace Server Decomposition

The **KasbahServer** object is the top-most object in the marketplace server hierarchy. The **KasbahServer** contains the three threads of execution in which all other marketplace server processes run. The main thread periodically polls a specified socket for connections from the Web server. If a connection is established between the Web and marketplace servers, the message sent by the Web server is parsed and a dispatcher is called to take the appropriate action for the contained message.

The secondary thread, the `MarketplaceRunner`, is responsible for scheduling and executing each agent that is currently active in the marketplace. The `MarketplaceRunner` thread executes an agent, yields the processor, and then executes another agent. The scheduling algorithm randomly selects the agent to be executed next and guarantees that no agent will run twice before every agent is run at least once. Once every agent receives its time-slice of the processor, the scheduler resets the loop and continues from the beginning of its random, round robin algorithm. During its time-slice, each agent can send messages in its outgoing message queue and read the messages in its incoming queue.

The tertiary thread of execution is the `KasbahDoctor`. The `KasbahDoctor` is diagnostic in nature and monitors the marketplace server. If the `KasbahDoctor` senses an error in the marketplace, it will notify a specified list of administrators via electronic mail. The `KasbahDoctor` periodically polls the `MarketplaceRunner` for activity. If the `MarketplaceRunner` is inactive longer than some customizable duration, the `KasbahDoctor` assumes that an error exists and sends the notifications accordingly. The `KasbahDoctor` runs in a separate Java virtual machine so it and the `KasbahServer` are two distinct

processes. However, their interoperation makes it convenient to think of them existing as a single entity.

The marketplace object is the containing object for all agents in the Kasbah@MIT system. A reference to the marketplace object is passed to the MarketplaceRunner thread. Each agent is provided a message queue in the marketplace object that accepts messages from other agents as well as the marketplace. The marketplace is also responsible for notifying every agent when a new agent is added to or removed from the marketplace.

The DataCollector object is responsible for capturing any and all relevant data generated by agent-to-agent and agent-to-marketplace interactions and to store them to disk. The system captures all item description and agent configuration data whenever a new agent is created and when an agent is reconfigured. Additionally, the system captures transaction completion details, and system malfunction information. These diagnostic data proved invaluable while doing system maintenance and tracking odd behavior.

The marketplace server is physically stored in a secure location and password protected. However, since it is connected to the network, additional measures were taken to ensure the system could not be

easily compromised. Whenever the marketplace server receives a connection request, it compares the IP address of the requesting machine with a list of trusted machines. The machines on the list are located in the same room as the marketplace server, connected to the same sub-network, and under our group's control. A person from outside the laboratory or on a different sub-network will have their connection request rejected whenever they attempt to access the marketplace server directly.

Since the system operates on real data in real time it is necessary to provide adequate measures to ensure that the system is able to recover from errors or, in the worst case, fails gracefully. To provide this desired behavior, all agent, user account, and item information had to be persisted to fixed storage. Furthermore, all interaction and interoperations had to be transaction-based to prevent inconsistent states and corrupt data should the system fail. We employed Object Design Inc.'s ObjectStore Persistent Storage Engine Professional for Java, a commercially available persistent storage engine, to achieve both goals [ODI]. All marketplace operations are within a transaction that either completes fully or not at all. Also, all agent, user, and non-transient marketplace information is persisted to

disk periodically. As a result, the system can tolerate hardware and software errors. Restarting the system following failure requires backing up to the last, fully completed transaction. At that point the system is guaranteed to be in a safe state and execution can continue normally. It would be unacceptable if, following a system malfunction, every user had to recreate their user accounts and all of their agents. Object persistence and transaction-based execution minimize the likelihood of such a disaster.

The system does not scale well due to the use of a single marketplace and the round-robin scheduling algorithm. With several hundred agents coexisting in the marketplace it takes on the order of several minutes for two agents to complete a transaction. This is a factor of how often the agents are run and not necessarily the size of the time-slice. If one of one hundred agents is selected for execution each second, using a round-robin scheduling algorithm would imply that the agent will be executed again in approximately ninety seconds. As more agents are added to the system, the per-agent execution frequency drops proportionately. If an agent was selected to execute more than once per second the system would be unresponsive to users.

Mail Server

The mail server is a Simple Mail Transfer Protocol (SMTP) server. Agents rely heavily on the mail server in communicating with their owners. As mentioned before, typical agent-to-owner communication includes periodic status reports, “prospective buyer/seller found” messages, or “deal completed” messages. These messages are posted to the user’s online mailbox and, optionally, sent to the user’s electronic mail account. The mail server makes this possible.

Agents send mail messages to the mail server by first composing a `MailMessage` then posting the `MailMessage` to a `MailServer`. `MailMessage` and `MailServer` are Java objects that abstract the details of electronic mail messages and SMTP servers, respectively. A `MailMessage` includes a list of the intended recipients, the sender information, the subject of the message, etc. A `MailServer` object is configured, during construction, to emulate a specific SMTP server and user account on that server. Passing a `MailMessage` object to a `MailServer` causes a series of events to occur. First, a TCP/IP connection is made to the underlying SMTP server being emulated by the `MailServer` object. Following the standard handshaking process

the MailServer uses the data stored in the provided MailMessage object to send the actual message via the SMTP server.

RootOperations

RootOperations is a group of tools designed to assist in the maintenance of the Kasbah@MIT system. There are many instances when it is necessary to access or modify the system during operation. However, administrative type of operations must be accessible only to system administrators. RootOperations allows system administrators to access the Kasbah@MIT system in privileged mode without necessitating that the machine be taken offline.

The following is a list of operations provided by the RootOperations administrative toolkit:

- Change a user's password
- Delete/View a users' accounts
- Delete/View users' agents
- Change the periodicity of system backups

The RootOperations user interface has two implementations. Administrators are able to access RootOperations' functionality via a Java 1.1 applet that residing on the Web server or from a command line interface implemented as a Java 1.1 application. **Figure 16** shows the RootOperations applet interface with the "Delete Agent" operation

highlighted and **Figure 17** shows the RootOperations command line interface following a “Set Backup Timer” operation.

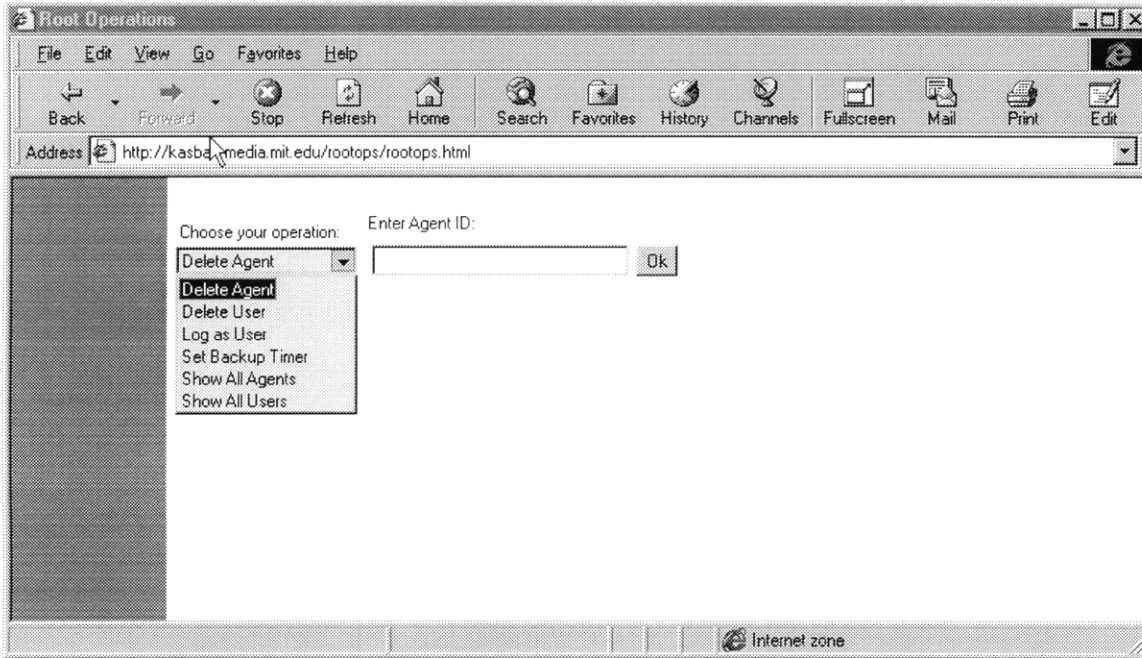


Figure 16. RootOperations Applet Interface

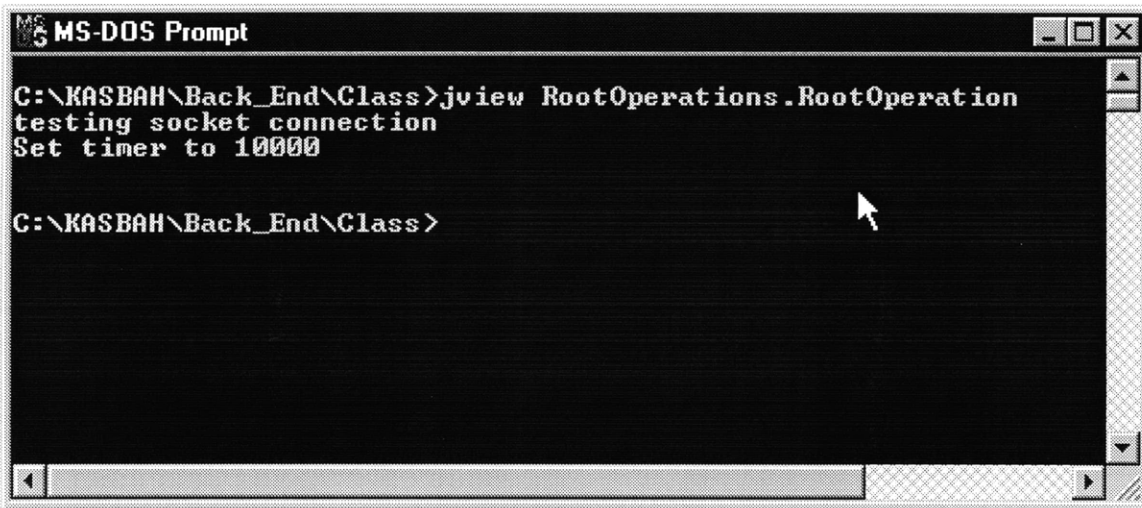


Figure 17. RootOperations Command Line Interface

Discussion of Experiment

Overview

On February 3, 1998, the Kasbah@MIT system was released to the MIT community to begin a real-world experiment in agent-mediated electronic commerce with real merchandise and real money. All previous experiments were done in a controlled environment with a predefined set of goods [Chavez96] [Chavez97a]. The purpose of this experiment was to study the benefit and practicality of using an agent-mediated electronic marketplace to transact goods.

The goal was to have students transact their used textbooks and music recordings using the Kasbah@MIT system instead of other well-established means. Since the MIT community is generally above average in their knowledge and experience with the Internet and, more specifically, the World Wide Web, we assumed the indoctrination and familiarization period would be rather short. We intended to evaluate the merits and drawbacks of the system by assessing the following issues:

- Did people use the system?
- Were people confident in their agents' abilities to work on their behalf?
- Were people pleased with the results of their agents' efforts?
- How did the system perform (e.g., did it scale well?)

- How closely does consumer buying behavior in the electronic marketplace mimic that in the real world?

The system was configured to capture data that could be used to provide the following metrics:

- Total number of transactions completed by agents
- Number of agents created by users
- Average time for deal completion
- Ratings of users as a function of time
- Periods of highest and lowest user activity
- Number of times agents are reconfigured
- Number of each type of agent created
- Number of each type of good bartered
- Number of agents unable to make a deal before deadline
- Different types of bargaining functions used
- The number of times Find agents are created versus the number of Buy agents

We hoped to draw conclusions based on these data and user feedback. To make submitting feedback non-intrusive, we provided a simple online form users could access while logged in. Also, a special electronic mail alias was created for users who wished to send comments and suggestions without being required to navigate to the Web site.

Quantitative Analysis

The graphs in **Figure 18** and **Figure 19** illustrate the number of each type of agent created for books and music, respectively. **Figure 20** shows the total number of transactions completed between

users since the system was put online. The low number of transactions shown in **Figure 20** was somewhat disturbing but equally informative. As shown in the images, many agents of each type for each particular good type were created. This information combined with the fact that 719 people joined the system implies that there was some interest in the concept of automating the transaction of goods. Furthermore, we received no user complaints concerning their agents' inability to make transactions on their behalf.

BOOK AGENTS		
BUY	Active	(72)
	Total	(95)
SELL	Active	(66)
	Total	(70)
FIND	Active	(76)
	Total	(94)

Figure 18. Book Agent Statistics

MUSIC AGENTS		
BUY	Active	(89)
	Total	(130)
SELL	Active	(103)
	Total	(141)
FIND	Active	(49)
	Total	(77)

Figure 19. Music Agent Statistics

TRANSACTIONS	
MUSIC	(15)
BOOK	(2)
TOTAL	(17)

Figure 20. Transaction Statistics

Those monitoring the marketplace daily would quickly realize that the Kasbah@MIT system is not commodity-based like online auctions tend to be. In fact, in terms of transaction throughput, the system more closely follows online classifieds where the majority of the items posted are never sold. Bootstrapping the marketplace requires that a form of marketplace already exist. That is, to increase the number of transactions in a marketplace, the marketplace must contain many items that a lot of people want. This is a fundamental problem in a non-commodity-based system such as Kasbah@MIT. Unfortunately, due to the low number of transactions, the results of this experiment are quantitatively inconclusive. The qualitative data received via user feedback were more encouraging.

Qualitative Analysis

Qualitatively, the system performed very well. In the many demonstrations and presentations of the system, the system and concept were showered with praise. As mentioned before, users were not particularly concerned with the low number of transactions provided by the system. The idea that their agents were tireless and would continue to work on their behalf should a match occur is more than they are guaranteed via conventional methods. Although the

user interface and layout of the system were highly praised in user feedback, there was also some major dissatisfaction surrounding general usability and system responsiveness.

Many users requested the ability to create agents in batches rather than one at a time. The idea would be that those wishing to sell multiple music items at the same price using the same negotiation strategy should not be required to go through the item description and agent configuration processes for each item.

Another recurring problem dealt with system responsiveness whose origins lie in system scalability. As more and more agents became active in the system, the tradeoffs of system responsiveness to user browsing and agent execution became more apparent. The marketplace server is responsible for executing each agent as well as retrieving the data necessary for the Web server to dynamically construct user Web pages. Although the marketplace server ran on a dual-processor machine, the contention for shared resources—particularly agent and user data—caused each thread to block while the other executed its critical section. This was doubly bad since users had to wait increasingly longer periods of time to view their agents' activity only to realize their agents had made very little progress in the

interim. There was no way to permanently solve this problem given the resources at our disposal. Instead, we monitored the system for periods of greatest user activity and increased adjusted the threads' time-slices accordingly.

Future Work and Conclusions

Future Work

There are many areas in which the current system excels and at least as many areas in which it can be improved. As we realized in our experiment, the system is not quite ready to host thousands of users with millions of agents transacting hundreds of items in dozens of categories. The architecture must change drastically for this to occur. The successor system has already begun to replace Kasbah@MIT. This new infrastructure will support distributed marketplaces and will allow agents to transact an unlimited set of goods. Additionally, this new infrastructure will allow negotiation over multiple dimensions rather than the single dimension of price. This last feature is more vendor-friendly than the consumer-to-consumer focus of the Kasbah@MIT system. There are three major problems in the current system that will be addressed in the future. These are:

- 1) Limited number of categories and items
- 2) Scalability
- 3) Limited agent negotiation strategies

The Kasbah@MIT system limits the good types to music and books. This constraint is the result of the inherent difficulty of thoroughly describing the distinguishing traits of an arbitrary good to

an agent. It is easier to parameterize a known, predetermined set of goods (i.e., books and music) than it is to support goods from arbitrary categories. Furthermore, in a system where categories are not predetermined, two people may describe the same item using completely distinct sets of parameters, syntaxes, and features. For example, if you asked a group of people to describe the important features when choosing a car or computer—or the harder question of what makes one particular car better than another—you are almost guaranteed to get a wide variation in responses. Natural language parsing is still in its infancy making it difficult for computers to automatically extract meaning and to comprehend elaborate textual descriptions of items the way humans can. We bastardize this process by using forms in the Kasbah@MIT system thus limiting the types and amount of information one can use to distinguish items. This process would lead to a maintenance nightmare were it used for arbitrary categories of goods. One would be required to develop templates and queries for each particular item covering every conceivable nuance and idiosyncrasy of methods and parameters used to describe that item by every possible user of the system.

The scalability issue, while more straightforward than support for arbitrary item categorization, is as integral to a more general architecture. Scalability can be realized by simply adding more resources to the architecture. Faster processors, more processors, and faster I/O are a few of the ways that will temporarily “solve” the issue of scalability. The problem with this approach is that it is temporary and will quickly lose its appeal once more agents, users, and items are added to the system. Another more practical approach would be to make marketplaces distributed objects. There is no strong argument for having agents interoperate in a single marketplace. With an architecture that supports multiple marketplaces (each handling a different good or set of goods), agents could be created in another marketplace having more available resources. This does not necessitate mobile agents. However, a registry of some sort would be required to locate other agents and marketplaces.

Kasbah@MIT agents are restricted to using one of three specified negotiation strategies. As mentioned before, these strategies would not be attractive to all users in a more general system. In particular, many merchants refuse to make their information available if the sole comparison criterion is price. This is to be expected since

price is one of many dimensions one can use to compare merchants of the same item. For example, a particular car dealership may include more features, a longer warranty, and better financing than another dealer. However, an agent comparing the two dealers solely on price would get its owner the less expensive car—which may not be the most economical one.

Find agents could be more useful if they could be configured to use wildcards in their search criteria. For example, a user may wish to be notified whenever any Mozart compact disc or Jazz album becomes available in the marketplace.

Conclusion

The Kasbah@MIT system was developed to study the practicality and feasibility of using software agents to transact goods in a real-world experiment. Although the quantitative data for the ongoing experiment are inconclusive many users of the system find the system conceptually useful. There were many software engineering issues, tradeoffs, and decisions that were made over the course of the system design, development, and deployment. This experience has been very rewarding and we hope it will be rewarding to those who

save time by allowing software agents assist them with future transactions.

References

- [Alba96] Alba et al. "Interactive Home Shopping and the Retail Industry." Report for the Marketing Science Institute, p. 1, July 1996.
- [Chavez96] Chavez, A., and P. Maes. "Kasbah: An Agent Marketplace for Buying and Selling Goods." Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM96), pp.75-90. London, UK, April 1996.
- [Chavez97] Chavez, A., A. Moukas, and P. Maes. "Challenger: A Multi-Agent System for Distributed Resource Allocation." Proceedings of the First International Conference on Autonomous Agents, Marina del Rey, CA, February 1997.
- [Chavez97a] Chavez, A., D. Dreilinger, R. Guttman, and P. Maes. "A Real-Life Experiment in Creating an Agent Marketplace." Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM97), London, UK, April 1997.
- [Classifieds] Classifieds2000 Web site. <http://www.classifieds2000.com/>
- [Excite] Excite Shopping Web site. <http://jango.excite.com>
- [Fido] Fido: the Shopping Doggie! Web site. <http://www.shopfido.com/>
- [Finin93] Finin, T., J. Weber, G. Widerhold, M. Genesereth, R. Fritzson, D. McKay, J. McGuire, R. Pelavin, S. Shapiro, C. Beck. "Specification of the KQML Agent-Communication Language." Enterprise Integration Technologies - Technical Report EIT TR92-04, Palo Alto, 1993.
- [Firefly] Firefly Web site. <http://www.firefly.com/>
- [Guttman97] Guttman, R. and P. Maes. "Agent-mediated Integrative

Negotiation for Retail Electronic Commerce.” Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET98), May 1998.

[Guttman98] Guttman, R. and P. Maes. “Agent-mediated Integrative Negotiation for Retail Electronic Commerce.” To appear in Proceedings of the Workshop on Agent Mediated Electronic Trading Workshop. Minneapolis, MN. May 1998.

[Guttman98a] Guttman, R. and P. Maes. “Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce.” To appear in Proceedings of the Second International Workshop on Cooperative Information Agents. Paris, France, July 3-8, 1998

[Labrou94] Labrou, Y. and T. Finin. “A Semantics Approach for KQML—A General Purpose Communication Language for Software Agents.” Proceedings of CIKM94, New York, 1994.

[Maes94] Maes, P. “Agents that Reduce Work and Information Overload.” Communications of the ACM, Vol. 37, No. 7, pp. 31-40, July 1994.

[Maes95] Maes, P. “Intelligent Software.” Scientific American, Vol. 273, No. 3, pp. 84-86. Scientific American, Inc., September 1995.

[ODI] Object Design Inc. Web site. <http://www.odi.com/>

[Online] OnLine Exchange Web site. <http://www.uce.com/>

[Onsale] ONSALE Web site. <http://www.onsale.com/>

[Shneiderman97] Shneiderman, B. “Direct Manipulation Versus Agents: Paths to Predictable, Controllable, and Comprehensible Interfaces.” Software Agents, edited by Jeffrey M. Bradshaw. American Association for Artificial Intelligence. Menlo Park, CA, 1997.

[Voyager] ObjectSpace, Inc. Web site. <http://www.objectspace.com/>

[W3C] The World Wide Web Consortium (W3C) Web site.

<http://www.w3c.org/>

[Yahoo] Yahoo! Classifieds Web site. <http://classifieds.yahoo.com/>

[Zacharia98] Zacharia, G. and P. Maes. "Collaborative Reputation Mechanisms in Online Communities." MIT Media Laboratory Software Agents TR-1998-1.
<http://agents.www.media.mit.edu/groups/agents/techreports/1998/1/latest/paper.pdf>