Information Management at a Bank using Agents: Theory and Practice

F. Dignum

Faculty of Mathematics & Computing Science
Eindhoven University of Technology
email: dignum@win.tue.nl

Abstract

This paper describes a theoretical architecture of a multi agent system that offers an integrated access to the data and knowledge that is available in local applications. By integrating the local data management activities into one transparent information distribution process, modern organizations can offer better support for its workers in the execution and coordination of their work activities.

After the theoretical discussion we show how this architecture was implemented at a bank and which alterations were forced on the theoretical model due to practical considerations.

1 Introduction

Because the amount of information and the number of information sources have increased dramatically in recent years, it becomes increasingly difficult for individual professionals to filter out the right information. The advent of Internet availability within companies discloses a vast amount of information directly to the professionals. While the information filtering was first done by special departments in the hierarchical structure of a company, nowadays the individual stands for the difficult task to perform the filtering himself.

Many attempts have been made to provide Internet and Intranet users with tools that aid them in finding valuable information in the many gigabytes of data they have access to. Most of these attempts have resulted in content-based search engines like Alta Vista, Excite and Infoseek. However, these tools are geared towards finding as much information that is related to the keyword for which a search is performed as possible. This means that a search on e.g. the keyword "Java" on the WWW leads to almost 10 million answers. A way to filter these answers is by using user profiles. A user profile is usually nothing more than a set of keywords in which the user is interested. These keywords are "added" to any search of the user to filter out only those items that are of interest for the user. The user profiles can be static or dynamic. In the first case they are defined once and can only be changed by an explicit update from the user. The dynamic user profiles use some type of learning algorithm to update the profile either through explicit feedback of the user or through observing the behaviour of the user. See [2,6,16] for some examples of algorithms that learn user profiles. Most of these filtering mechanisms only work when all the information comes from the same source and is compatible according to some metric, like texts from the WWW. Within companies this almost never is the case. Usually other sources such as company databases, reports and special services like Gartner, Forrester and NewsEdge are also used. Because the items within these sources are usually not directly accessible for the search and profile programs the information that is taken from these sources cannot be used directly to update the profile.

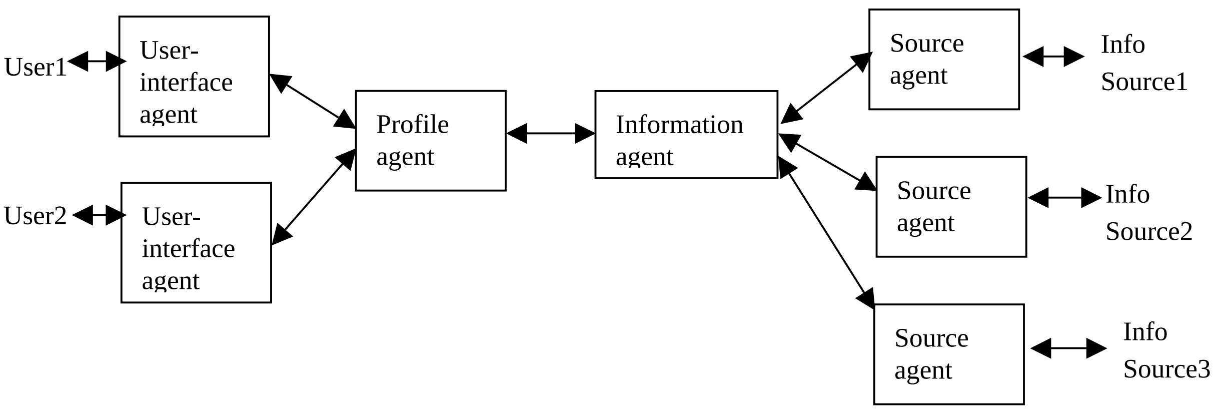
In this paper we will report on a project in which we tried to implement an information management system for the library of a large bank. The library is responsible for getting information for all employees of the bank. For this purpose they maintain a collection of books of interest for finance, but also yearly reports of many companies and reports from agencies such as Gartner and Forrester. In order to couple the heterogeneous information sources and filter the information according to some user profile we used a multi-agent architecture. Much research has been done on combining autonomous and heterogeneous information in the area of cooperative information systems (See e.g. [4,10,17]). However, most of this research assumes that the data is structured (in databases). In our case this is not true. Many sources contain texts that are more or less structured. We therefore have to combine the research done in information filtering with that performed in cooperative information systems.

Roughly the task of getting the right information for the user is split up into two parts. One part maintains the user preferences and interests. The other part integrates the information sources and maintains some type of index on the information sources.

We have chosen to implement the system using a multi-agent system for several reasons. The most important reason is that there are several independent parts to be distinguished in the overall system. Each part has its own goal and is relatively autonomous. The paper is not about the development of a new agent theory. We have used existing theories and agent libraries as far as possible. (For an overview of agent theories see e.g. [22]). The main contribution lies in the combination of agents that is used and the roles they fulfill in the information management and the influence of some practical constraints on this model. In the next section we will describe this multi-agent architecture in more detail. In subsequent sections we will describe the different types of agents that play a role in our model. After the description of this model we describe several practical considerations that had to be taken into account when we implemented the model. The article is concluded in section 6 with some conclusions and prospects on future research.

2 The multi agent system model

The architecture of our multi-agent system is depicted in the following figure.



From the figure it can be seen that there are two types of agents that are geared towards the user. The user agent wants to get only the right information for its user. It then tries to present this information in the optimal way for its own user. The agent can in principle show intelligent behaviour in two aspects. First it can maintain an optimal user profile by monitoring the user's responses to the information that she gets. Also the agent can try to optimize the presentation of this information for the user. E.g. it may learn that the user prefers to see the information that comes from Gartner before she wants to see the information that comes from the WWW.

The profile agent keeps track of the profiles of *user groups*. It may learn that most users that are interested in "agents" are also interested in "Java". It can communicate this information to the user agents that have "agents" in their user profile but not "Java". The profile agent also communicates with the information agent about profile updates. For instance, it hears from the information agent whether there are any new information sources that might contain new topics that should be added to the user profiles. Finally the profile agent also acts pro-actively in that it will make a selection of new information based on the user profiles. (In the current implementation the user will find new information based on its profile on her homepage each morning.)

The information agent tries to optimize the access to the heterogeneous information sources. It has to maintain a kind of information model that can be matched with the user requests and profiles. The information agent also knows which sources have information about which topics. This can enhance the efficiency of the system considerably.

The information model is translated to each of the different sources by the source agents. E.g. suppose that a user profile is nothing more than a list of keywords. The information model is a list of keywords that appear in all the information sources together. For each information source there will be a translation on how a search on a particular keyword should be performed. This can be a translation from a keyword into another word or phrase, e.g. "agents" becomes "software agents" for some resource. It can also be the translation of the search into a query for a certain system. E.g. a search on "agents" is translated into an SQL query over a relational database in which one of the fields should have value "agents".

As can be seen from the figure above, the multi-agent system can be influenced from two sides. On the one hand there are the requests of information from the users (plus possible feedback on the answers given). On the other hand there are updates on the information sources. The queries from the users lead to reactive behaviour of the system in the sense that the agents will try to gather the best information to fulfill the query of the user as a reaction to the query. On the other hand the system will present the user new information when any of the relevant information sources is updated. Because this action is not triggered by the user it is called pro-active behaviour.

The communication between the agents is relatively simple. The agents only request information and tell information to each other. There are no negotiations. Also the types of cooperation between the different agents are completely fixed through the application. It follows from the above that the communication protocols are very simple. The only messages that have to be connected by an agent are a request and the subsequent answer to that request.

2.1 Related work

Of course, we are not the first to design a multi-agent system for information management. In this section we will briefly discuss some well-known related work and indicate the differences with our own work.

The above architecture is based on a general idea of using *brokers* to mediate between users and information sources. In [21] these brokers are called *mediators*. A mediator provides a uniform interface for a number of heterogeneous databases. The mediator provides the semantic and syntactic translation between the data in the different databases using the schemas of the databases. These mediators are very similar to our information agents, except that we do not presume the information sources to be structured.

The Infomaster [7] also uses a central engine that provides access to several sources that can be either structured (SQL databases) or unstructured (WWW pages). The main limitation of this system is that it requires a complete domain description in order to translate queries to the diverse sources. In our setting this would mean a complete description of the domains of all departments of the bank, which range from technical IT issues through management literature to (of course) financial issues. This was not feasible at this moment.

In the Infosleuth project [1] also uses a multi-agent system. It incorporates broker agents that keep track of which agents provide which services. The ontology of the information is maintained by specialized ontology agents. We have combined the broker agents and ontology agents in the information agents. The advantage is that the ontology can be used to find the right source for a query. This is not dependent on the way the services are advertised by the source agent, as is the case with a separate broker agent that maintains the services provided by agents.

Another project that uses a type of mediator structure and several types of agents is described in [15]. In this paper a complete architecture is described for an enterprise information system. However, it does not provide an implementation.

At last we should mention the RETSINA project [18]. The RETSINA architecture is a general architecture for multi-agent systems that use brokers and matchmakers. The main difference from our system is that they suppose the system to be completely open. Therefore, some emphasis is placed on the advertisement and communication structure within these systems. In our case these functions are quite simple, because the agents work in a controlled environment. I.e. within one company.

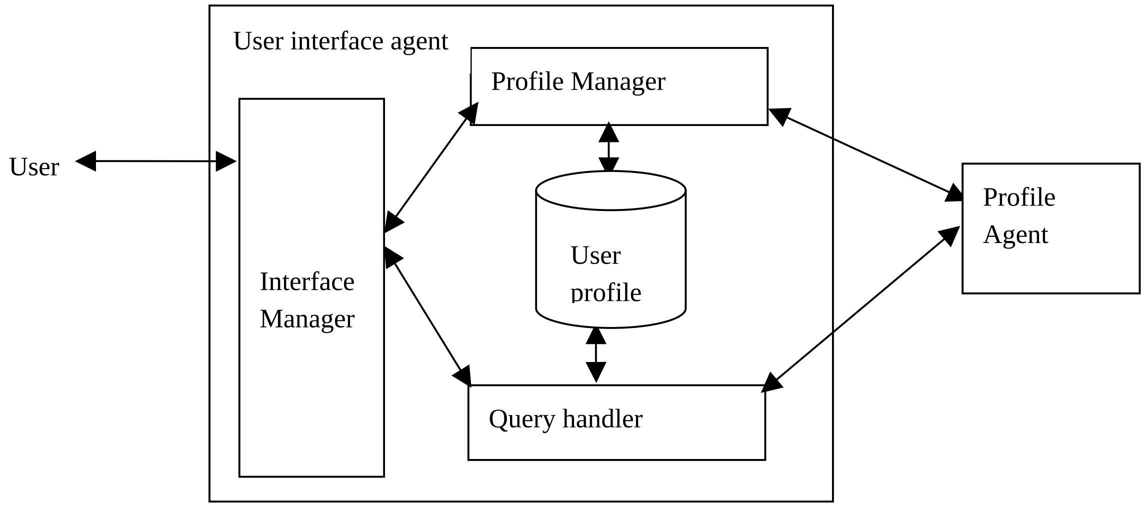
One of the main differences of our architecture with those described above is the use of profile agents. It takes advantage of the fact that users can be grouped and profiles of these user groups can be maintained. Some interface agents on the WWW also try to categorize a user into a certain group. However, these groups are completely based on the similarities in the behaviour of the different users. E.g. buying the same type of articles at a Web site. The groups in our setting are not merely formed on basis of similar past behaviour but also on the basis of departmental structure of the company and project groups within the company. I.e. a user can be categorized on the basis of the type of work that is performed within its department (e.g. investment banking) but also on the basis of her function within the department (e.g. evaluating electronic commerce opportunities), which is often linked to persons performing similar functions in other departments. There can be an electronic commerce workgroup over several departments looking at all different aspects of electronic commerce. These organizational groups are taken as a basis to group users and form user group profiles. They can be refined and altered based on the history of the users, but give a strong basis to start with.

3 The user interface and profile agents

The user interface agents and the profile agents are taking care of the user side of the system. We will discuss each of them in turn.

3.1 The user interface agent

The architecture of the user interface agents mainly consists of two parts: the profile manager and the query handler. All actual communication with the user is handled by the interface manager. The following figure denotes the architecture plus the communication between the components.



The user profile consists of a list of topics and with each topic a rating of the sources and a weighted list of terms that determine the most interesting articles for the user within this topic. This is based on a system of Bayesian classifiers ([6]). The user profile is used to extend a query before it is send through, but also to filter the answers before presenting them to the user. We will come back to this point when describing the query handler.

As said before, the user has two types of dialogue with the user interface agent. Most often the user will pose a query which is forwarded to the query handler by the interface manager. The query handler compares the query with the current user profile. On the basis of the query and the profile a new (extended) query is composed, which is send to the profile agent. At this moment the query handler does not inform the profile manager about the (new) topics in the query. However, registration of new topics is done automatically by the profile agent that registers all queries from all users. This enables an automatic update of the profiles when a certain topic is queried often enough by many users and did not appear in the profiles yet.

The information sources that are connected to the system at the moment are not full hypertexts such as the WWW. They are all indexed through a hierarchy of topics and articles that can be found by either searching for a specific topic or by browsing through this hierarchy. (There are no useful links from one article to other articles.) This organization implies that the queries from the user have to be expressed in terms of the topics that are used to index the information sources. The system cannot find the set of articles that match the query as close as possible based on some metric on the contents of the article as is done in Alta Vista, Excite, etc., because the articles can only be accessed through the topic index. It follows from the above that only the topic part of the user profile is used to extend the query. The other information is of no use for the actual sources.

After some time the answer to this query comes back from the profile agent. The query handler will now filter the answer based on the user profile. The ratings of sources and interests within topics is used to determine the final answer, which is forwarded to the interface manager that will show the answer to the user in a format that is (partially) based on the ratings of the user profile. That is, the order of the items and the number of items on the screen depend on the user profile.

The second type of interaction that the user has with the user interface agent is to (explicitly) update her profile. In this case the interface manager opens up a dialogue screen in which the user can interact with the profile manager. She can add and/or delete topics of interest from the current profile. The profile manager will forward any changes in the profile to the general profile agent such that the new profile can be compared to the profiles of other users. The profile manager also receives information from the profile agent about possible updates on the user profile due to new discovered correlations between items, new topics and/or new information sources.

The way a user interface agents gets a new user profile from the user is by showing the topic hierarchy and letting the user indicate in which topics she thinks she is interested. The most recent articles of those topics are then shown and the user can decide whether she is really interested in these topics or not. This brings us to the last type of interaction with the profile manager.

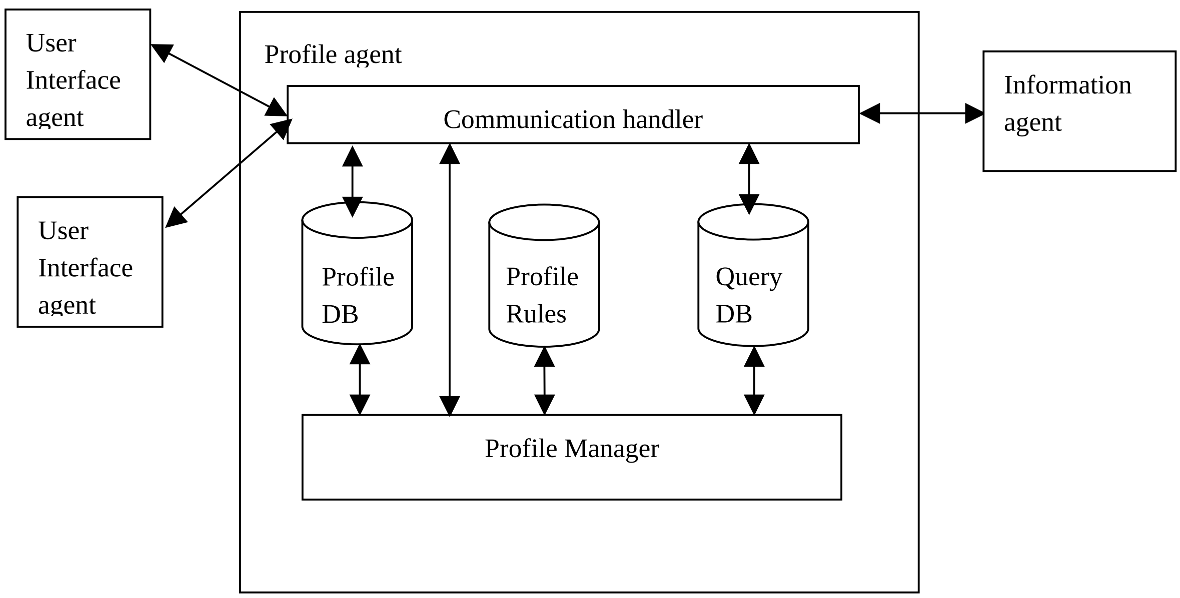
The user can also give feedback on the answers that it gets for each query. Based on this feedback the ratings of the terms that determine the interesting articles within a topic for this user are adjusted. Also the ratings of the sources themselves can be adjusted through the feedback of the user.

Besides this reactive behaviour of the user interface agent it also exhibits some pro-active behaviour. Based on the current profile the query handler will (periodically) compose queries to trace new information that fits with the user profile. At this moment every morning the user will automatically get updates on all the changes in the information sources that might be of interest to her.

3.2 The profile agent

The main reason to introduce a profile agent is to collect information about profiles from groups of users. By comparing the profiles from many users the profile agent can find out correlations between different topics and interests within topics. The profile agent is an ideal place to locate this type of knowledge about profiles. The queries of all the users also pass through the profile agent. Although this agent does not act directly on these queries it will collect them and compare the queries with the group profile. When it discovers that most users with a certain type of profile search for the same topic it can pro-actively deliver the answer to these queries also to the other users that have that type of profile. For instance, if all users that have "agents" included in their profile start searching for "Java", the profile agent can decide that the profiles should be updated to include "Java" and also send the information on "Java" to all users with "agent" in their profile.

From the functions described above it is clear that the profile agent mainly is a type of filtering mechanism. The architecture of this agent is best described by the information that is kept and the processors that work on them:



In this agent we have an explicit communication handler, whose main task is to forward messages between the user interface agents and the information agent. While forwarding the messages it will record the appropriate content of the messages in the databases that are kept by the profile agent.

The profile manager can be seen as a kind of datamining tool that abstracts interesting correlations out of the databases concerning the user profiles. In fact this forms the pro-active part of this agent. The profile manager continuously tries to find new rules about profiles and will send the results to the appropriate agents. Another use of the databases that are kept by the profile agent is to find out general user interests in specific departments, shifting user interests and users across departments that are interested in the same topics. This information can be used for knowledge management. Based on a perceived shift of user interests certain new information sources can be bought and some old ones might be discontinued. It can also serve to get people with the same interests from all over the company together. (At the moment the first experiments are conducted to find out so called “hot spots”. I.e. topics that are of rising interest. The experiment resembles quite closely the datamining efforts within the Infosleuth project (see [20]). Existing datamining techniques, as described in [9] are used.)

4 The source and information agents

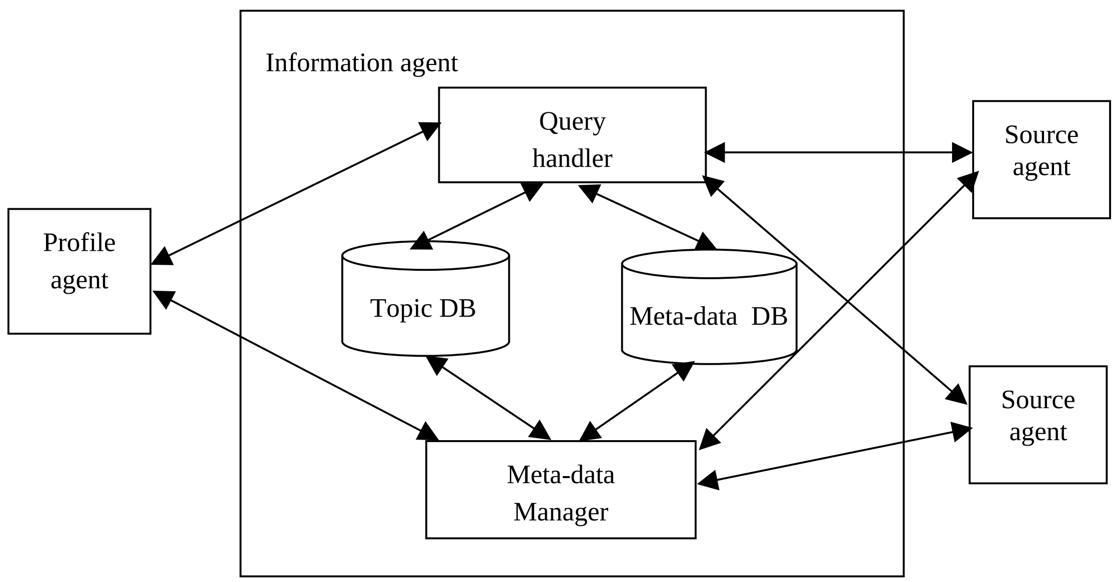
In section 3, we have described the agents that were geared towards the user and the user profile. In this section we describe the information agent and the source agents that are geared towards an optimal organization of the information.

4.1 The information agent

Very simply said the information agent keeps track about which information source contains information about which topics. It also is responsible of merging the information from these heterogeneous sources. Because all the information sources consist of sets of articles, at the moment this just means adding all the articles from the different sources together. However, it happens frequently that exactly the same information is given by different sources and can be reduced to one item. E.g. the stock market prices of some companies at the close of the day, the announcement of a new product, etc.

When the information sources are databases much more can be done about the integration of the data from the diverse sources. See e.g. [10,17] for techniques that can be used in this case.

The architecture of the information agent can be depicted as follows:



When a query arrives the query handler decomposes the query using the meta-data database. At the moment the meta-data base contains a general topic hierarchy. The query consists of a logical expression that indicates which topics should be searched for and which should not be included. E.g.

"(Agents and Java) or (Agents and Internet) and not(travel agents)"

The query handler will look in the topic database for each of the topics occurring in the query which sources have information on that topic. It then will intersect the sources that have information about "Agents" with those that have information about "Java". Only sources that have information about both are interesting. The same is done with "Agents" and "Internet". The two resulting sets of sources are combined and the query is send to all source agents of these sources. Note that nothing is done with the negation (on "travel agents") at this place. This part is only taken into account by the source agents. The reason is that the other parts of the query restrict the sources that should be searched, while the negation does not exclude any source (yet), because we do not have an index on the sources to check which topics they cover. Only by checking whether a certain source covers only the topic "travel agents" and no other topics from the query can we exclude the source. In practice these sources are usually already excluded by the first part of the query. It is therefore more efficient to only handle the negation at the sources themselves.

The source agents will each send back the answer to the query from their particular source. The query handler will collate all the answers and send it back at one time to the profile agent. The information agent keeps track of the answers of each source in order to update its topic database. If a source did not find any article for certain topics, that information is put in the topic database.

Notice that the query handler should keep track which answer belongs to which query. It also should keep track whether all source agents have answered already. At the moment this is organized very simple. All queries that have not been answered yet are kept in a list. With each query a timer is associated that is set when the query arrives. All answers that come back from the source agents are collated to the query in the list. When the timer of that particular query expires the collated answers are send back to the profile agent. Answers that arrive after this time are just ignored. In a more elaborated version the interaction between the information agent and the source agent could be more sophisticated. For instance, the information agent can ask when an answer can be expected and when answers do not arrive in time it might contact the source agent to find out what happened.

Because at the moment all sources are under control of the company itself (i.e. contracts have been made to assure the availability of the information) and downtimes hardly ever occur, the simple timer scheme works perfect. More elaborate protocols are especially interesting when sources are used that do not reside within the company. Things like payment for each search will then influence the way the information agent will operate. For instance, it will relate the amount of information that a source gives on each search to the price that has to be paid for it. Because the information that is presented to the user is organized to show the source of each item as well, it is possible to use the feedback of the user on the information she gets from each source separate. If the information of a certain source is always rated bad by all users the organization can decide to discontinue the services of that source.

One function that is often mentioned for the information agent, in its capacity of being a broker between users and information, is to pro-actively search for new information sources. In the setting of the bank, this is not possible and even not desirable. The bank wants to have some control about the information that is used to make decisions. Therefore it selects reliable sources for its information. Suppose a dealer in the dealing room closes a transaction based on information that comes from an obscure web site then the risk of the transaction becomes far to big. It is essential that all information can be trusted upon and thus the sources limited to trusted sources.

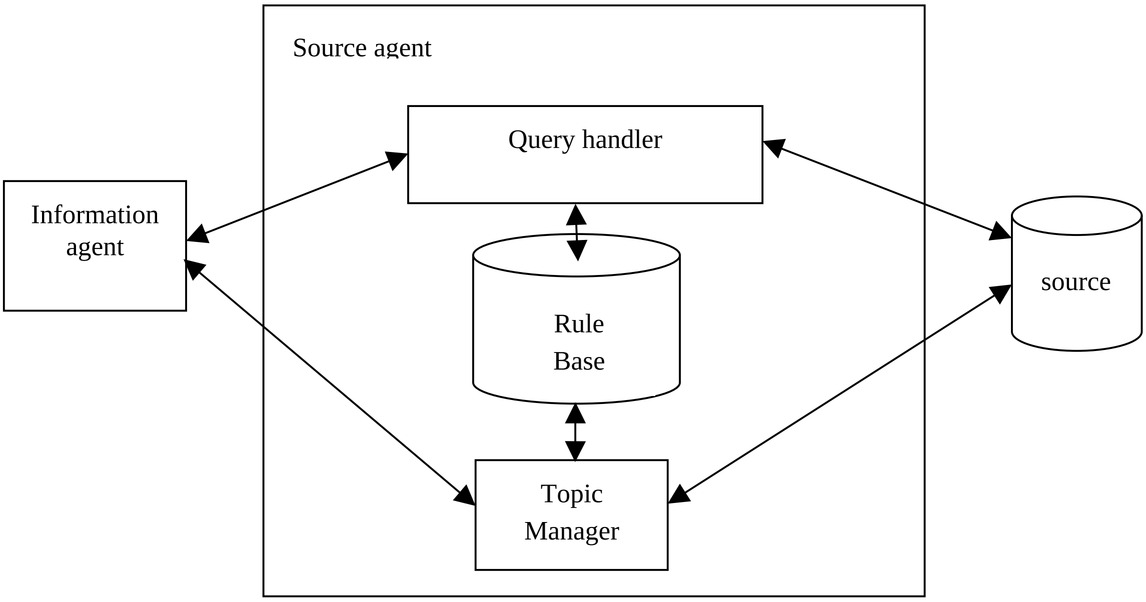
The meta-data manager maintains the meta-database (i.e. the topic hierarchy). It will update the hierarchy when a new source is registered or when a registered source informs it that it contains information about a new topic. In both cases the source agent will send a message to the information agent. Of course the topic database will also be updated in both cases to include the new source or the new topic.

In case a new topic is registered, a message is send to all source agents that already cover related topics to find out whether their source also contains information pertaining to this topic. Related topics in this case means topics that are siblings or parents of the new topic in the general topic hierarchy. For instance, a source might indicate that it contains the new topic "electronic commerce". Other sources might also have articles about electronic commerce, but have indexed those under the topic "EDI". We will come back on how this case is handled when we discuss the source agents.

Finally the meta-database manager will inform the profile agent about any new topics that are part of the topic hierarchy and can also be included in the user profiles. The profile agent will forward a message to the user interface agent that can alert the user of new topics and/or new sources.

4.2 The source agent

The last type of agent in our system is the source agent. Its architecture is depicted in the following figure:



As can be seen from the figure above the source agent does not form a part of the source itself. That is, the source agent cannot determine the internal organization of the source. It functions as a wrapper that provides a uniform interface towards the system for all sources. For instance, one of the sources that is used at the moment is the Gartner on-line service. The source agent provides the translation of queries from the system to a format that can be processed within the Gartner service. The answers are "translated" back to a format that can be handled by the other agents. In our case there is no real translation back. The results of a query are always a set of articles. The source agent will collect the URL's of these articles and send them back. Only the user interface agent will fetch the actual articles when it determines the format of the presentation of the answer to the user. The user interface agent will also use the actual texts to adjust the user profile based on the feedback of the user.

In [19], E. Verharen and F. Dignum described an architecture and implementation of a cooperative information agent (CIA) that fulfills the same functions as the source agents in this system. We did not make use of this architecture because these CIA’s are meant to function mainly between organizations. Much emphasis is put on establishing contracts and communication between the agents, because there is no central information agent in that case. In the present situation no need for communication between the source agents exists, because all communication can be directed (naturally) through the information agent.

The rule base contains the rules that are used for the translation. They form a kind of thesaurus. The topics that are used by the system are transformed into terms that match the specific terms used in this source. E.g. the term "vacation" is rephrased as "holiday". It can also be that a combination of terms is translated to a single one. E.g. "software agent" is translated to "agent", because this source only has information about agents and does not make any further distinction. Also there are rules that add terms to a phrase. E.g. "Java" is translated to "Java and software", because the source also contains information about the island "Java".

Finally the rule base contains some rules that translate the syntax of the queries into the proper format. Most important the "and", "or" and "not" have to be translated into the proper symbols.

The only pro-active behaviour of the source agent consists of monitoring the source for new topics. At the moment the source agent does not check on updates of the information of the source. It only does that on the basis of (indirect) requests from the users or the pro-active behaviour of the user interface agents. The source agent does however check whether the topic hierarchy of its source is updated. If it detects this it will update its rule base and inform the information agent of the change if it involves new topics. Besides the new topic itself, the source agent will also send a list of terms that it found most distinctive in the articles that were classified under this new topic.

As said before the source agent also gets informed by the information agent when a new topic is introduced in the system's topic hierarchy. The source agent has to find out whether its source contains information about this topic and if it does report this back to the information agent and add some rules to the rule base on how to search for this topic.

The easiest case is when the source already uses the new topic internally. In this case the source agent probably has some rules that translate other terms to some phrase that also uses this topic already. Nothing has to be changed in this case.

The other case is when the topic was not used by the source yet. In this case the source agent will do a complete search on the source information and checks whether it finds articles that are close enough to the terms that were identified with the new topic. Close enough means in this case that enough of these terms appear in the text to classify it as belonging to the topic. Once all the articles are found, rules can be established to use the new topic. First it is registered for each article under which topic it is registered. Then a query is composed that will deliver the smallest set of articles containing at least the articles in this set. The rule then translates the new topic into this query.

5 Practical considerations

In the previous sections we have sketched the architecture of the information management system as it was developed in theory. In this section we will discuss some practical considerations that influenced the architecture as sketched above and also some points that influenced the implementation of the system.

5.1 Architecture

The main question is the following:

“Do we need four types of agents and do we need separate agents for all the users and resources, while having a central profile agent and information agent?”

Source agents. From a practical point of view it is convenient to have a separate source agent for each source. The main reason is that the translation of the source format towards the system can be localized. Therefore it is easy to include new sources one by one during the project and also to exclude sources that are no longer interesting.

Information agent. Do we need a central information agent? First of all we need a place where we can locate the information about which source should be consulted about which topics. As in many other systems there is a clear need for this brokering function. Because the sources used are relatively uniform at the moment, it seems best to have one central agent for this broker function. In a later stage when other sources like news about stock markets or other financial news (like e.g. CBS MarketWatch [5]) is added, a separate information agent can be used for each type of information source. In that case separate broker agents (like the ones used in RETSINA) are used to find the right information agent. This will also increase the scalability of the system.

The second function of the information agent is the data integration. Using a meta data base about the formats of the sources it is possible to merge the information from different sources. This can best be done through one central agent per type of information source. Because we only consider one type of sources it leads to one information agent for our case. If the system is expanded in future, a separate agent for different information sources (like financial databases, stock information, etc.) is foreseen. Each agent can use the integration methods best suited for its own type of information.

Profile agent. This agent performs the same type of role towards the users as the information agent performs towards the sources. It collects the profiles of the users and also registers the behaviour of the users through the queries. This could also be done by the user interface agents but this would be very inefficient, because they would all need to access all the other user profiles and all the queries. It seems logical to combine the managing and datamining functionalities in one agent as is done now and not spread that functionality over many user interface agents. The profile agent also can become a bottle neck if many users connect to the system. In that case one profile agent per department could be used. This would already give a first selection of user profiles and has the advantage that the agents are spread over the servers of the departments. Just like with the information agents a second type of general broker agent has to be introduced to find the right profile agent.

Interface agent. The last agent type to be considered is the user interface agent. This agent type was not implemented for the following reasons. In principle the user interface agent should translate the user profile into a data presentation format. However, for the moment this translation is the same for all users. For each source that contributes some articles to the answer there is a window on the screen that shows the answers. The answers are ordered on the basis of the user profile, which is also known to the profile agent. The user interface agent also filters the answers through the user profile, but this work can also be done by the profile agent. Because the functions that are performed and the rules for each function are the same for all users, this work can as well be done by one central agent.

Server or client. A last decision that should be made with respect to the agent architecture is where the agents should reside. For the source agents it seems logical that they reside on the server that contains or connects to the information source. The information agent is located at a central server (in this case of the library). The profile agent is located at the same server as the information agent, but could also be located at the department. Ideally, the interface agent resides on the PC's of the users. However, there are at least two problems. First users can not use the PC's of other users without using their agents as well. It happens regularly that users want to get information while in a meeting with some other persons. They should have a laptop with them and be able to connect at the other place in order to do this with the use of their own agent. Even making the connection is not trivial. Because of security reasons not every laptop can be connected to every network connection. Secondly, if the agents would reside on the PC's they can only perform pro-actively while the PC is powered (and most useful things only when it is connected to the network.). If the user interface agents would reside on the server they could perform this pro-active behaviour during the night.

Both arguments indicate that it would be better to have the user interface agents reside on the server. However, the overhead that each agent takes when they are build with full functionality and/or using standard agent frameworks are such that it is not possible to have several hundreds of agents working in parallel on the server. This lead to the conclusion that at least for the moment it is better to incorporate the functionality of the user interface agents into the profile agent. Because this agent can coordinate the work it does for all users internally it can spread the load of getting information updates over the entire night. Also it balances the workload that comes from the user interaction and thus optimizes the use of resources of the server.

Security. A point that was not considered during the design of the system were security regulations. At the moment all sources that are connected to the system can be connected without security problems. However, if departmental databases are also included this is no longer the case. The employees from the trading department (stocks, etc) are not allowed to see the information from the investment department that advises big clients on their investments. This can very easily cause cases of trading stock with foreknowledge, which is illegal. So, information from these two departments should be strictly separated. Having a central information agent makes it easy to implement this type of security regulations.

5.2 Implementation

When this project started one of the goals for the organization was to get experience with agents and see how they could benefit from the use of agents. The idea was therefore not to try to develop new theory or implementations for the agents themselves, but to make use of existing theories and tools as much as possible.

In order to define an architecture and find out whether this would suit the situation we made use of the agent modelling tool DESIRE [3]. In this tool the agent architecture of each agent can be defined independently and also their interactions can be precisely defined. The tool proved very helpful in defining the architecture of the overall system. However, the tool does not provide a translation of the agent model into running code on a Windows platform. One of the requirements of the project was that the user interface should be incorporated into a browser. It followed almost immediately that it would be best to have the code of the actual agents written in Java. Using Java it would be very easy to make an interface to a browser using a servlet.

Therefore we looked for standard agent libraries in Java in which the architectures of the agents that were defined in DESIRE could be implemented. We have looked into several agent libraries such as JAT0.3 [11], JATLITE [12], JAFMAS [13] and KAFKA [14]. Without going into the details of our comparison at this point it appeared that some of the libraries were not bugfree yet and others already required a certain type of agent architecture that did not fit with our agents. We finally chose JATLITE to implement the first (prototype) version of the system. Although the agents could be implemented easily in JATLITE there are also some disadvantages in the use of this package. The main disadvantage is that JATLITE supports full KQML communication between the agents. KQML stands for Knowledge Query and Manipulation Language and is a general purpose communication language for agents [8]. Due to the fact that it is an all purpose language it is very extensive and its expressive power far bigger than what is needed for this setting. JATLITE provides all types of handlers to process the KQML messages, keep track of the protocols, etc. This means practically that JATLITE is quite large (around 2 Mb). This would not be a problem when the communication between the agents would necessitate complex protocols and many types of messages. However, in our case the messages are very simple and only consist of two types. Therefore the overhead of the use of KQML is too large for the benefits of it in this setting. It also played a role not to have separate user agents running on the server, because there is no way the server could support a hundred agents running at the same time, while each agent takes around 2 Mb. Therefore, it was already decided that the agents would be implemented again from scratch if the project is continued beyond the stage of a prototype and showcase.

In the prototype system only the basic components of the agents have been implemented completely. That is, the user can make a user profile using the topics explicitly. Every day she will get all the news updates pertaining to these topics. The queries from the user are translated and send to four types of sources. The answers are shown per source. The project had to be implemented this way, because only in this way it was possible to show the users within a very short time what the potential uses of agents are in this setting. At the moment a start is made to incorporate more functions, especially the thesaurus function for the resources and the datamining function for the profile agent are now being developed.

6 Conclusion and future work

In this project we have tried to answer the question whether it is useful to use agents as part of an information management system within organizations. Specifically we have been looking at a situation where there are several autonomous information sources with (partially) unstructured data. This is a type of information organization that lies in between the strictly hierarchically ordered (distributed) database systems and the complete anarchy of the WWW. This is the most typical information organization form for most companies that use a form of intranet.

The main conclusion that can be drawn from the pilot project is that in this situation it is very useful to make a distinction between the organization of the information on one side and the user interests (and related interface) on the other side. There are clearly several tasks to be distinguished that each have their own goal. One task is to organize the information in an optimal way and make it uniformly accessible. The other task is to filter the information according to user interests and present the information in an optimal way. Each of these tasks can very well be performed by an agent (type).

The prototype implementation that is described in the previous section is used (occasionally) by several dozens of people. These users are mainly people working in staff functions. They are very happy with the uniform interface and the results they get on their queries. The real value of the system as well as its scalability can only be tested when the system is used throughout the whole bank. Before this can be tried out the implementation will have to be improved as sketched above.

Within the project we have also indicated the new possibilities that the use of agents can give us in this situation. Those are related to datamining aspects of both the user interests as well as the information sources. On the user side it will be possible to find out which people in an organization are interested in a certain topic, which can be helpful for the knowledge management of an organization. Also it will be possible to easily check whether it is profitable to maintain a certain source or whether it should be discontinued because of a lack of interest. Or (what happens more frequently) should only be available for certain departments within the organization. On the information side it will be possible to couple all types of information from different sources.

From this project we have also learned that there is a clear lack of agent methodologies. That is, methodologies that indicate how systems that involve agents should be designed and implemented. Although DESIRE was a good tool to aid in the design of the agent system itself it does not help in the problem analysis. How do we describe the situation in a way that it is clear which types of agents are needed and how the tasks of these agents are divided? Some of the practical considerations only became apparent during the implementation while they did influence the agent model.

The next steps within this research are already clear. First we will rewrite the implemented agents directly in Java in order to reduce the overhead for the communication, which is unnecessary. We will subsequently look into ways of implementing separate user interface agents for all users trying to circumvent the problems that were described above. Some work is already being done on implementing the datamining aspects described above. This is developed independent from the agents but in a way that it can be incorporated into the agents once they have been completely reimplemented.

In parallel to these implementation issues we will do some research into learning algorithms for the user profiles. It is not very clear which method should be used under the circumstances of this system. An important point here is that we do not assume that the user profile will be learned at one time and be static after that. We assume that the user profiles keep changing over time, because the interests of organizations and people change over time.

Finally, we will look into the integration of research done in cooperative information systems into the tasks of the information agent. The main difference with most work in this are is that we assume that the data is only partially structured, while most work in the area of cooperative information systems assumes that the information is structured (in databases).

Acknowledgements

At this place I would like to thank Edwin Eversdijk who performed all the practical work at the ABN AMRO bank, which formed the basis for this article and the anonymous reviewers whose comments greatly helped to improve the paper.

References

[1] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, D. Woelk: "Semantic Integration of Information in Open and Dynamic Environments". MCC Technical Report MCC-INSL-088-96, October, 1996

[2] D. Billsus and M. Pazzani. “Learning Collaborative Information Filters”. Proceedings of the International Conference on Machine Learning. Morgan Kaufmann Publishers. Madison, Wisc., 1998.

[3] F. Brazier, B. Dunin-Keplicz, N. Jennings, and J. Treur. DESIRE: modelling multi-agent systems in a compositional formal framework. International Journal of Cooperative Information Systems, vol 6, no. 1., 1997, pp. 67-94.

[4] M. Bright, A. Hurson and S. Pakzad. “A Taxonomy and Current Issues in Multidatabase Systems”, IEEE Computer, 25(3), 1992, pp.50-60.

[5] CBS MarketWatch. http://www.cbs.marketwatch.com/

[6] R. Duda and P. Hart. “Pattern classification and scene analysis”. New York: John Wiley & Sons, 1973.

[7] O. Duschka and M. Genesereth. “Infomaster – An Information Integration Tool”. In Proceedings of the International Workshop on Intelligent Information Integration, Freiburg, Germany, September 1997.

[8] T. Finin, Y. Labrou, and J. Mayfield. “KQML as an agent communication language”, in J. Bradshaw (Ed.), ``Software Agents'', MIT Press, Cambridge, 1997.

[9] M. Holsheimer and A. P. J. M. Siebes. “Data Mining: the Search for Knowledge in Databases”. Technical Report CS-R9406, CWI, Amsterdam, The Netherlands, 1994.

[10] H. Jamil and P. Johannesson. “Semantic interoperability - context, issues and research directions”, Proceedings of the second conference on Cooperative Information Systems, Toronto, Canada, 1994, pp. 180-191.

[11] JAT0.3. http://cdr.stanford.edu/ABE/documentation/overviwe.html

[12] JATLITE. http://java.stanford.edu/

[13] JAFMAS. http://ececs.uc.edu/~abaker/JAFMAS/

[14] KAFKA. http://www.fujitsu.co.jp/hypertext/free/kafka/

[15] L. Kerschberg. “Knowledge Rovers: Cooperative Intelligent Agent Support for Enterprise Information Architectures”. In P. Kandzia and M. Klusch (eds.), Cooperative Information Agents, (LNAI 1202), Springer-Verlag, 1997, pp.79-101.

[16] K. Lang. “NewsWeeder: Learning to filter news”. Proceedings of the Twelfth International Conference on Machine Learning. Lake Tahoe, CA, 1995.

[17] M. Papazoglou, S. Laufman and T. Sellis. “An organizational framework for cooperating intelligent information systems”, Journal of Intelligent and Cooperative Information Systems, vol 1, no.1, 1992, pp. 169-202.

[18] K.Sycara, D.Zeng. Coordination of Multiple Intelligent Software Agents. International Journal of Cooperative Information Systems, vol 5, num 2 & 3, 1996.

[19] E. Verharen and F. Dignum, “Cooperative Information Agents and Communication”, Cooperative Information Agents 1st Int. workshop, (Springer LNAI 1202), 1997, pp.195-209.

[20] A.Unruh, G. Martin, and B. Perry. “Getting Only What You Want: Data Mining and Event Detection Using InfoSleuth Agents”. Technical Report MCC-INSL-113-98, 1998.

[21] G. Wiederhold. “Foreword to Special Issue on the Intelligent Integration of Information”, Journal of Intelligent Information Systems, vol 6, 2/3, 1996, pp. 93-97.

[22] M. Wooldridge and N.R. Jennings. “Intelligent Agents: Theory and Practice”, The Knowledge Engineering Review, vol 10 (2), 1995, pp. 115-152.