BABSy: Basic Agent Framework Billing System*

Rupert Rockinger and Hubert Baumeister

Institut für Informatik
Universität München
Oettingenstr. 67
80538 München, Germany
{rockinge,baumeist}@informatik.uni-muenchen.de

Abstract

The fast growing field of electronic commerce brings today's applications to their limits. Agents are now being used to further automate the purchase processes. Agents already support many stages of the buying cycle, but currently there is only little agent support for accounting and payment. In this paper we describe the design and implementation of BABSy, an accounting system that helps automate payment in an agent-based electronic commerce environment. The implementation was done using the FollowMe agent framework and was tested with the Event Notification pilot application.

Keywords: agents, e-commerce, accounting

1 Introduction

The Internet is a great way of reaching customers all over the world with comparably little effort. No other type of media can offer such global presence at the same low price. This makes electronic commerce ideal for businesses of any size.

However, the Internet's greatest power also turns out to be its biggest problem: the amount of data it contains is almost infinitely vast. Whatever information you are looking for, chances are pretty good it can be found on the Internet. However, to gain access to that data seems to be an extremely arduous task. This is especially a problem for those who want to offer their products online. Due to the number of competitors, being present online does not guarantee being recognized.

Many new ways of bringing order to the chaos and giving structure to the masses of data have been conceived. From setting up search engines to channels. One idea is to use mobile agents to do the awkward work of searching for the user.

This calls for a new type of service called trading. It is provided by a so-called trader that will help agents to locate the providers of the information they are looking for. Any provider within this system registers its services at the trader upon startup. The agents will then contact a trader and find out where the desired services are available. Any findings are reported to their users in a specified way.

Agents also provide a great way to make more customized offerings. Customers could have their own personal assistant. This special agent is equipped with information about the customers, like their whereabouts, preferences, or interests. With this information a company can understand the customers' needs much better.

In addition to searching for a product on behalf of a user, agents can take over many other steps in the purchase process. For example, an agent can help identify the need for a product. It can deal with product and merchant brokering and handle the negotiation with the merchant. Since the products bought online often are information goods, the agent can even take over the delivery of the product.

What is missing to fully automate the entire process is an accounting system so that the agent can also purchase the product. This can become important for users that are infrequently online. Their agents can then execute

*submitted to MAMA'2000

purchases that are time-critical like bargains right away without waiting for a response from the user.

A fully automated process can also help busy users who want to spend as little time as possible with a single purchase. An agent capable of accounting can return the needed product without requiring any further attention by the user. This does not mean that agents are supposed to buy anything they like. The users still specify what their agents are supposed to acquire.

In this paper the design and implementation of the basic agent framework billing system (BABSy) is described. In Sect. 2 we give an overview about payment models in everyday life. Sect. 3 shows in what parts of the buying process agents are already being used today, and where BABSy takes its place in this context. Sec. 4 gives an overview over the FollowMe framework and the Event Notification pilot application. Then, Sec. 5 describes the architecture of BABSy and its implementation using the FollowMe framework. And finally, in Sec. 6 the experiences with this project are summarized and some perspectives for future extension are presented.

2 Accounting models in everyday life

Just about everyone of us has been in the role of a customer many times. And so we all have encountered many models of payment already. Just think of the rent for your apartment, the monthly payments for your brand new car or simply the purchase of a new pair of shoes at the mall.

A brief overview over the most important payment models is given in Fig. 1. First, a distinction is made between those services that are free to the users and those the users have to pay for themselves.

The term 'service' is used here instead of merchandise since in the electronic world goods are often virtual. Buying information for example can be considered the service of supplying this data. Even the purchase of hardware goods online can be described as the service that will deliver those items to your doorstep.

Free services

Those services free to the user are either paid by advertisement or by sponsoring. Usually accounting is not needed in these cases if the sponsor compensates all emerging costs. Only if the sponsor wants to just cover certain services, an accounting system becomes necessary.

Paid services

To calculate the price for a service one can either measure the amount of that service or the time the service is being used. There is no strict border between the two categories and some payment methods might actually fall under both. So the accounting for services that the user has to pay for can be further divided into the following categories:

Paid per use

This is the standard case where services are paid per use. Whereas 'per use' can either stand for a single (atomic) use or for multiple uses.

Paid per time

This payment method represents all sorts of subscriptions. There are two aspects to these subscriptions that must not be confused. Consider your daily newspaper. If you have a subscription for it, it will be delivered to your house every morning. But still you don't have to pay for it every morning. Rather you just pay once for the entire month. This is the first aspect to a subscription: a single payment that will cover for services over a certain amount of time.

However, after the month is over the newspaper will still be brought to your home and the publisher will simply send you a new bill. This is an entirely different aspect: the automatic renewal of a subscription.

Since in agent frameworks the user's agent will always initiate the services this type of automatic renewal is not very appropriate. Indeed it seems more plausible to simply charge the user again when he once more requests the service after his subscription has expired. The advantage here is that the customers initiates all the communication. A quick example will explain: A store that rents videos charges an annual membership fee. Instead of sending bills to all their customers every year they simply charge the fee whenever the customer wants to rent a video after the old membership fee has expired. The point is that the merchant does not need to initiate any sort of communication. Users will simply be asked to pay their bills when they come around again anyway.

Mixed versions

Finally the above versions can be mixed in just about any possible way. Sometimes this becomes necessary when a basic supply charge is combined with a "per use" fee. This system is often used for goods that the customer can use on demand. The basic charge is for making the supply possible. The use-dependent fee relies on the actually used amount of service. Simple examples here are the phone and power lines.

3 State of the art in commercial agents

With all the benefits of agent technology in mind, one would expect that there are many solutions available for

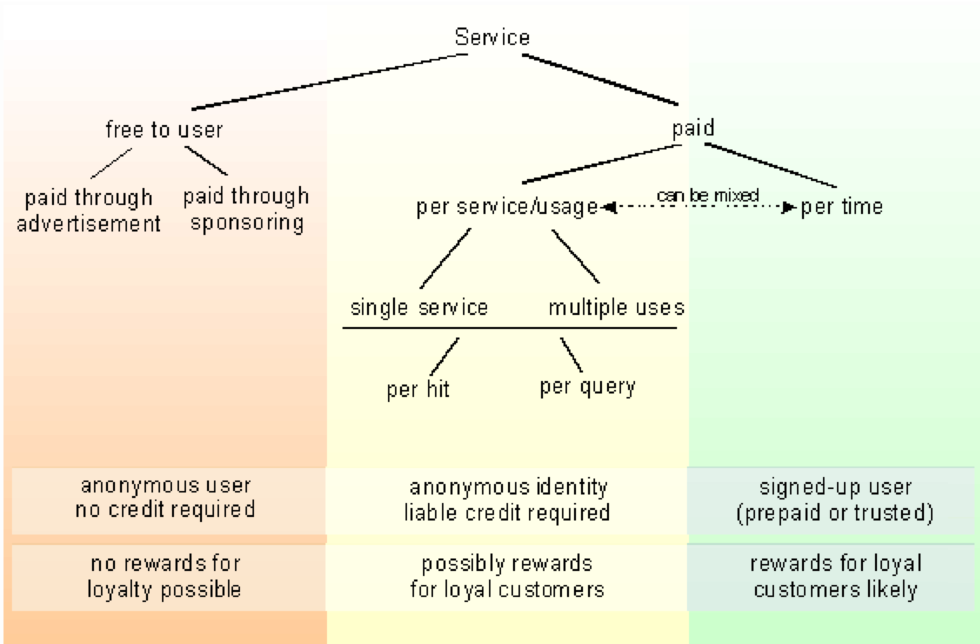

Figure 1: Payment models

accounting in commercial agent frameworks. This, however, is not the case at this time. Mostly this is due to the fact that agents cover only a certain aspect of the buying behavior model.

3.1 The buying behavior model

According to [4] there are several descriptive theories and models on buying behavior. For example the Nicosia model, the Howard-Sheth model, the Engel-Blackwell model, the Bettman information-processing model, and the Andreasen model. Whatever differences these models may have, they all agree on a list of six fundamental stages of the buying process. These stages also help to categorize existing agent-mediated electronic commerce systems (adapted from [4]):

1. Need Identification: This stage characterizes the buyer becoming aware of some unmet need. Within this stage, the buyer can be stimulated through product information.

2. Product Brokering: This stage comprises the retrieval of information to help determine what to buy. This encompasses the evaluation of product alternatives based on buyer-provided criteria. The result of this stage is the "consideration set" of products.

3. Merchant Brokering: This stage combines the "consideration set" from the previous stage with merchant-specific information to help determine who to buy from. This includes the evaluation of merchant alternatives based on buyer-provided criteria (e.g. price, warranty, availability, delivery time, reputation, etc.)

4. Negotiation: This stage is about how to settle on the terms of the transaction. Negotiation varies in duration and complexity depending on the market. In traditional retail markets, prices and other aspects of the transaction are often fixed, leaving no room for negotiation. In other markets (e.g., stocks, automobile, fine art, local markets, etc.), the negotiation of price or other aspects of the deal are integral to the buying process.

5. Purchase and Delivery: The purchase and delivery of a product can either signal the termination of the negotiation stage or occur sometime afterwards (in either order). In some cases, the available payment (e.g. cash only) or delivery options can influence product and merchant brokering.

6. Product Service and Evaluation: This post-purchase stage involves product services, customer service, and an evaluation of the satisfaction of the overall buying experience and decision.

These stages are, of course, only a simplification of the more complex behaviors. They may often overlap, and migration from one to another can be non-linear and iterative. Still, they provide a means to identify the roles of agents as mediators in electronic commerce.

Obviously an accounting system applies to stage five. This is where the actual purchase takes place and the money is transferred. This is also the stage addressed by the framework developed in this paper.

There is no automated system today that covers all these stages. Here are some representative examples of agents used in current commercial applications. They assist or cover various stages of the buying process.

Notification agents support the need identification phase. They can be especially useful for repetitive or predictable purchases, like e.g. supplies. Amazon.com is an online bookstore that offers its customers a notification agent called "Eyes" which monitors the catalog of books for sale and notifies the user when a desired event occurs. Typically when new books become available from a certain author or in a certain category. On FastParts.com a similar system is used to buy and sell semiconductors and other electronic parts. The agent called "AutoWatch" will notify the user when certain parts become available. "Cool Notify"[1] is a part of Excite.com's Classifieds2000 service and will contact the user whenever the desired items appear in the Classifieds database.

Recommendation agents support the product brokering and help the users find a product that satisfies their needs. These agents usually operate on a personal profile describing the users' habits and interests. This information can also be obtained by direct input of the user. Based on this data they will then come up with product suggestions. Firefly[2] uses a technique called collaborative filtering. This means that it recommends products via an automated "word of mouth" recommendation mechanism. The user will rank a group of products and the system will search its database for another user with similar preferences. Products that have ranked high in this other user's opinion are then presented to the first.

Once the product is identified, the *merchant brokering agents* will help find the merchant offering it with the best conditions. This is not always the lowest price, though. Pricewatch[3] for example will find computer parts sold online. It provides the user with as much information as possible, like e.g. extra shipping times and charges, applying taxes and warranty information. A similar thing for books does DealPilot.com (formerly acses.com). It lets the user search for books by title, ISBN, author or keyword. Once the book is identified it will search for it in online bookstores. The user will then get a hot-linked table with offerings.

Negotiation agents. In retail, prices are usually fixed and negotiation isn't necessary or even possible. This feature is more important in business to business transactions.

However, typical situations where customers have to deal with price negotiation are auctions. The University of Michigan has developed a general purpose Internet auction server called AuctionBot. Users can create their own auctions to sell products. The necessary parameters like clearing times, the numbers of sellers permitted etc. can be customized.

One of the first online agent systems for negotiating consumer products is Kasbah from the MIT Media Lab [3]. This software was developed in 1996 and is now superseded by Market Maker[4], also a MIT Media Lab project. These systems provide a so called marketplace that agents use to sell and buy products to and from other users. After logging on, users can create an agent designated to carry out their orders. The agents can be adjusted in what they look for and how they behave in negotiations. Users will then be notified when a deal is done. They can rank their business partners in how reliable they are, and their agents will receive a reputation visible to others.

What we have seen so far are solutions that only cover one aspect of the buying process. Tete-a-Tete (T@T)[5], once again a MIT Media Lab project, handles everything from product brokering to negotiations covering phases two to four. It is yet to be seen how a research project like T@T will be applied in real world applications, but the perspectives are very promising [5].

As we have seen these agents are mostly used to support the user in stages two through four of the buying process. All of these agents have the following in common: They don't support all of the necessary stages to fully automate the process, and in particular they do not support the stage of payment within their system. Therefore any accounting system used before the introduction of agent technology will still be applied. In other words these agents only help utilizing already existing electronic commerce applications. They simple provide a friendlier front-end to the user, while payment transactions are still kept out of the system.

[1] `http://www.classifieds2000.com/cgi-cls/display.exe?c2k+StartCoolNotify`
[2] `http://www.firefly.com/`
[3] `http://www.pricewatch.com/`
[4] `http://ecommerce.media.mit.edu/maker/maker.htm`
[5] `http://ecommerce.media.mit.edu/tete-a-tete/`

4 The FollowMe framework

This section gives a brief introduction to the FollowMe agent framework and its use in the implemented Event Notification pilot application. More detailed information can be found at [6].

The agent framework FollowMe was built within a project funded by the EU by APM Ltd and the University of England in the UK, Inria and TCM in France, and FAST e.V. in Germany [7]. One of its objectives is to "exploit mobile agent technology to develop a support infrastructure for information consumers" [8]. FollowMe by itself is no application, but an efficient and easy-to-use infrastructure for building agent applications. FollowMe is built upon a mobile agent infrastructure called the "Mobile Object Workbench" [2]. This infrastructure hosts several components:

Autonomous Agents. These agents are autonomous in how they react to external events. Only their destruction can be forced by the place they reside in. Their mission is described in agent-scripts, and they hold a profile with their characterizing data.

Personal Profiles are pieces of information associated with a user. They are held by a special kind of agent called personal assistant. Any information from the user's name to "how she can be reached at a certain time" can be stored here.

The *Information Space* provides the necessary facilities for persistent storage of objects within the framework's distributed environment. It works much like a standard dictionary to the outside, but information stored here is available across the system and won't be lost if a component fails.

Service Interaction interfaces act as service proxies to allow agents to interact with non-mobile services.

The *Service Deployment component* applies data mining techniques to optimize the availability of objects on different locations.

The *User Access* provides an interface for users to contact their agent or, in most cases, the other way around. This component is specially designed to support all kinds of output media such as a web browser, personal digital assistants (PDAs), fax machines, or even phones. For interactive communication, a Java enabled device is required.

Currently there are two pilot applications built upon these components: The already mentioned event notification and a stock portfolio agent. The later can help users keep track of their stock values and will report when certain limits are exceeded.

[6] `http://hyperwave.fast.de/FollowMe/`

In the current version of the Event Notification application allows the users to keep a personal profile of name, address, and information about how they can be reached (e.g. phone number, e-mail, etc.). The users can create personalized task agents that will search for events, like cinema, concerts, e.t.c., matching certain categories and that will take place in a desired area at a specified time. Also, the frequency of the reports can be adjusted and how they will be delivered.

For interaction with the agents a web browser interface via the User Access component is used. Reports can be issued by e-mail, fax or Small Message Service (SMS).

5 Accounting architecture for agent frameworks

In this section we give an overview over the basic architecture of BABSy and its implementation within the FollowMe agent framework. A more detailed description can be found in Rockinger [6].

To identify the participating agents let's take a look at a conventional transaction. Say, for example, you buy a meal at your favorite restaurant. Two parties are obvious: you, the paying customer, and the selling restaurant owner. The third party in the interaction is a banking service that provides the guarantee for the payment used. When paying cash this is the central bank of a country that guarantees the value of your bank notes; for credit card payment this is the credit card company; and when paying by check this will be bank that issued the check. Either way there is always a third party involved that both others must recognize and trust. So within the agent framework three independent components are needed. One for the provider, one for the customer and a third that represents a banking service.

- The Service Agent represents the business side and deals with requests from the outside. It holds the information about the pricing of offered goods and available subscriptions to its services.
- The Accounting Agent can be considered the bank within the system. It keeps track of the payments made towards the service providers. It will also notify them when a transaction in their favor has been booked.
- An agent within the system that is associated with the customer. It keeps the user's preferences about handling payments for the requested services. It will typically be a companion of the user's Personal Assistant; however, this is not strictly necessary and therefore the Personal Assistant is not considered part of the accounting system. This

also helps minimize adjustments in existing Personal Assistants and greatly enhances the reusability of the accounting system.

5.1 Interaction modeling

So far only the components of the system have been described. What's essential, though, is the interaction of those components. The main interaction of the system is issuing a request to the service agent. Therefore we shall have a closer look into this interaction.

All three parties cooperating in a purchase need control over the transaction. In particular every single agent needs some sort of veto capability to be able to stop the transaction.

A circular information processing path becomes apparent: Consider a shopping trip with your credit card. First you will ask the clerk for whatever you want to buy. He will then tell you the price and you can choose whether you think it is appropriate. If it is, you will hand him your credit card and he will try get the confirmation of you card company that this transaction is valid and booked. Unless you exceeded your limit the clerk will get this verification and hand you your merchandise.

Fig. 2 shows the simplified object model with additional information about information flow. The steps one through five define the normal flow of information for a "service" use case. That is, if the request is completed and carried out - the primary scenario.

In step number one the User Agent (customer) issues a request for service by contacting the respective service agent. The Service Agent will then send a payment request to the User Agent (step number two), which in turn forwards this request to its Accounting Agent (step number three). After the transaction is booked the Service Agent is informed (step number four). Finally the service is carried out (step number five). Note that the action of contacting the service provider and getting the results have no numbers in this sequence. This is due to the fact that they can indeed happen at just about any time desired. A closer look reveals that there are two cases that are of interest:

- The request (e.g. a query to a database) is very cost intensive. The Service Agent then will not start the actual request to the provider until the expenses for it are covered. In this case the request to the provider will happen after the Account Agent has notified the Service Agent of the completed transaction (immediately before step five). This lazy evaluation is always the preferred option for the provider since operating costs are kept to a minimum.
- The results of the query might be needed, however, to calculate the price for this request. This applies for all pricing models that use "per hit" as a factor. In this case the entire request has to be carried out by the time the payment request to the User Agent is issued (between steps one and two).

These are even more cases which will mix both variations:

- If the request doesn't involve the provider, the steps can be omitted entirely. One example would be if the customer only wants to make a subscription to a service. This also requires a payment, but the only result is that the user is noted as subscriber to the service within the agent.
- On the other hand a "per hit" pricing model might apply, but the requests are too cost intensive to be carried out before the payment is conducted. Here the Service Agent might check for availability of a good (and its quantity) to calculate the price, but hold the actual request until the payment has been confirmed. One example would be a test query to the database that is used in combination with heuristics to calculate the price for the actual query.

The request typically comes from the user directly, a User Agent (e.g. Personal Assistant) or is triggered by an alarm. The answer is usually presented to the user in a desired fashion. The query is representative for a contact to the service provider. In fact, it can be any kind of request.

On this level of abstraction we assume that all communications reach their respective recipients and that the agents work properly. But still, as pointed out earlier, any party may choose to stop the transaction.

The most common reason for the UserAgent to stop a transaction is that the he decides that the price exceeds the limit set by the user. It could also mean that the agent has gotten a better offer from another service provider.

Another reason is that the bank aborts the transaction. Typically this happens when the Accounting Agent finds that a user surpasses his/her credit limit. It can also happen when the User Agent is not registered at the Accounting Agent (i.e. has no account).

5.2 Implementation within the FollowMe framework

The FollowMe agent framework is implemented in Java which provides a great platform for this type of application. It is object oriented and has built in functionality

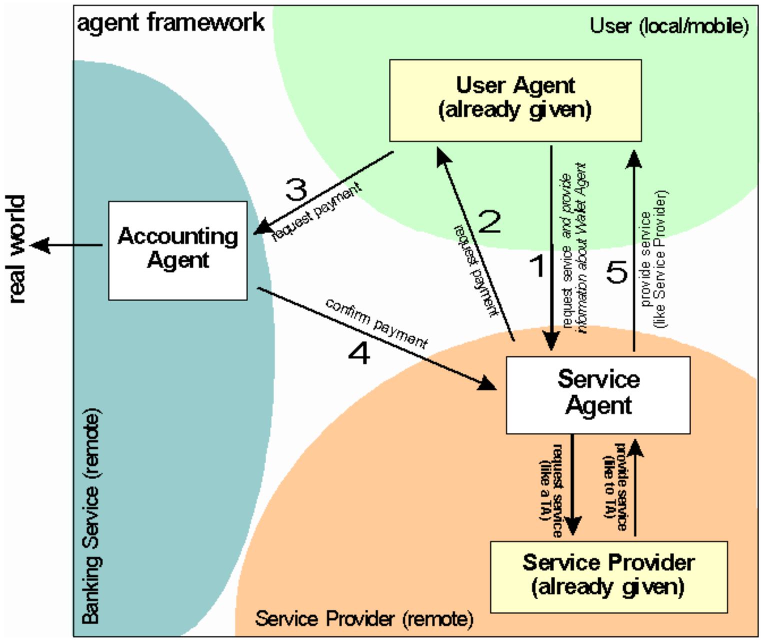

Figure 2: Information flow in the simple object model

for networking and remote method invocation (RMI). In addition, it is platform independent and therefore ideal for code that moves in a heterogeneous network.

For the communication between the individual agents the FIPA agent communication language (ACL) is used [1]. ACL defines a standard of how messages between agents are built. It does not imply the mechanisms used to transport the messages nor does it imply any particular content language for the messages.

ACL only defines the structure of a message. It does not define the structure of the content of the message. This means that in the content attribute of an ACL message any arbitrary expression can be used. In order to be able to understand the message content without aids of artificial intelligence a content language must also be defined. The content language used in BABSy is defined as very simple functional expressions of the form `keyword(par1,..,parn)`. Example of keywords are `requestService`, to request a service, and `confirmPayment`, to confirm a payment.

BABSy also makes extensive use of the components provided by the FollowMe framework. In particular the Information Space and the User Access are utilized. Furthermore the trading mechanism of the pilot application provides a whitebook service and the basic means for communicative acts. The ACLmessage class builds on these capabilities to transfer messages.

The graphical user interfaces were implemented using the Java Abstract Windowing Toolkit (AWT). This provides a fast and easy way to implement simple windows and dialogs. It is also part of the core language and is therefore available on all systems that support Java.

6 Summary

In this paper the design of the accounting system BABSy for services in an agent community was presented. Though the implementation focuses on the Event Notification pilot application of the FollowMe framework it can be easily applied to other applications as well.

For the first time, BABSy brings together all the main steps of buying behavior, from product and merchant brokering over negotiation to the purchase itself. This can greatly improve the value of electronic commerce applications in the future.

The design of the BABSy core components covers the most important payment models while still leaving room for further functionality. Most components can be extended for much greater complexity if desired.

To fully exploit the features of BABSy, TaskAgents are required that offer better communicational interfaces. In future projects several providers and banking services can compete with each other to offer the users a greater variety of options.

The FollowMe framework provides powerful tools for distributed applications. Setting up communication through ACL messages was fairly easy with a given trader mechanism and transparent remote method invocation.

While BABSy in its current implementation makes heavy use of the FollowMe infrastructure it is no major difficulty to exchange the used components. The agents of BABSy can be put to work with any other agents capable of ACL.

References

[1] Fipa 97 specification, version 2.0, part 2: Agent communication language. `http://www.fipa.org/`, October 1998.

[2] M. Bursell, R. Hayton, D. Donaldson, and A. Herbert. A mobile object workbench. Submission to the Mobile Agent Workshop '98, Stuttgart.

[3] P. Maes and A. Chavez. Kasbah: An agent marketplace for buying and selling goods. In *Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology*, London, UK, April 1996.

[4] P. Maes, R. H. Guttmann, and A. G. Moukas. Agents that buy and sell: Transforming commerce as we know it. *Communications of the ACM*, 43(3), March 1999.

[5] P. Patton. Buy here, and we'll tell you what you like. *The New York Times*, September22nd 1999.

[6] R. Rockinger. Design and implementation of accounting models for services in agent-based information systems. Master's thesis, Institut für Informatik, Universität München, München, Germany, September 1999.

[7] A. Showalter. Frictionless commerce gets premiere position on lycos. *Boston Software News*, September 1999.

[8] H.-G. Stein and M. Breu. An architecture for automated information retrieval. In *Proceedings of the AAAI Spring Symposium on Intelligent Agents in Cyberspace*, Stanford, California, March 1999.