Kasbah: An Agent Marketplace for Buying and Selling Goods

Anthony Chavez and **Pattie Maes**
MIT Media Lab
20 Ames Street
Cambridge, MA 02139
asc/pattie@media.mit.edu

Abstract

While there are many Web services which help users find things to buy, we know of none which actually try to automate the process of buying and selling. Kasbah is a system where users create autonomous agents to buy and sell goods on their behalf. In this paper, we describe how Kasbah works. We also discuss the implementation of a simple proof-of-concept prototype.

Introduction

Kasbah is a Web-based system which allows users to create autonomous agents which buy and sell goods on their behalf.

Kasbah originated with the notion of reinventing the classified ads. We observed that there are many sites on the Web that list ads. Some of these sites allow users to perform keyword searches on the database of ads (Seattle Times/PI 1996). Some have their ads nicely categorized , making it easy to find ones of interest (ADWorld 1996). Other sites have more advanced searching capabilities. For example, Stanford's Infomaster (Infomaster 1995) (Geddis et al. 1995) allows users to specify a query that precisely describes the kind of apartment they are looking for. Infomaster searches the ad databases of several local newspapers and returns ads for those apartments which match the user's description.

These "classified ad" sites have different types of functionality, yet all work under the same basic premise: to provide a tool for the user to find ads which they are interested in. Certainly, such tools are important and useful. Yet, they only help with one part of the entire process of buying or selling, that of finding ads which match what one is looking for. The idea behind Kasbah is to provide agents which help users by actually *buying and selling goods for them.*

Overview of Kasbah

Kasbah is a Web site where users go to buy and sell things. They do this by creating *buying* and *selling* agents, which then interact in the *marketplace.*

Selling and Buying Agents

Think of a selling agent as being analogous to a classified ad. When a user creates a new selling agent, they give it a description of the item they want it to sell. Unlike the traditional classified ad, though, which sits passively in its medium and waits for someone to notice it, Kasbah's selling agents take a pro-active role. Basically, they try to sell themselves, by going into the marketplace, finding interested parties (namely, buying agents) and negotiating with them to reach a deal.

A selling agent is autonomous in that, once released into the marketplace, it negotiates and makes decisions on its own, without user intervention. Of course, the user does have overall control of its behavior. When the user first creates a selling agent, they set several parameters to guide its behavior as it tries to sell the item. These parameters are:

- *Desired date to sell the item by.* People often have deadlines by which they need to sell something. For example, a graduating student might want to sell their bicycle before they leave school, because they cannot take it with them. In other cases, a person may not care when they sell something by, and are willing to wait until they receive their desired price.
- *Desired price.* This is the price the user would like to receive for the good they are selling.
- *Lowest acceptable price.* This is the lowest price the user will sell their good for. If the user has "junk" in their basement that they want to get rid of, they may set the desired price rather high, hoping someone might be willing to pay it, and also set the lowest acceptable price to a more realistic level. On the other hand, a person willing to accept nothing less than their asking price would set the lowest acceptable price to be the desired price.

These parameters define the agent's goal: to sell the item in question for the highest possible price — ideally, the desired price, but as low as the lowest acceptable price, if that is what it takes to attract buyer interest. Exactly how to achieve this goal is left to the agent. The appropriate metaphor here is that of

a personal assistant (Maes 1994). You tell your personal assistant what you would like to be done ("sell this for the best possible price"), and trust it to figure out how to accomplish this task, freeing your time and energy for more interesting pursuits. In addition, we hope that agents might be able to sell (and buy) goods better than the user would be able to, by taking advantage of their edge in processing speed and communication bandwidth.

While an agent is "free" in terms of how to achieve its objective, the parameters described above suggest how it should work. The crude heuristic is: begin by offering the good at the desired price. If there are willing buyers, great. If not, as time goes along, lower the asking price to entice more interest. When the desired date to sell the good by rolls around, the asking price should be about the lowest acceptable price. Of course, all the interesting action is in the subtleties and nuances of how the selling agent goes about lowering the price. It is possible that there will be no buyers (perhaps the lowest acceptable price is too high, or no one interested in what the agent is selling). In this case, the agent fails to achieve its goal.

The user can check on its selling agents, see who they have talked to, and what prices they have been offered. This information might prompt the user to do something like lower an agent's price parameters, if they see that offers are coming in much lower than expected.

The user will always have final control over their agents. When a selling agent reaches an agreement with a buying agent, the user may want to give the ok, so to speak, before the agents "shake hands" on the deal. The user has a couple of parameters they can set on both selling and buying agents:

- *Get user approval before finalizing deal.*
- *Send email notification when agreement reached.* The user might not be logged into Kasbah when their agent reaches a tentative agreement, and sending email is a convenient way to alert them.

The five parameters given above are by no means exhaustive. One can imagine many more controls a user might want to set on their agents, depending on the complexity of these agents. For example, there could be a parameter that defines the function used by the selling agent to lower its asking price over time. There might be a parameter that tells the agent to only negotiate with agents whose users are in a certain physical region (e.g., within the city of Boston). There could also be a parameter which tells the agent what selling strategy to use, if indeed the agent is smart enough to have multiple strategies. The parameters we have given constitute what is minimally necessary for the agents to be of use.

Thus far we have discussed selling agents. There are also buying agents. They are essentially the symmetric opposite of selling agents. Their job is to buy goods on behalf of users. One can think of a buying agent as a want ad which actively seeks to find and buy what it's looking for. When creating a buying agent, the user describes the item of interest. Alternatively, they could specify a set of selling agents already in the marketplace, and direct their buying agent to buy from one of them. Like for selling agents, the user sets parameters to guide the agent's negotiation behavior:

- *Desired price.* What the user wants to pay for the good.
- *Highest acceptable price.* The highest price the user is willing to pay for the good.
- *Date to buy the item by.*

Once created, the buying agent is released into the marketplace, where it negotiaties with selling agents, trying to make the best possible deal.

Once a buying and selling agent have reached agreement on a price, and gotten their respective user's approval, then the physical transaction of the good can occur. At this point, the human users must take over. In the future, agents may be able to do this too, using electronic cash, and if the goods in question are things which do not require a physical medium, such as information and knowledge.

We would not be building Kasbah if we did not think buying and selling agents would be useful for the everyday end-user. Here are some of the services and benefits which these agents will provide:

- Spare the user from having to find, negotiate, and in general deal with buyers and sellers. Kasbah will be eliminating human-human contact. On the surface, this might sound like a bad thing. But let's admit it, a lot of people don't like talking to strangers, which is what is generally required when buying or selling something via the classifieds. Language barriers and misunderstandings are often a problem. With agents doing the talking, though, this process is depersonalized. Here is a case where technology is not increasing human interaction, but reducing it, freeing people to pursue their more meaningful relationships.
- Know who are prospective buyers and sellers. The buying and selling agents we are building remember everyone who they have talked to. This information can be accessed by their creators, which can be very useful. Suppose that a potential buyer asks you to clarify your item description. You respond to their query, but you may receive several more requests for clarification, and answering them all is annoying. Armed with information about which people are interested in what you're selling, you can pre-emptively send out a clarification to all of those users.
- Enable better pricing. In addition to recording who they have talked to, the agents also record the content of their conversations (e.g., Agent 14 offered me

$60.) This allows users to gather price information. They may see that they are asking too much, or alternatively, asking too little.

The Marketplace

Buying and selling agents meet and negotiate in the Kasbah marketplace. The marketplace's job is to facilitate interaction between agents. There are many possible roles the marketplace could play depending on the type of market. At a minimum, the marketplace needs to ensure that the agent's participating in it speak a common language. Kasbah's marketplace will be more pro-active. It will direct agents to areas of common interest within the marketplace. What this means is that when an agent enters, the marketplace will ask what it is buying or selling, and direct it to other agents buying and selling the same kinds of things. The other agents in the marketplace are also notified of the arrival of the new agent. Think of this process as the marketplace forming "tents" within itself. For example, there might be a tent for cars, a tent for apartments in Cambridge, a tent for stereo equipment, etc. The marketplace also determines the terminology spoken, that is, how goods are described. In Kasbah, this terminology will be extendible by users.

We can consider more complex marketplaces. We might want to have marketplaces which regulate the activities that occur within them. As buying and selling agents become more intelligent, we can easily imagine malicious and deceitful agents, trying to rip off honest ones. In such a world, there will need to be some kind of law enforcement. We might have regulator agents roaming the marketplace to ensure that no illict activity takes place. We have not yet considered in depth the social issues associated with complex agent communities, but we are aware of them, and a lot of theoretical research has been done in this area (Rosenschein and Zlotkin 1994).

It is very important to note that Kasbah's architecture is agent independent. As long as an agent can speak the common marketplace language, i.e., can present the appropriate interface, then the agent can participate in the marketplace. Our goal is not to try to once and for all create the world's best buying and selling agents. People will always want to design new, improved agents, using clever techniques to make them as smart as possible. We want to encourage people to do this. Hopefully, Kasbah will provide a real-world incentive to create many new, innovative agents.

Kasbah Prototype

To test the basic concepts and feasibility of what has been described above, we have built a Kasbah Prototype.

Implementation

The Kasbah Prototype is implemented in CLOS using Harlequin Lisp. As is standard in CLOS, everything is an object (an instance of a class) — the marketplace, the agents, the item descriptions, etc.

The marketplace language is implemented by requiring the agents to support certain methods that can be called on them. All these methods can be called on both buying and selling agents. They are:

- `accept-offer?(agent, from-agent, offer)`
  This method is used to ask `agent` whether or not they accept the offer of `offer` from `from-agent`. `agent` returns either "accepted" or "rejected".
- `what-is-price?(agent, from-agent)`
  This method is called by `from-agent` to ask `agent` what its price is, which is returned. If `agent` is a buying agent, then its price is how much it is willing to pay. If `agent` is a selling agent, then its price is how much it is willing to sell for.
- `what-is-item?(agent, from-agent)`
  This method is called by `from-agent` to ask `agent` what item it is trying to sell or buy, depending on whether `agent` is a buying or selling agent. An item description is returned.

A marketplace object contains buying and selling agents. Agents are added to the marketplace by calling the methods `add-sell-agent` and `add-buy-agent`. When an agent is added to the marketplace, it is notified of agents who are interested in buying (selling) the item it is selling (buying). In addition, the appropriate agents already in the marketplace are notified of the existence of the new agent. The marketplace does this notification by calling the following two methods, which must be supported by all agents:

- `add-potential-customers(sell-agent, potential-customers)`
  Notifies `sell-agent` that the buying agents specified by `potential-customers` want to purchase the type of item it is selling.
- `add-potential-sellers(buy-agent, potential-sellers)`
  Notifies `buy-agent` that the selling agents specified by `potential-sellers` wish to sell the type of item it wants to buy.

In addition, buying and selling agents must also support the methods `remove-potential-sellers` and `remove-potential-customers`, which the marketplace calls to notify agents that other agents are no longer of interest (for instance, because they have terminated or already reached a deal with someone else).

Agents also need to be able to send messages to the marketplace. There are two methods which the marketplace supports:

- `agent-terminated(marketplace, agent)`
  Called by `agent` to notify `marketplace` that it has ceased to exist.
- `deal-made(marketplace, sell-agent, buy-agent, item, price)`

Notifies **marketplace** that **sell-agent** and **buy-agent** have agreed to transact **item** for the given **price**.

The items that are bought and sold are described by feature vectors. These vectors consists of (feature, feature value) pairs. Describing items in this way makes it easy to determine if two item descriptions match.

Conceptually, buying and selling agents in the marketplace are constantly talking to each other, moving from agent to agent, all at the same time. Because we cannot really run the agents in parallel, the marketplace simulates this by implementing a simple scheduling algorithm. The algorithm works as follows. Each agent is allowed exactly one "slice" of execution time per marketplace "cycle". During this slice, an agent can do whatever — talk to other agents, do internal computations, etc. There should be a mechanism that limits how much processing time an agent can consume per slice, but this has not been implemented. The order in which the agents execute per cycle is determined randomly. To execute an agent during its slice, the marketplace calls the method **do_thing**, which all agents must support.

Note that the architecture described is independent of agent implementation. The agents are only required to support a specified interface, i.e. a certain set of methods that can be called on them. What they do within these methods is up to them.

We now describe how selling and buying agents work. We will refer to selling agents; buying agents work in the obviously symmetrical way. (While it is possible to build buying and selling agents which use different strategies, we chose for convenience to have ours use the same framework.)

An agent consists of the following components: control parameters, negotiation history, and internal state. Each of these will be described in turn. In addition, an agent stores the date it was created and a description of what it is selling (or buying).

The control parameters are just the five user-specified parameters described earlier in the paper.

The negotiation history stores each conversation that the agent has had with other agents. It consists of a list of (date, event) pairs, where each event describes the conversation that occured on that particular date. An example conversation is "I offered Agent 3 a price 100. They rejected the offer.", or "Agent 14 asked my asking price. I replied 91." As discussed previously, recording all the conversations that an agent has had can provide useful information to the user.

The internal state of an agent contains information that the agent uses to decide what it will do each slice, i.e. when **do_thing** is called. The internal state contains a list of "potential contacts", which are those agents interested in buying (selling) what that agent is selling (buying). With each potential contact is recorded its last known asking price (what the agent wishes to buy or sell the item for), and whether it has been asked this round (explained later). The internal state also stores the agent's current asking price.

The strategy an agent uses to decide what to do each slice is very simple, and is described below.

1. *Determine the current asking price.*
   The agent lowers (raises) its asking price according to a linear decay (growth) function. When the agent is created, its asking price is the desired price. By the date to sell by, the asking price is the lowest price. At any moment in between, the current asking price can be linearly interpolated.
2. *Decide which agent to talk to.*
   The agent's strategy is to talk to each potential contact exactly once per "round". In other words, an agent will never talk to a given potential contact until it has first talked to all other potential contacts. The algorithm for deciding which potential contact to talk to during that slice works as follows: Consider the potential contacts that have not yet been spoken to in the current round. If all have been spoken to, then consider all the potential contacts. From this set of considered agents, pick one that has never been contacted, or, if all agents under consideration have been contacted, then pick the one whose asking price is the highest (lowest). The idea is to first talk to those agents which seem the most promising — first those who have never been spoken to, and then agents who have indicated a willingness to pay a higher (sell for a lower) price.
3. *Talk to the potential contact.*
   First, the agent offers to sell (buy) the item at its current asking price. If the contacted agent accepts, then the agent's job is done! If the contacted agent rejects the offer, then the agent asks its asking price. This price is recorded for that potential contact, and its asked-this-round flag is set to true.

Experimental Application Domain

To test the Kasbah Prototype, we have come up with a simple experiment: buying and selling playing cards. The idea of the experiment is to have a small group of users buy and sell playing cards using the Prototype's buying and selling agents, rather than conducting the negotiations themselves. The goal for each user is to maximize the value of their hand (as in poker, so a royal flush would be the most valuable). Because this is meant to be a fun experiment, the person with the highest hand at the end will receive a small prize.

At the start of the experiment, each person is given a random set of cards and some play money. They then proceed to buy cards that improve their hand and sell cards that don't, in order to get more money to buy cards. Thus, there is an incentive for people to create both buying and selling agents.

We are presently building a Web interface for the Kasbah Prototype to make it easy to conduct the experiment. Preliminary results are encouraging, and we

expect more results in the near future.

Related Work

Much work has been done over the past few years on intelligent agents. The overall goal is to help users deal with "information and work overload" (Maes 1994), and that it is what we are trying to do with Kasbah.

The notion of autonomous agents is not a new one. It appears extensively throughout computer science literature, in several different contexts. In the field of Distributed AI, agents are entities which collaborate to solve a specific problem (Demazeau and Muller 1990). In Decentralized AI, the focus is more on the interactions of agents with different motivations. The underlying notion, though, is that the agent interaction should further some organizational goals (Demazeau and Muller 1990).

Agents are often seen as a general technique for solving problems, be they very specific (planning the path of a robot) or broader (managing resources). This notion of agents is somewhat different from the one we take, which is more task-oriented. Also, Kasbah's agents not only don't share common goals, they have conflicting aims. This contrasts to a system such as Firefly (Shardanand and Maes 1995), where agents serve individual users (to make music recommendations), yet cooperate and exchange information in a mutually beneficial fashion.

A lot of work has also been done in the area of agent communication. KQML is perhaps the most notable attempt to design a general purpose agent language (Labrou and Finin 1994). We have chosen not to use KQML thus far, since all our agents are local and can easily communicate via a predefined set of methods. For the future, though, we are considering using KQML to easily allow agents designed by outside parties to participate in the marketplace.

As Kasbah's agents evolve and become more advanced, we will want a richer semantics of agent communication to allow more complex and subtle negotiations to take place. The field of speech acts (Winograd and Flores 1986) has investigated such theoretical issues in depth.

Conclusion

Kasbah is a system where users create buying and selling agents to transact goods on their behalf. We have built a simple prototype to test the basic concepts and feasiblity, and are currently conducting more user tests. Though we have only just scratched the surface in terms of making a truly useful system, we are excited about this work and believe it has the chance to fundamentally change the way people buy and sell goods and services in the not-so-distant future.

References

AdWorld, 1996.
http://www.scbe.on.ca/int/adworld.html

Seattle Times/PI, 1996.
http://webster.seatimes.com/classified/index.html

Infomaster, 1995.
http://infomaster.stanford.edu:4000/ASK/RENTAL

Demazeau, Y., and Muller, J. Decentralized Artificial Intelligence. In: Decentralized AI. Eds. Demazeau and Muller. Elsevier Science Publishers, North Holland. 1990.

Geddis, D., Genesereth, M., Keller, A., and Singh., N. 1995. "Infomaster: A Virtual Information System". In Proceedings of CIKM 95 Workshop on Intelligent Information Agents. Baltimore, Maryland.

Labrou, Y., and Finin, T. 1994. "A Semantics apporach for KQML — a general purpose communication language for software agents." In Proceedings of CIKM 94. New York: ACM Press.

Maes, P. 1994. Agents that Reduce Work and Information Overload. *Communication of the ACM* Vol. 37, No. 7. 31-40.

Rosenschein, J., and Zlotkin, G. 1994. *Rules of encounter: designing conventions for automated negotiation among computers.* Cambridge, Mass.: MIT Press.

Shardanand, U., and Maes, P. 1995. "Social Information Filtering: Algorithms for Automating Word of Mouth". In Proceedings of CHI 95 Conference, Denver, Colorado.