From: AAAI Technical Report SS-99-03. Compilation copyright © 1999, AAAI (www.aaai.org). All rights reserved.

Adaptive Agent-based Systems for the Web:

An Application to the NECTAR Project

Bruno Errico
Etnoteam S.p.A
Via A. Bono Cairoli, 34
20127 Milan, Italy
berrico@etnoteam.it

Igor Jurisica
University of Toronto
Faculty of Information Studies
Toronto, Ontario M5S 3G6 Canada
juris@ai.utoronto.ca

Abstract

The paper presents an overall framework for devising adaptive Web systems characterised by a smart interface that exploits personalised anthropoid characters. Part of the framework is currently investigated for the NECTAR system, a smart interface for on-line shopping. We propose an agent architecture and highlight the roles of different agents that can carry out tasks required for a smart interface. We define knowledge structures and reasoning abilities that can be considered by such adaptive systems. In particular, we point out how adaptive Web applications could benefit from using case-based reasoning, data mining, and knowledge discovery techniques.

1. Introduction

Currently, there is an explosion of Web applications, spanning over different fields: electronic commerce, electronic publishing, health, education, entertainment, etc. Researchers and designers working on such applications often need to take into account new peculiarities and opportunities that should be carefully analysed.

On the one hand, Web systems must effectively interact with their users, supporting fast navigation across a site and easy retrieval of the sought items (Arocena et al., 1997). This implies, for instance, to avoid the problems of deep hierarchical structuring of Web pages and long itemisations of information, which could both lead to a user being lost and to skip important information.

On the other hand, automatically collecting a large quantity of information about users (e.g., using log files), offers the possibility to analyse and anticipate user behaviour in order to facilitate future interactions and optimise the application tasks (Choo et al., 1998).

In this work, we apply agent technology to solve these problems. We address challenging application environments, which are characterised as follows:

- complexity, coming from the need to include different and sophisticated processing abilities;
- dynamics and openness, since a system can rapidly evolve and operate in a way that can not be completely foreseeable during system design;
- friendliness and appeal, coming from the need to establish a pleasant and enjoyable interaction with users;
- diversity, stemming from the existence of different user types, needs and expectations.

We present a general framework for developing a personalised agent-based application over the Web. This work has been partly developed within an ongoing Esprit Project, NECTAR (Nice Electronic Commerce through smarT Agents for Retail). The main goal of the NECTAR project is to design and implement a framework for electronic commerce applications using agent technology. We expect to exploit the implemented architecture to get feedback on the feasibility and usefulness of the framework. Hence, our main concern is a flexible architecture that is applicable to Web systems with a need for a personalised agent-based interface.

Section 2 describes the role of agents within the system, with particular emphasis on the Personal Agents module, aimed at devising an adaptive interaction with respect to the current user. In Section 3, we define the architecture of the agents, by describing the operations for managing and supporting the agents and their internal structure. Section 4 introduces the main knowledge structures considered within the system. We emphasise the main reasoning abilities, namely case-based reasoning, data mining and knowledge discovery techniques. Section 5 describes the ongoing NECTAR project, which applies some of the ideas presented above to the on-line shopping domain. The last section presents a discussion and a comparison with other relevant literature and systems.

2. Agent Roles

Agent technology can play an important role in the design of advanced applications, such as adaptive systems that use an anthropoid agents interaction. This approach offers important features, as described below:

- abstraction and modularity, which helps to deal with a complex environment;
- autonomy and learning, which helps to deal with an open and dynamic environment;
- life-likeness, which helps to develop an engaging and appealing interface;
- distribution and delegation, which helps to effectively deal with multiple different users.

In our framework, we have defined several basic agents with different roles, as depicted in Figure 1. The main distinction is between Personal Agents and Task Agents.

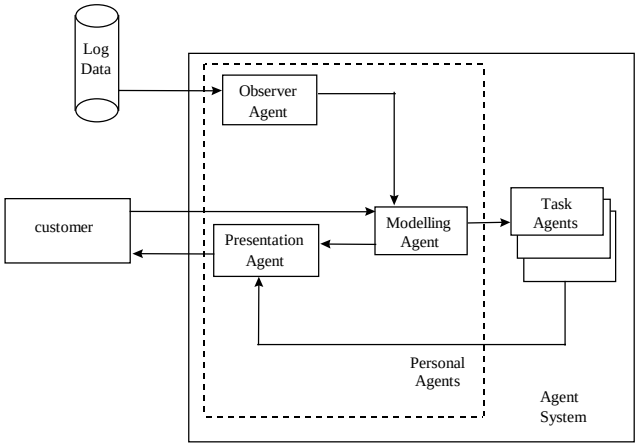

Figure 1. Agents Roles

The role of *Personal Agents* is to observe a user of the system, define her model by identifying and representing significant aspects of the user, and establish the modalities and features of the interaction with the user. More details about roles of individual personal agents is provided below:

1. *The Observer Agent* gathers information about the user and her behaviour. It observes the requests to the site server that are issued by the user and identifies the corresponding actions they represent. Identified actions are thus available for being processed by the Modelling Agent.
2. *The Modelling Agent* relies on information coming from the Observer Agent to determine a model of the current user behaviour and attitudes. As we shall see in the following sections, this task is carried out by exploiting suitable knowledge structures and techniques, e.g., stereotypes, contexts, and knowledge-discovery techniques. Further information that is stored by this agent can be derived from data or indications that are directly provided by the user, through interviews and forms. Thus, the Modelling Agent can provide information to other agents that solve specific tasks, or Task Agents.
3. *The Presentation Agent* communicates information to the user and tailors it to the current context, user's tastes and preferences. To do so it exploits its own knowledge, expressed in terms of techniques and policies that could be used when information should be presented. The selection of a specific presentation mode is based on the information about the current user that is acquired by the Modelling Agent. This includes user's attitudes, preferences and features. In practice, this agent can customise a piece of information to be presented by personalising the following features:
   a) graphical presentation, by selecting among a library of anthropoid characters and adding suitable images and animations, tailored to the individual user and presentation kind.
   b) content, by selecting the most appropriate information personalised for the individual user and the request that has been issued;
   c) structure, by suitably organising the structure of the page content, e.g., precedence among different items, hierarchical representation, and levels of highlighting;
   d) level of detail, i.e., once the nature and the structure of the information that need be provided has been determined, the amount of details could be varied according to the current user. For instance, a new user would need a greater amount of explanation; or a user with a limited bandwidth will be given, by default, a lower amount of graphical information.

Task Agents perform specific tasks in response to direct or indirect user activation or to activation by the system. They rely on information collected by the Modelling Agent. The task results are presented to the user through the Presentation Agent. Naturally, Task Agents should also possess additional, domain-specific knowledge and inference abilities in order to accomplish the associated tasks.

Based on the aims and the business logic of the applications we can define several types of Task Agents. Some general task Agents and their role in various domains is described below:

- *An Analyser Agent* is in charge of processing the data collected by the system concerning interactions with users and the interface. It provides significant statistical information and discovered knowledge, which is useful for management decisions. Furthermore, it can support the Modelling Agent by identifying classes of users and classes of domain items, thus improving the modelling abilities of the system.

- *A Helper Agent* represents an approach to on-line help. It gives technical support to a user by explaining tools and features of the system. It can become active either on demand (i.e., upon request of the customer), or proactively (the agent itself recognises that a customer has some difficulties).
- *A Search Agent* executes a search of domain items as specified by a user. Our Search Agents consider various levels of matching, including exact and partial.

3. The Agent Architecture

This section analyses the functionality provided by the proposed agent system. In addition, we give a description of the internal structure of an agent, thus providing a functional decomposition of agents along with the main operations that need to be performed and the data they operate on.

3.1. The Agent System

The overall agent system is divided into two modules:

1. *The Agent Manager* (AM), which controls other agents, by creating, activating, and destroying them. Thus, this agent supports monitoring of the agent system.
2. *The Agent Environment* (EA), which supports the execution of agent operations.

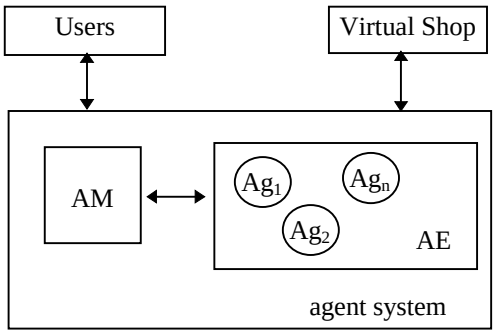

Figure 2. Agent System

The Agent Manager supervises the activity of the agents. In particular, the following tasks are performed:

- Create new Personal Agents for a user. This task corresponds to create an Observer, Modelling and Presentation agents for a new user and assign them the identity of the user.
- Retrieve the Personal Agents assigned to a user. This task associates the user with the Observer, Modelling and Presentation agents with the same identity as the user.
- Activate an agent. This task corresponds to directly requesting the intervention of an agent for executing a specific task.
- List all agents. This task can be used for testing or monitoring purposes and shows all agents and their identity.
- List all active agents. This task can be used for testing or monitoring purposes and shows all active agents, their identity and their current task.
- List the properties of a specified agent. This task can be used for testing or monitoring purposes and shows the properties of an agent, i.e., the identity, the state (active or not), and the task that it is performing (when active).

The Agent Environment should provide all the necessary tools for the execution of agents. In particular, the major tasks that are supported are as follows:

- Multithread management, which ensures that several agents may be executed autonomously and simultaneously.
- Distribution management, which makes the agent architecture scalable by allowing addition of more servers for executing agent. In addition, the load of any potential server is balanced by choosing where to execute a new active agent.
- Message delivery supports communication among agents, with the environment, and with the users.
- Persistence management stores information needed by the agent in different situations. In addition, the manager also supports crash recovery.
- Security management enforces the rules governing agent behaviour, e.g., protects against unwanted access.

3.2. The Agent Internal Structure

We have defined a general internal structure for an agent, which we apply to all of the agents considered in the proposed framework. The structure of an agent that we have chosen comes from an anthropomorphic metaphor, which has been already exploited by other authors, and recalls the one adopted in (Brancaleoni, Cesta, and D'Aloisi 1997). As shown in Figure 3, an agent is viewed as decomposed into two functional modules:

1. *The Brain* controls the agent operations and determines tasks and actions;
2. *The Body* executes tasks and actions, and communicates with the environment.

The brain is devoted to the effective management of the agent abilities in order to solve the current problem. Thus, it should include advanced functionality, such as:

- reasoning abilities to determine the goal that is to be achieved,

- planning abilities to map the available tasks on the current goal, and
- decision-making abilities to monitor and modify the action execution according to the actual situation.

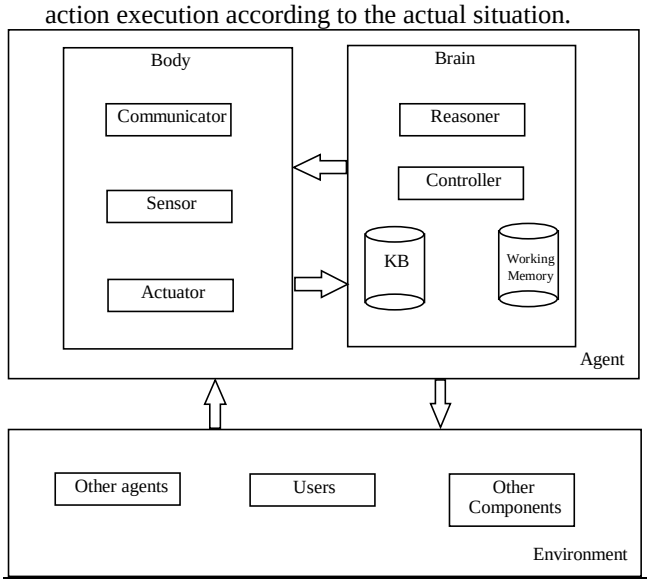

Figure 3. Agent Architecture

Three main components are considered within the brain:

1. *A Controller*, which determines when the agent should act and what should it do as specified by the following operations:
   a) selecting the current goal;
   b) selecting the action that should be executed next;
   c) monitoring the execution of the actions;
   d) activating the Reasoner to decompose complex actions (goals) into executable ones managing incoming messages.
2. *A Reasoner*, whose task is to determine a suitable course of action, by performing the following operations:
   a) assessing the current situation;
   b) determining the plan that achieves the current goal.
3. *A Memory* that contains necessary knowledge for solving the task, and information about the current state. This is distinguished into:
   a) a knowledge base, containing rules, plan libraries, representation of relevant knowledge;
   b) a working memory, containing the current goal and a description of the current situation.

The body is in charge of interacting with the system. It perceives the environment in order to collect needed information and to affect it by performing the scheduled actions. Thus, the body comprises:

1. *A Communicator*, which ensures communications with other agents or other components in general. This is done by means of a message manager that supplies the chosen communication protocol, e.g. KQML;
2. *A Sensor*, which contains a set of available sensing actions that allow an agent to test the environment;
3. *An Actuator* that is in charge to execute predefined actions and tasks that are selected by the controller.

4. Representational Issues

Adaptive systems need to define suitable data structures in order to handle needed information in an effective and efficient way.

Domain knowledge can be organised into classes representing relationships and associations among individuals. For instance, classes can be organised into hierarchies representing *is-a* and *part-of* relationships. Likewise, relevant features about users can be structured around suitable taxonomies that represent classes of users, called stereotypes. In order to successfully associate appropriate suggestions, we need to keep track of user actions and attitudes. States can be used to represent the context, e.g., the current stage, of the interaction taking place between the system and user. Finally, suitable data structures are introduced to represent analyses of the overall behaviour and utilisation of the system.

The discussion that follows will provide general definitions concerning representational issues for these kinds of knowledge. Most definitions will intentionally range over a formal and abstract level. This allows us to both state clear and well-founded definitions, and to have a general framework that can be exploited in different domains.

4.1. Stereotypes

Stereotypes are used to relate a user to a set of assumptions that capture significant aspects of the user model, based on a given evidence. This idea has been widely exploited in the user modelling and user adapted interaction communities and dates back to the work of Rich (1979). Evidence can be expressed by means of triggering and deactivating conditions that are used for ascribing a stereotype to a user or retracting it. Thus at any moment a user is assigned to a set of different stereotypes that contribute to define her user model. Stereotypes can be organised into hierarchies and multiple inheritance can be supported. Features can be considered as a set of facet-value pairs that hold for all users ascribed to the stereotype. Finally, a context can be expressed as a condition restricting the set of scenarios which the stereotype can be

applied to. The data structure of a stereotype can be described as follows:

- *Triggering condition* that defines a condition in terms of actions performed by the user and features acquired from the user that make the stereotype potentially applicable;
- *Deactivating condition* that defines a condition in terms of actions performed by the user and features acquired from the user that make the stereotype no longer applicable;
- *Features* that define a set of facet-value attributes that could be ascribed to the user when the stereotype is applicable;
- *Context* that specifies those scenarios where the stereotype could actually be exploited once the stereotype is triggered.

Stereotypes thus enable defining different, possibly overlapping, classes across the universe of users, along different dimensions.

We extend this generic approach to stereotypes with case-based reasoning (Kolodner, 1993). Case-based reasoning paradigm is founded on solving new problems by remembering (representing), retrieving and possibly adapting experience, represented as cases. Informally, a case comprises an input (the problem), an output (the solution) and feedback (an evaluation of the solution). However, the representation of problems, solutions and feedback varies from domain to domain.

For our purposes, cases represent experience of specific users. Using knowledge-discovery techniques described later, specific cases are grouped into clusters of customers with related needs and tastes. We use hierarchical clustering, which thus creates a generalization hierarchy of known user stereotypes. New users can then be assigned to certain stereotypes, which in turn are defined in terms of specific cases. Thus, in this representation stereotypes can be viewed as a composite of several cases. Cases are described by set of features -- attribute-value pairs, conditions -- triggering and deactivating, and scenarios. We use a prototype of a case-based reasoning system TA3 (Jurisica and Glasgow, 1997). The system is *flexible* -- it can be used in diverse domains and can support various tasks, and *scalable* -- it works with large and complex case bases. An essential part of TA3 is an incremental, variable-context, similarity-based, retrieval algorithm (Jurisica and Glasgow, 1998). Defining context explicitly supports flexibility - controlling what is considered similar and why. Incremental implementation of the retrieval algorithm helps to achieve scalability. TA3 also includes a knowledge-discovery component, which is used for: *(1) TA3 optimization* -- locating descriptors relevant for a given context and task, and organizing case base into context-based clusters; *(2) case base and domain knowledge evolution* -- adding descriptors to assist case discrimination during prediction and classification, removing redundant cases and descriptors, creating hierarchies of descriptors and their values, finding associations; *(3) evidence-based reasoning* -- analyzing created clusters, hierarchies and associations to identify underlying principles in the domain.

Our approach is similar to user modeling systems that use stereotypes, such as GRUNDY (Rich, 1983) or GUMS (Finin, 1989) with the following differences:

- cases are not static structures and can store more complex information than stereotypes;
- support for partial matching, constraint satisfaction, uncertainty and similarity, which give us higher flexibility than activation by triggers;
- support for incremental model matching that diminishes the problem of incomplete information at certain stages of reasoning;
- flexible context-based model of relevance assessment, which reduces the problem of choosing between stereotypes that are at the same level of generality but which are unrelated;
- cases support multiple matches that diminish the problem of typical and particular user characteristics that stereotype-based models suffer from;
- case hierarchies help to distinguish between explicitly and implicitly acquired knowledge, which can be used for improved inheritance and reuse of models from one user to the other;
- consistency of the defined and acquired models is maintained by using similarity-based hierarchical organization of cases.

4.2. Actions and Attitudes

In this section we consider the problem of how to summarise the actions performed by users and express some of their relevant attitudes, e.g., preferences. To this end, we will introduce the concepts of action and attitude functions, which quantitatively express actions that are performed more frequently by users and the importance they (are assumed to) give to some object of the application domain, with respect to some of their attitudes. In general, various attitudes can be taken into account, e.g., the knowledge, the familiarity or the preference with respect to a given domain object. Thus, we can express that a user has visited a certain page several times and that she shows some preference towards a given domain object.

Let us consider a generic problem domain, where $\{A_1,\ldots,A_m\}$ is a predefined set of significant attributes, each of which takes on values from an underlying set $dom(A_k) = \{v_{k,1},\ldots,v_{k,n}\}$. We consider a set of action functions, $\{Act_{i,1},\ldots,Act_{i,n}\}$, where any action function $Act_{i,k}(v):dom(A_k)\rightarrow[0..n]$ maps, for each user U_i and an attribute A_k,, a value of the attribute domain in an integer

number expressing the number of times the action is performed.

Moreover, we consider a set of attitude functions, $\{Att_{i,1},\ldots, Att_{i,p}\}$, where any attitude function $Att_{i,k}(v):dom(A_k)\rightarrow[-m..M]$ maps, for each user U_i and for an attribute A_k, a value of the attribute domain in a real number expressing an attitude value. The two constants, m and M, represent the lowest and highest storable values. A value 0 means that there is no evidence with respect to that attribute value; a negative value shows a negative attitude; and a positive value shows a positive attitude.

The attitude values can be assigned based on users' behaviour according to some predefined assumption and heuristic.

In order to avoid strong influence of actions that happened a long time ago on the current values stored for attitudes, the attitude values can be periodically reinitialised or multiplied by a scaling factor. For instance, on entering a system for the tenth time, the former attitude values can be multiplied by a scaling factor of 0.1 or else any time a user exits the system the attitude values are scaled by a factor 0.9.

4.3. States

States can be considered as variables, with a rather coarse granularity, that capture the dynamic behaviour of a user. It is important to associate a user with a state that defines the applicable context, in terms of an underlying scenario and a certain goal that is attempted to be achieved. The state detected by the system can be used for determining how to interpret the behaviour of the user, which inferences can be applied (e.g., the context of a stereotype), and which actions of the system are most appropriate.

The evolution of states can be expressed by means of suitable rules for computing the next state that is reached from a certain state, once a specific action is performed.

4.4. Discovered Knowledge

Knowledge discovery is defined as "*a nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data*" (Fayyad, 1996). To cope with the complexities of real-world domains and to take advantage of domain knowledge the process is usually interactive and iterative (See Chen, Han, and Yu (1996) for a review on data mining.) In our system, knowledge-discovery techniques serve several purposes:

1. Discovered regularities, structures and patterns can be exploited by the site administrator to take strategic decisions.
2. The analysis on data of real users is crucial to update and iteratively improve the modeling and inference mechanisms that are initially built in into the system by means of elicited knowledge from marketing experts and heuristic hypotheses and assumptions.
3. Existing stereotypes can be analyzed to find their generalizations and possible higher-level clusters.
4. Grouping user profiles, in terms of related goals and actions taken can lead to finding associations that can be exploited during task-oriented interaction with the system.

Knowledge-discovery techniques can be used to find:

- *Association rules*, which we express as:
  $X_i=v_i,\ldots,X_j=v_j \rightarrow X_k=v_k,\ldots,X_l=v_l$,
  where $\{ X_i,\ldots, X_j\} \cap \{ X_k,\ldots, X_l \}=\varnothing$;
  X_t belongs to:
  $\{A_1,\ldots,A_m\} \cup \{Act_1(A_p),\ldots,Act_n(A_q)\}$
  $\cup \{Att_1(A_r),\ldots, Att_p(A_s)\}$;
  and, accordingly, v_s is a value belonging to the domain of an attribute, the application of an action function, or an attitude function to such a value. An association rule means that when the attributes, actions or attitudes on the left-hand side of the association rule assume the defined values, then also the attributes, actions or attitudes on the right-hand side of the association rule assume the corresponding values. For instance, it can be detected that users that are between 30 and 40, buy milk and visit the biscuit department usually also buy honey and cereals. A *confidence factor* c of an association rule is defined as the percentage of transactions that contain both the left-hand side and the right-hand side with respect to all those containing the left-hand side. The *support* s of a rule is defined as the percentage of transactions containing both sides with respect to the overall number of transactions.
- *Sequential patterns*. A sequential pattern is defined as a sequence of actions that gets executed during a transaction: $Act_i(A_i),\ldots,Act_m(A_m)$ (unlike association rules, the relative position is significant here). These patterns can play a particularly relevant role for anticipating the actions of users.
- *Path traversal patterns*. These can be seen as a special case of sequential patterns where the actions are visits of web pages. They are particularly relevant for analysing and optimising site organisations with respect to the accesses of the users.
- *Clusters*. Clustering helps to identify classes by trying to maximise intra-cluster similarity and minimising inter-cluster similarity. Clusters are discovered based on some emerging common features. Features that are taken into account are attributes, association rules, sequential patterns. Particularly interesting are clusters that can be generated by special association rules where the left-hand side contains only actions performed on

some attributes. In this case, new clusters can be associated with new stereotypes that are dynamically generated based on the attributes of the left-hand side of the association rule. For instance, customers of virtual shop who are between 30 and 40, buy a certain amount of milk and have visited the biscuit department can be automatically gathered into a new stereotype. Likewise a new cluster of products can be discovered gathering together honey and cereals in a cluster of products, even if they were not associated to each other before. Note that these inferences can be drawn automatically by an agent, which can thus refine and update the domain knowledge by adding new clusters and stereotypes, e.g., a stereotype whose triggering conditions are between 30 and 40, buying milk and visiting the biscuit department and has as feature a high preference for the new cluster of products containing honey and cereals.

5. The NECTAR Project

NECTAR (Nice Electronic Commerce through smarT Agents for Retail) is a European Commission funded project in the framework of ESPRIT IV. The project started on March 1st, 1998 and has duration of 18 months. The NECTAR Consortium comprises three groups of partners:

- End-users: two large retail companies in Italy and Spain;
- Technology providers: three state-of-the-art companies in the respective market sectors in France, Italy and Spain. The Italian partner, Etnoteam, is also the project co-ordinator;
- A Business analyst: a BPR company in Italy.

Main project results aim at defining a general model of a virtual shop for retail with Intelligent Agents. Two instances of the virtual shop, one for each of the retailer partners, are currently being developed. The system is composed of several software modules:

- a *back-end module* to interface the existing Information System of the Retailer with the Virtual Shop module;
- a *virtual shop*, which maintains a dynamic product catalogue and drives the commercial offer;
- a *front-end module* based on Intelligent Agents: they will support a personalised interaction between the customer and the system, taking into account preferences, habits and trends.

The NECTAR agent system is under development in Java; KQML is used for communication and co-operation among agents. Preliminary results are foreseen at spring 1999, while the final system will be available at the project end (August 1999).

A set of agents resulted from the specific requirements of the retail market. In addition to the general agents Personal and Task Agents considered in this framework and described in Section 2, the following agents have been taken into account:

- *The Watcher Agent* monitors offers and promotions and notifies customers about them. Notification can take place either by e-mail (off-line) or on-line. In both cases, notification occurs only if the customer is interested in some way, according to the customer model built by the Modelling agent.
- *The Reminder Agent* reminds a customer about special events and significant dates (e.g., birthdays, anniversaries, religious holidays) that are relevant. It proposes possible gifts for these special occasions. This could be done either by e-mail or on-line.
- *The Chef Agent* suggests menus and recipes for special occasions. According to the provided suggestion, this agent also proposes the best list of products to prepare these recipes based on the customer's preferences.
- *The Shopper Agent* helps the customer to manage the personal shopping list. It notifies the customer about observations and advises her on the current shopping basket. For example, the system may notice that the customer has forgotten an item, because this is considered strongly related to some other item that has been selected by the customer.
- *The Pro-Merchant Agent* provides sorting criteria to be applied to lists of products. Criteria are based on weights and rules designed to improve the profitability of the shop.
- *The Pro-Customer Agent* provides sorting criteria to be applied to lists of products. Criteria are designed to improve the shopping advantage from the customer side and to take into account customer preferences and tastes.

As an example, the following ones are two possible predefined stereotypes for customers of the NECTAR project: *economical* -- related to product cost, and *famous brands* -- related to product features:

Economical stereotype:

- triggering condition: the customer ticks the "care for price" box in the registration questionnaire;
- deactivating condition: the customer rejects at least ten offers of products of the product cluster *economical*.

Famous brands stereotype:

- triggering condition: the customer buys at least three products of the cluster *famous-brands;*
- deactivating condition: the customer rejects at least two offers of products of the product cluster famous-brands.

Additional project's contribution will be the reorganisation of several business processes of the Retailers (delivery, logistics, marketing, finance, etc.), according to the new commercial channel. Again, a general model of the Retailers new Business Processes will be established and exploited for the implementation of the two specific systems.

A great challenge of the NECTAR project is the application of the technology of Intelligent Agents to the Retail market. A complete system for advanced electronic commerce in the retail market will be developed and the relevant business processes will be analysed and reorganised according to the needs of the new commercial channel. This involves an innovative integration of existing software technologies and the exploitation of new business processes.

From the customer perspective, the innovative aspect is the chance to have a convenient, interesting and fun experience in the electronic retail.

6. Conclusions

The possibility of providing users with a more effective interface is one of the future challenges and has been investigated in a number of project.

The persona project (Ball et al. 1997) explores a prototype system for a social user interface, that interacts in natural language by means of a lifelike character. The assumption of this project is that future systems should go beyond current graphical user interface where users accomplish tasks by a selection from a predefined set of alternatives. Thus, it allows for a description of tasks, which can be in part performed autonomously by the system. Though the emphasis is mainly on conversational issues of the interface, this project is in many respects related to the motivations of our work for developing a (partly) autonomous interface based on lifelike characters.

The idea of detecting user preferences for determining system behaviour has been exploited in several systems, particularly in the area of user modelling. At present, there is a considerable number of projects in this area, with a great focus on adaptive Web systems.

In the AVANTI project the possibility of providing individualised information is based on three models for capturing relevant features of users, system usage, and domain. This project also strives to satisfy the need of users, e.g., with particular I/O requirements. See (Finik, Kobsa and Schreck, 1997) for a description of the project.

WebMate (Chen and Sycara 1998) is a personal agent that improves a user ability of retrieving document through the Web, by monitoring a user searching and browsing activity and exploiting user evaluation on positive results retrieved by the agents. Preferences are managed by the technique of multiple TF-IDF vectors, which represent documents as vectors of words and weights. The searching task is further refined by means of techniques based on the generation of keywords and on relevance feedback.

For what concerns the application of intelligent agent to e-commerce, many works have been focusing on devising agents that could perform customer tasks, like wondering on the net, selling and buying goods (see Guttman, Moukas, and Maes (1998) for a review).

Surprisingly, there seems to be fewer ongoing projects aimed at devising advanced user interfaces for e-commerce providers. In this light, the Nectar project assumes a crucial role, having the aim to provide a reusable framework to be applied to several applications of the retail domain and, possibly, to generalise it to other related domains. Nwana (1998) summarises views of some participants in a panel session on agent-mediated e-commerce on the research state-of-the-art and future challenges. Interestingly, one of the panellist (Tuomas Sandholm) singles out the problem of interest generation as one of the stages of commerce and e-commerce, where *"a user's agent keeps a profile of the user, and selectively chooses which advertisements the user should read"* (p. 192). This idea strongly resembles the tasks performed by the agents introduced in our framework, i.e., the Personal Agents and the Watcher Agent.

Among commercially available products, Brightware (1998) offers a packaged software application aimed at developing virtual shops with automated customer interaction on the Net. Brightware is designed as an application for interacting with customers, organising and accessing relevant content and guiding customers to execute the sales or service transactions that best meet their needs.

However, compared with the agents considered in our framework, a reduced set of functionality is covered by the agent system. This amounts to two agents based on fuzzy matching and machine-learning technology: an Answer and an Advice Agent. Note that further extensions are under development.

Similar to our ideas, Tecinno GmbH uses case-based reasoning to support the user in finding desired information or products in electronic catalogues (Breen and Wilke, 1998). The main advantage over existing technologies is its measure of similarity. In comparison, our approach goes further since we not only measure similarity among products but also among user preferences and stereotypes.

7. Acknowledgements

We thank Vittorio Patera for valuable comments, revisions and suggestions on some ideas presented in this

paper. We also thank Graziano Ferrari for helping us describe the NECTAR project.

8. References

Arocena, G.; Mendelzon, A. O.; and Mihaila, G. 1997. Applications of a Web Query Language. *Computer Networks and ISDN Systems* 29:1305-1316.

Ball, G.; Ling, D.; Kurlander, D.; Miller, J.; Pugh, D.; Skelly, T.; Stankosky, A.; Thiel, D.; Van Dantzich, M.; and Wax, T. 1997. Lifelike Computer Characters: The Persona Project at Microsoft Research. *Software Agents*. Bradshaw, J. ed. MIT Press.

Brancaleoni, R.; Cesta, A.; and D'Aloisi, D. 1997. MASMA:. A Multi Agent System for Scheduling Meetings. Proc. of the Third International Conference on Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM 97).

Breen, S. and Wilke, W. 1998. Fuzzy searching of product Catalogues. Technical White Paper, Tecinno GmbH.

Brightware. 1998. URL: http://www.brightware.com/ products/index.html

Chen, M.; Han, J.; and Yu, P. S. 1996. Data Mining: An Overview from Database Perspective. IEEE Transactions on Knowledge and Data Engineering 8(6):866-883.

Chen, L. and Sycara, K. 1998. WebMate: A Personal Agent for Browsing and Searching. Proceedings of the Second International Conference on Autonomous Agents. Sycara K. and Wooldridge M. eds. 132-139.

Chin, D. N. 1989. Acquiring user models. *Artificial Intelligence Review* 7(3-4):185-197.

Choo, C. W.; Derlor, B.; Turnbull, D. 1998. A behavioral model of information seeking on the web: Preliminary results of a study how managers and IT specialists use the web. *Proc. of the 61st Meeting of the American Society of Information Science*, 290-302.

Fayyad, U. M.; Piatetsky-Shapiro, G.; and Smyth P. 1996. From data mining to knowledge discovery in databases. *AI Magazine* 17(3): 37-54.

Finik, J.; Kobsa, A.; and Schreck, J. 1997. Personalized Hypermedia Information Provision through Adaptive and Adaptable System Features: User Modeling, Privacy and Security Issues. 4th Int'l. Conf. in Services and Networks.

Finin, T. W. 1989. GUMS - A General User Modeling Shell. *User Models in Dialog Systems*. A. Kobsa and W. Wahlster eds. Springer-Verlag. 411-430.

Guttman, R. H.; Moukas, A. G.; and Maes, P. 1998. Agent-Mediated Electronic Commerce: A Survey. *Knowledge Engineering Review* 6.

Jurisica, I. and Glasgow, J. 1997. Improving performance of case-based classification using context-based relevance. *International Journal of Artificial Intelligence, Special Issue of ICTAI'96 Best Papers* 6(4):511-536.

Jurisica, I. and Glasgow, J. 1998. An efficient approach to iterative browsing and retrieval for case-based reasoning. Angel Pasqual del Pobil, Jose Mira and Moonis Ali eds., *Lecture Notes in Computer Science (IEA/AIE'98*), 535-546. Springer-Verlag,

Kolodner, J. 1993. *Case-Based Reasoning*. Morgan Kaufmann, San Mateo, CA.

NECTAR. 1998. EP26992 Project Programme. Issue 1.01, 28/01/98.

Nwana, H. S. 1998. Agent-Mediated Electronic Commerce Issues, Challenges and some Viewpoints. Proceedings of the Second International Conference on Autonomous Agents. Sycara K. and Wooldridge M. eds. 189-196.

Rich, E. 1979. User Modelling via Stereotypes. *Cognitive Science* 3:329-354.

Rich, E. 1983. Users as Individuals: Individualising User Models. *International Journal of Man-Machine Studies* 18:199-214.