Agent Technologies for the Development of Adaptive Web Stores

Agent technologies for the development of adaptive Web stores*

L. Ardissono, A. Goy, G. Petrone, M. Segnan,
L. Console, L. Lesmo, C. Simone and P. Torasso

Dipartimento di Informatica, Università di Torino
Cso Svizzera 185; 10149 Torino; Italy
{liliana, goy, giovanna, marino, lconsole, lesmo, simone, torasso}@di.unito.it

Abstract. With the increasing popularity of Web shopping, the personalization of the front-end of on-line stores has become a critical issue: these systems are accessed by customers with different backgrounds, expertise and preferences; therefore, their usability can only be improved by personalizing their interfaces to the needs of each specific user.
In this paper, we describe the architecture of SETA, a prototype toolkit for the creation of adaptive Web stores, where user modeling and flexible hypermedia techniques are exploited to personalize the interaction with the user, dynamically generating the catalog pages on the basis of the user's features. The SETA architecture is composed by different specialized agents, which cooperate to the management of the interaction with the user; we will discuss how the application of agent-based techniques has been essential to the design and development of this system.

1 Introduction

The research on electronic commerce has opened several scenarios where agent-based technologies can be exploited in the design of open systems having auction, negotiation and brokering capabilities. For instance, in the business-to-business area there is a lot of on-going work on the development of complex architectures for enabling an automatic management of the supply chain [16, 33, 37, 45]. Moreover, in the business-to-customer area, agent-based technologies are exploited in the development of complex systems where agents search for products and services on behalf of a user, compare the solutions offered by different providers, and so forth [19, 22].

In addition to the above issues, agent-based technologies can be successfully applied to the enhancement of other features of electronic commerce systems, among which the adaptability of the interfaces to the users' needs: the popularity of Web shopping is increasing and very different types of customers purchase goods by accessing on-line catalogs on their own, without relying on intermediaries. Noticeably, this trend does not only concern traditional domains like

* This paper appeared on *Lecture Notes in Artificial Intelligence, Springer Verlag (http://www.springer.de).*

air-flight booking, or the market of movies and music, but also other domains, like the electronic power trade: in these domains, the user needs information about the alternative services offered by the providers, the advantages and disadvantages of the various options, their cost, in order to compare them and identify those satisfying her needs and constraints at best [44]. To comply with these requirements, several commercial tools for the creation of on-line stores, like Microsoft Merchant, IBM's Net Commerce and ATG's Dynamo, have been extended to offer more or less sophisticated personalization functionalities for tailoring the suggestion and the presentation of goods to the specific user. One major goal of these systems is to improve the interaction with the customer, in order to establish a long-term relationship with her, in contrast to the typical one-shot type of interaction supported by the first electronic commerce systems.

The customization of on-line stores is a complex activity and requires very different types of expertise, such as knowledge about users and products, and techniques to personalize the layout and content of the catalog pages; therefore, complex software architectures are needed. The exploitation of agent-based technologies can help to manage this complexity in an effective way. In particular, multiagent systems can be designed, where several agents offer specialized services, interacting with each other to produce the overall, complex service to the user (e.g., see [3, 39]). In principle, the agents devoted to specific services can exploit very different technologies, so that heterogeneous multiagent architectures are designed.

The work described in the present paper concerns the development of adaptive Web stores, capable to assist the customer during the selection of goods, varying the interaction style and the suggestions of the system on the basis of the user's preferences and needs. In the last three years, we have developed a prototype toolkit for the creation of Web stores supporting personalized interactions with users [8]. We have focused on the design and development of the front-end of a Web store, concentrating on the management of a flexible interface, and we have pursued two main goals:

1. The definition of the requirements of an adaptive Web store, concerning the interaction with the customer and the management of the system's resources, and the design of a system architecture meeting such requirements. To this extent, we have designed the system architecture by identifying a set of main roles to be filled and by associating a specialized agent to each role. The first testbed of the architecture is a prototype system presenting telecommunication products [7].
2. The design of a toolkit for the development of new adaptive Web stores. In order to reach this goal, we have revised the system architecture and developed a second prototype, where knowledge representation techniques and agent-based technologies are exploited to improve the configurability of the system and its scalability [5, 8].

Our current system, SETA, includes a customizable store shell and some configuration tools needed to help the store designer to set up a new Web store [5,

6, 8]. An on-line demo of a Web store created using SETA is available at the following URL: http://www.di.unito.it/~seta. This store presents telecommunication products, like phones, switchboards, and so forth, and will be used in the rest of the paper as a concrete example to describe the functionalities of SETA.

The paper is organized as follows: section 2 discusses the issue of personalizing interactions in Web stores. Section 3 introduces the main ideas on which our approach is based, while section 4 describes the SETA architecture, outlining the main roles identified in our Web store architecture and the agents filling them. Section 5 discusses the way how agent-based technologies have been useful in the design and development of SETA and section 6 provides some technical details about the system. At last, section 7 concludes the paper.

2 Personalization issues

With the expansion of the Internet, not only business users, but also a huge number of home users are browsing the Web and accessing on-line stores. As discussed in [12], users differ with respect to many characteristics, such as their status, expertise, preferences and the reason for connecting to the Web store, which should be taken into account to enhance the usability of systems. Therefore, while the initial research on electronic sales mainly focused on back-end activities like the provision of efficient order processing and secure payment transactions, other aspects, concerning the personalization of the interaction with the customer, are now becoming important. For instance:

- Since an on-line store is accessed by heterogeneous users, it should satisfy different needs and preferences in the selection of goods; this ability requires filtering capabilities, to identify the items most suited to the specific customer (e.g., see [35]).
- The advertisements included in the catalog pages should be targeted to the customer base and, in particular, selected on the basis of the interests and life style of the specific user accessing the system (e.g., see [2]).
- Furthermore, since Web stores are hypermedia systems, they should meet the users' needs in what concerns the interaction style [28].
- Finally, in addition to the customers' interactional needs, also their technological constraints should be taken into account to facilitate their access to the stores. For instance, they should be allowed to select alternative media for the delivery of the information about goods, depending on the efficiency of their connections to the Web (e.g., see [21, 25]).

The customization of a Web store involves many tasks: on the one hand, a detailed description of product features is essential to support a flexible selection of items suiting the customers' needs; on the other, the features characterizing the customers to which the store products are directed have to be identified, and an effective personalization can be reached if the peculiarities of the customer accessing the store are learned by observing her behavior.

By personalizing the interaction with the customer we mean that the relevant products should be selected for presentation and their description should be tailored to the user's expertise and interests, so that she can evaluate items more accurately than just looking at a raw product catalog. This aspect influences the selection of the information to be presented and its linguistic (or pictorial) form [14]; for instance, descriptions might use more or less technical terms and include images to improve their comprehension; the product descriptions should be focused on the features relevant to the users. Customers could even be helped by virtual sales assistants, who guide them in their browsing and product selection activity (e.g., see [24, 32]).

In some works, very general techniques, like those exploited in the information filtering research (e.g., collaborative filtering, or TF/IDF - term frequency/inverted document frequency) have been used to select interesting items in environments where heterogeneous information sources are exploited, or little information is available about the user's needs; for instance, see [9, 13, 35].

While those techniques are suited to deal with large-scale applications, such as information retrieval on the Web, other works show that more specific techniques can be applied to personalize the interaction with the user in applications where the domain-specific information is at least partially structured. For example, [34] exploit the typical structure of movie databases in a VOD recommender system to analyze the user's selections and evaluations of the items. They use the collected information to learn which product attributes affect the user's choices, so that user modeling techniques [27] can be applied to customize the suggestion of other items. However, the lack of knowledge about the meaning of the attributes and the relations among them does not support the adoption of effective strategies for tailoring the description of products to the individual user, taking into account aspects such as her interests and domain expertise.

Other researchers have adopted sophisticated approaches based on the application of natural language generation techniques for the dynamic production of personalized descriptions in the catalog pages. For instance, [17] exploit the structure of the records describing the items of a museum to analyze their content and dynamically generate summaries of the descriptions; the summaries may focus on different concepts, depending on the user's interests. Although these approaches support highly personalized interactions, they are not easily applicable in system shells, which have to be instantiated on different domains: in fact, these approaches require the definition of very detailed domain ontologies, which impose a strong overhead at configuration time.

3 Our approach

The description of the related work in the previous section supports the idea that a deep representation of the knowledge about products and customers is the basis for effectively personalizing the interaction with the customer. However, to face the trade-off between offering highly adaptive systems and constraining the effort in setting up these stores on the various sales domains, we have defined a

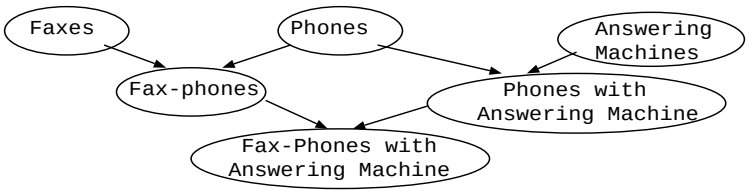

Fig. 1. Portion of the Product Taxonomy.

relatively compact representation of the domain-specific knowledge, concerning users and products: this representation supports the exploitation of powerful personalization strategies, without requiring that the store designer characterizes the semantics of each domain concept in too much detail.

3.1 Representation of the knowledge about products.

As far as product classes are concerned, we superimpose on the products database (which stores the information about the individual items available in the store) a conceptual representation of the domain: the Product Taxonomy contains the definition of all the products of the Web store and of their relations. In this representation, the relations among products are explicitly declared: for instance, similar products are grouped together; moreover, several basic products can be related to the products which offer the same functionalities in a single, integrated solution. This representation supports the generation of a rational sequence of catalog pages, making the navigation of the catalog easier; in particular, the taxonomy graph represents the structure underlying the hypertext forming the Web catalog. Figure 1 shows a portion of the product taxonomy of our prototype.

The Product Taxonomy specifies the features characterizing the items of each product class. The product features are represented in a declarative way, as structured entities with the following attributes:

- *Type*: the features concern different aspects of the product description. Since a detailed specification of the semantics of each feature would require an extremely detailed definition of the domain knowledge, we have characterized features in an abstract way, by defining the following feature types:
  - *Functionalities* are basic facilities representing the purposes for which a product has been designed; e.g., phones support vocal communication, while faxes are designed to transmit documents.
  - *Technical features* concern technical details (e.g., the resolution of a fax).
  - *Functional features* include minor facilities offered by the product (e.g., the agendas offered by phones).
  - *Aesthetic features* concern aesthetic aspects, such as color and size.
  - *Generic features* include information not belonging to the previous types, such as the price.
- *Importance*: this slot specifies to which degree the feature represents a mandatory piece of information in the description of a product.

```
Record 1:
  Name: Facile;
  Code: I00025;
  Type: phones;
 Features:
  Price: LIT. 108000;
  Color: grey, black;
  Agenda: 20;
  ...
 Properties:
  Quality: high;
  Ease of use: high;
  Cheapness: medium;
  Design: high;
  ...
```

Fig. 2. Representation of an item.

– *Domain*: this slot represents the range of values that a feature can take.

3.2 Representation of items.

The Products DB stores the description of all the items available in the store. Each item is described by means of a record with attributes specifying the features and properties of the item: the features correspond to the set of features associated in the Product Taxnonmy to the product class to which the item belongs. Consider, for instance, the record of the "Facile" phone item, shown in Figure 2: the description includes the internal code and the external name of the item, and the product class of the item ("Type" field); moreover, it includes information about the features of the item (e.g., its price, color, etc.); finally, for each product property (e.g., quality, etc.), the record includes a value representing a qualitative evaluation of the item with respect the property.[1]

3.3 Representation of the knowledge about users.

The knowledge about users is represented in a declarative way, by specifying the user features to be modeled and their possible values. A user model contains:

– Descriptive information about the user, concerning personal data, like the user's age, education level and job, provided by the user by filling a registration form.
– Predictive information, concerning the user's preferences for the properties of goods, like quality and ease of use, and user features such as her receptivity, domain expertise, and so forth. These data are represented by means of

[1] While the description of the features is standard for a product database, the specification of the properties represents supplementary information that has to be embedded into the database by the store designer, after an evaluation of the product.

Identification:
First Name: George;
Family Name: Smith;
Personal data:
Age: 55-64;
Job: employee;
Education Level: high_school;
...
User features:
Receptivity:
Values: low: 0.5; medium: 0.25; high: 0.25;
Domain Expertise:
Values: low: 0.5; medium: 0.3; high: 0.2;
Technical Interest:
Values: low: 0.1; medium: 0.5; high: 0.3;
Aesthetic Interest:
Values: low: 0.3; medium: 0.3; high: 0.4;
...
Preferences:
Quality:
Importance: 0.8;
Values: low: 0; medium: 0.4; high: 0.6;
Ease of Use:
Importance: 1;
Values: low: 0; medium: 0.25; high: 0.75;
...
User classification:
life style:
Values: yuppie: 0.2; average: 0.6; modest: 0.2;
domain expertise:
Values: ...

Fig. 3. An example user model.

parameters characterized by an *Importance* slot and a list of <*Linguistic Value, Likelihood*> pairs [40]: the importance slot specifies how relevant is the property to the user and takes values in the range [0..1], where "0" denotes an irrelevant preference, while "1" denotes a maximally important one. Each <*Linguistic Value, Likelihood*> pair contains a linguistic value that the parameter can assume and the likelihood that the user prefers that value for the related product property. For instance, in Figure 3, George's preference towards the products quality is rather important (0.8) and he prefers high-quality products (<low, 0>, <medium, 0.4>, <high, 0.6>).

The data specified in the user models are domain-dependent and have to be defined for each Web store instance; however, the same data could be exploited in several Web stores, if the systems are instantiated on similar sales domains.

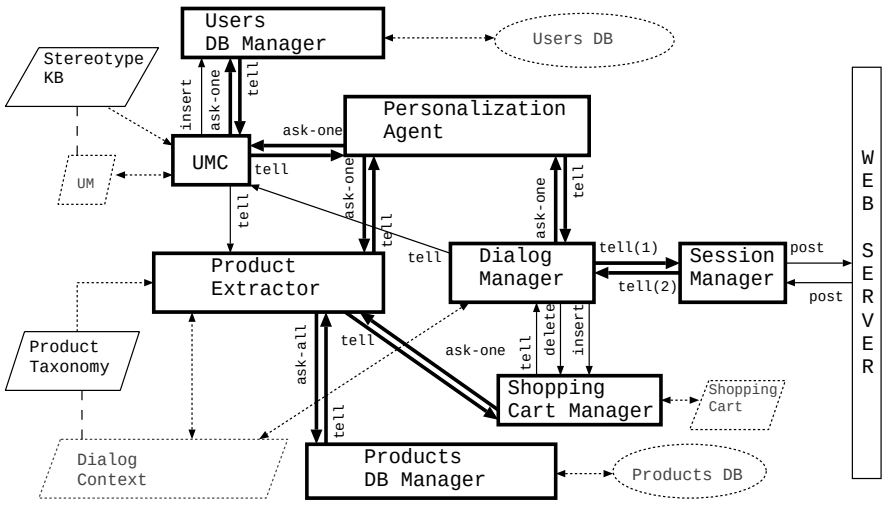

Fig. 4. The Web store architecture: rectangles denote agents; solid parallelograms denote knowledge bases; dashed parallelograms represent the session-dependent contextual information handled by the agents. Bold (thin) arrows represent synchronous (asynchronous) messages between agents.

4 The SETA architecture

In the design of the architecture of our system, we identified a number of functionalities that an adaptive Web store should offer while interacting with a user; then, we isolated the activities necessary to obtain the desired system behavior. The result of this analysis is the identification of a set of basic roles: for instance, the maintenance of the user models containing the customers' preferences and needs, the application of personalization strategies to tailor the generation of the hypertextual pages and the personalization of the suggestion of items. The identified roles are necessary to offer a personalized navigation of a Web catalog, but are not exhaustive: depending on the requested system functionalities, the architecture might need to be further extended; for instance, we did not include the management of orders and payment transactions in the set of basic roles we have developed.

We designed a multiagent architecture where each agent fills a role [43], offering the other agents the services associated to the role. Our multiagent architecture is shown in Figure 4, where the boxes denote agents; the parallelograms represent the knowledge bases used by such agents to retrieve the domain-dependent knowledge (e.g., the description of user and product features); the databases are represented as dashed ovals. Each agent has its own context, denoted by a dashed rectangle, that keeps the session-dependent data: every agent maintains a single context for each active user session. The arrows among agents represent the types of messages they can exchange. In the following, we will describe the main roles and their associated agents.

4.1 Communication with the Web

The Session Manager handles the communication with the browsers and creates the system agents. Each time a user connects to the store (or performs actions in the Web catalog), the Session Manager catches the user's action and sends a synchronous "tell" message to the Dialog Manager, to notify the agent that the event has occurred. The possible actions (e.g., following an hypertextual link, clicking on a button, etc.) can be identified because the related events are tagged by exploiting hidden labels in the HTTP requests.

4.2 Management of the interaction flow

Since the system appears as a dynamic hypertext and the content of the pages is planned "on the fly", on the basis of the user's actions, the interaction flow is not completely determined at the beginning of a user session. The pages produced by the system and the actions performed by the user can be interpreted as very generic turns of a dialog. In order to interact with the user in a rational way, it is important to maintain the (logical) interaction context and decide how the dialog should continue, given its status and the user's previous actions.

In our architecture, this task is performed by the Dialog Manager, that handles the interactions and maintains the contextual information. The Dialog Manager monitors the user's actions, by interpreting the generic events caught by the Session Manager. These events are used to decide how to continue the dialog and to collect data about the user's interests.

From the system's viewpoint, each page represents a dialog turn. Thus, the admissible sequences of turn types can be represented by means of a finite-state automaton which specifies, for each page type, which user actions can be performed (such actions are available via buttons and hypertextual links in the displayed page) and which page type must follow each action type. For instance, from a page describing an item of a product class, the user may follow a "more information" link, in which case, the same type of page has to be displayed next.[2] However, she can also inspect the technical details, create a comparison table, and so forth; in these cases, a parallel page, showing the requested information, is handled by the system.

4.3 Management of the user models

A basic role in an adaptive system is the management of the user models: the characteristics of the specific user have to be identified to apply the appropriate personalization rules and adapt the interaction as a consequence. In SETA, various types of user models can be used: e.g., static user models, instantiated on the basis of the user's features, or dynamic user models, continuously updated on the basis of the user's behavior [26, 41].

[2] If the "more information" link is followed on a page, the page is redisplayed by the system, showing the whole list of features characterizing the item, instead of showing only the most relevant ones.

In the SETA architecture, a User Modeling Component (UMC) creates and maintains the models of customers. When a user accesses the store, her model is initialized either by retrieving her record from the User's DB, or by exploiting stereotypical information about customers available in the Stereotype Knowledge Base (KB). This KB contains a hierarchical taxonomy of stereotypes which cluster typical properties of customer groups [36]. The stereotypes are characterized by a classification part and a predictive part. The classification part concerns socio-demographic characteristics, corresponding to the personal data asked to the user when she first accesses the Web store. The predictive part concerns user features and preferences towards product properties. The user is classified by matching her personal data with the classification part of the stereotypes. Then, her features and preferences are initialized by merging the predictions of the best matching stereotypes [6].

In addition to the stereotypical information, dynamic user modeling techniques have been recently introduced in the system to update the user models on the basis of the users' behavior during the interaction. The UMC maintains in its working memory the history of all the relevant events delivered by the Dialog Manager. The events may regard clicks to hypertextual links and clicks on buttons to perform specific tasks (e.g., putting an item into the shopping cart). The UMC periodically analyzes the event history to refine the user model, so that it represents a description of the user consistent with her recent behavior.

In other approaches, such as [18, 25, 31], the user models are instantiated and updated on the sole basis of an unobtrusive observation of the customer's behavior. While we recognize that the minimization of questions is essential to improve the acceptability of the system, we believe that the exploitation of both implicit and explicit information is crucial to handle precise user models, which can hardly be obtained on the sole basis of the observation of the user's behavior. In particular, in our system the exploitation of the stereotypical information about customers is essential to initialize the user models and start a personalized interaction immediately after a customer accesses the store. Then, while she browses the catalog, dynamic user modeling techniques are exploited to revise her model on the basis of her individual behavior.

4.4 Generation of the catalog pages

Once the next interaction step is decided, the appropriate hypertextual page has to be dynamically generated. On the basis of the purpose of the system, the content displayed in the pages may strictly depend on the dialog context, or it may also vary on the basis of the user's interests; for instance, specific product features can be highlighted, or presented in a more or less technical way.

In our architecture, the Dialog Manager invokes the Personalization Agent, specifying the type of page to be produced and the product to be described. The Personalization Agent applies a set of customization rules and dynamically generates the HTML code for the hypertextual pages.

All the decisions about the layout and the content of a page are made on the basis of the user model, at the granularity level of the single product fea-

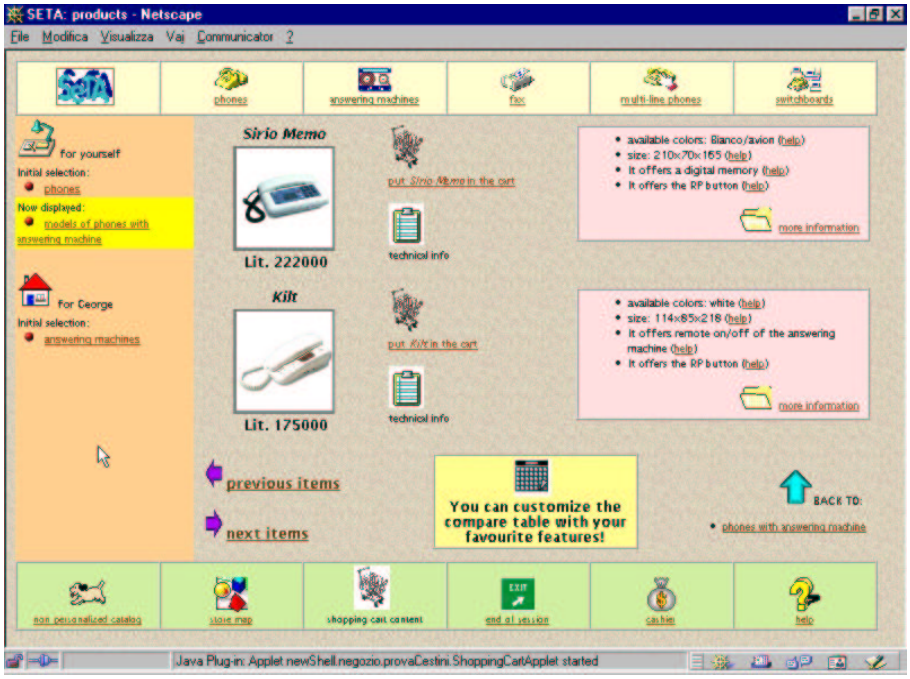

Fig. 5. A page dynamically generated by our system.

ture; moreover, the data to be included in the presentations are selected from the Products DB, which contains the whole information about the items. Thus, significantly different descriptions may be produced when presenting the same product to different users; furthermore, no additional (and redundant) information is needed to generate the product descriptions.

Figure 5 shows a page produced by the Personalization Agent of our prototype instantiated on the telecommunication domain. The page describes two phones with answering machine and contains both navigation buttons and presentation areas. More specifically:

- The upper bar in the page allows the user to inspect the main product classes described in the catalog: the bar can be used to select further products to be examined.
- The lefthand side area is a synthetic description of the interaction context: the products initially selected by the user for inspection are listed ("Initial selection"), specifying the intended beneficiary ("for yourself", "for George") and use (home vs. business, represented by means of icons). Moreover, for each selection, the last product displayed by the system is reported ("last visited", not shown in the figure), so that she can remember which hypertextual node she visited most recently in that context. Finally, the active context, associated to the product displayed in the main area of the page,

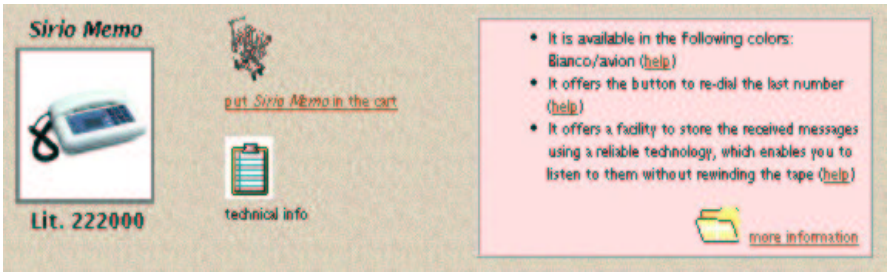

Fig. 6. A page dynamically generated by our system.

is highlighted ("Now displayed"). In this example, the user is looking for a phone with answering machine for her office (business use); she is also looking for a simple answering machine for her friend George (home use).
- The central area of the page describes the available models. For each of them, the area contains the name, a picture, the price, a button to put the item into the shopping cart, another one to get the technical details, and a description of the main features of the item itself: the system may display a subset of the features describing the item, to focus on those most interesting to the user.
- The rest of the page contains links and buttons available to perform actions like browsing the catalog, creating a personalized comparison table, inspecting the content of the shopping cart, and so forth.

The page shown in Figure 5 is tailored to an expert user, supposed to understand technical terms (like "digital memory", "RP button") and to be able to absorb relatively large amounts of information: for example, the system shows two items in a single page, instead of one. The description of item "Sirio Memo" in this page may be compared with that of Figure 6, representing the area describing the same item, extracted from a page generated for a non-expert user. As it can be noticed, the language used in Figure 6 is much simpler, the complex concepts are explained (e.g., consider "digital memory" vs. "It offer a facility to store the received messages..."), and fewer features are described to let the user focus on the most important ones.

The Personalization Agent plans the content and layout of a page by exploiting a set of personalization rules [6], used to:

- *Rate the relevance of the information items on the basis of the user's interests and of the intrinsic importance of the features to the presentation of the item.* There is a direct correspondence among the user's interests and the types of features defined in the Product Taxonomy. The features of an item are ranked by combining their importance with the user's interest in an additive formula: $score_F = w_1 * Imp_F + w_2 * Interest_T$
where Imp_F is the importance of F and (given the category T of features to which F belongs) $Interest_T = 1$(2, or 3) if the user's interest in T is

low (medium, or high). The value of the user's interest is retrieved from the user model, as the most probable linguistic value of the related user feature. For instance, referring to Figure 3, George's interest for technical features is medium, while his interest for aesthetic features is high.
- *Select the amount of information to be included in the page, depending on the user's receptivity.*
- *Choose the appropriate complexity level for the descriptions, on the basis of the user's domain expertise, and produce the descriptions to be included in the page.* The descriptions are structured as Natural Language templates, characterized by a difficulty level which basically depends on their technicality. For each feature to be described, the system extracts from the Products DB the values of the feature offered by the item and then it generates the complete linguistic description by filling in the place-holders of the template with the linguistic expressions of such values. For instance, the description of the "color" feature is generated by exploiting its template ("It is available in the following colors: #0") and replacing the place-holder ("#0") with the linguistic description of the colors offered by the item (e.g., "white").
- *Refine the layout of the features, choosing special sizes and styles (e.g., the boldface), depending on their ratings.*

4.5 Suggestion of items

An important role in the architecture is the retrieval of the items to be suggested on the basis of the user's preferences. In our system, the Product Extractor assists the user's selection of items, interacting with the Products DB Manager to retrieve the internal description of the items which might be suggested. In order to let the user inspect all the items available for a product class, the system does not filter out any item: thus, the suggestion consists of sorting the items, so that the most promising ones are shown to the user before the others.

The Product Extractor ranks the items by evaluating how close their properties match the beneficiary's preferences [6]. An individual item is ranked by evaluating its properties one by one and combining the results of such evaluation into an overall score, used to place the item in the sorted suggestion list. The general requirement is that an item should be preferred if it matches all the preferences important to the user; in contrast, the item could mismatch irrelevant user preferences. The individual score of a property A is a number in [0..1], where "0" denotes the fact that the item mismatches the related user preference, while "1" denotes a full match. This second value can occur in two main cases: either the preference is important to the user and the item matches it perfectly, or the preference is totally irrelevant and thus it should not be considered in the evaluation process. For each item property (A), we compute the individual score by applying a formula which combines the importance of the related user preference (Imp_A) with the likelihood associated in the user model to the linguistic value of A fitting the item (p_{a_i}):

$sc_A = Imp_A * p_{a_i} + (1 - Imp_A)$.

Basically, we want that an irrelevant property receive scores near to "1", inde-

pendent of how close it reflects the user's preferences: in this way, the influence of the property on the matching degree decreases proportionally to the importance of the user preference. In contrast, a score near to "1" should be given to a very influent property only when the likelihood that the user prefers products with that property value is very high.

The individual scores of the properties are then combined in a fuzzy AND: given two properties A and B, and their individual scores, sc_A and sc_B, the overall score for the item is calculated as follows:

$SCORE(sc_A, sc_B) = sc_A * sc_B/(sc_A + sc_B - sc_A * sc_B)$.

This formula produces matching degrees in [0,1]: if at least one of the individual scores is null, the overall score is null (in fact, the related property is totally incompatible with the user's preferences); in contrast, an item perfectly matching the user's preferences has an overall score equal to "1".

4.6 Management of the databases

A Web store architecture needs to store into a database the data about the specific items available in the store, as well as the information about its customers; the last information concerns the customer's previous connections to the store, purchases, preferences, and so forth. The data about customers enhance the system's capability to personalize the interactions, because they enable it to load precise user models at the beginning of the interaction.

In our architecture, we have designed two database managers (Products and Users DB Manager) that handle the Products and Users databases respectively. The Products database contains the information about the goods sold in the store (price, technical features, etc.), while the Users database contains the records of the customers who have accessed the store, including the goods that they purchased in previous visits.

4.7 Management of the customer's selections

A Shopping Cart Manager keeps track of the items selected by the user during an interaction: at any time, items can be put into the cart, or removed, and the cart always displays its content and the total amount of money to pay.

5 Comments

In addition to the roles described in the previous section, the architecture of a Web store should include other essential roles, such as the management of the orders and payment transactions, stocks, and so forth. We have omitted the roles concerning the management of the supply chain, and we have just sketched the one regarding the purchasing activity (see the Shopping Cart Manager). However, the design of the architecture as a multi-agent system supports the extension of this architecture to include such back-end activities as well.

Indeed, the exploitation of agent-based technologies has been very useful for the design of our architecture, as well as for the implementation of our Web store shell, for the following reasons:

- Components filling very different roles have to be coordinated within a single architecture and this fact may create serious organizational problems: these components may use heterogeneous knowledge sources and technologies; still, they have to cooperate with one another to offer the overall service to the user. For instance, the Session Manager handles the multi-user access to the store. This component operates in a strict event-driven way and works on low-level data which represent the events occurring in the active user sessions. On the other hand, other components carry on inferential processes; these components work on complex data structures and perform high-level reasoning tasks by applying different approaches, like for example the exploitation of production rules.
- Some of the roles fit well in a traditional Object-Oriented programming paradigm; however, others require that the components filling them are proactive and may initiate tasks although their methods are not explicitly invoked [30]. For example, the shopping cart might be monitored to check whether the customer's selections are consistent (for example, items such as a computer and a printer, which have to work together, must be compatible): in case of inconsistencies, the controller should autonomously trigger a dialog to warn the user about the problem. Although not all the components have to be represented as agents [42], the overall system architecture needs to exploit the technologies developed in the design of multiagent architectures.
- For efficiency purposes, the activity of the agents has to be performed in parallel whenever possible. Therefore, a sequential interaction model has difficulties in scaling up. On the other hand, agent-based technologies offer different types of communication, including synchronous and asynchronous messages, that enhance the parallelism in the execution of tasks.
- Although we have developed the architecture of a single marketplace, an interesting extension is the possibility provide broker agents with information about the items available in the Web store. This scenario raises interoperability issues, that can be faced thanks to the exploitation of agent communication languages and, at a lower level, of the facilities to interact in heterogeneous communication platforms (CORBA, RMI), offered by several tools to build multiagent systems.
- Although the integration of a new component into a complex system requires a careful design of its interaction with the rest of the system, the exploitation of agent-based technologies facilitates the task, by supporting the communication among the components in a seamless way.
- Other technical reasons have influenced our design decisions. For instance, the seamless distribution facilities offered by many agent-based technologies: in fact, given a complex system, the distribution of its components on several computers may be desirable and our experience showed that the exploitation of agent-based communication techniques and specific tools to build multiagent systems allows the developer to distribute agents easily and efficiently.

6 Technical details

The system architecture, described in detail in [5], is a parallel architecture where synchronous, asynchronous and multicast messages can be exchanged and handled by the agents. The asynchronous messages are exploited to enhance the efficiency of the system: we selected the activities that could be carried on in parallel and we identified a number of messages that can be handled asynchronously by the agents. We described the messages as performatives in a speech-act based agent communication language. Figure 4 shows the types of messages exchanged by the agents during a working session: the thin arrows denote asynchronous messages, while the thick ones denote synchronous messages. Each arrow has associated the message type sent by the agent; we have labeled the message types with names derived from the KQML performatives [20].

We have implemented this architecture by exploiting Objectspace Voyager [29], a tool for developing multiagent systems which provides system components with basic agent capabilities, such as distribution and communication protocols. The agents of our system are Java objects which offer all the services necessary to carry on the personalized interactions with customers; moreover, we have exploited Voyager's facilities to allow the agents' distribution over different computers and their communication. While synchronous messages are handled by the main thread of an agent, the multithread environment supported by Voyager enables the agents to spawn in order to handle the asynchronous messages; in this way, they can also manage the active user contexts in parallel. In this way, we can handle in a homogeneous way an almost complete parallelization of the activity of the various agents within a user session, as well as different sessions.

The only exception is the Session Manager, which is implemented as a Servlet. Servlets are used to enhance the functionalities of Web-based systems, by extending Web server capabilities. In particular, we have exploited them to handle the communication with the browsers (catching users' actions and sending to the users' browsers the Web pages of the store) and to conveniently manage concurrent user sessions (by exploiting session tracking) in our Session Manager.

We have considered several tools for building multiagent systems in a Java-based environment: all these tools offer communication and distribution facilities, and introduce an abstraction level with respect to the communication protocol, which might be RMI, DCOM, CORBA, or other. We have selected Objectspace Voyager [29], which best suited the needs of our architecture, allowing a convenient object distribution: objects created by the Voyager compiler can be remotely executed. Moreover, Voyager supports an almost seamless transformation of a Java object, which can only exchange synchronous messages, into an agent able to send and receive various types of messages (noticeably, although different message synchronization types are supported, the Java method declaration does not change). Voyager offers synchronous, oneWay, oneWayMulticast and "future" messages ("future" messages correspond to asynchronous messages, which do not involve the sender and the receiver in a rendez-vous). Messages are exchanged among agents by method invocation; however, the object invoking a

method has to specify in which way it wants to synchronize with the receiver. While synchronous communication corresponds to traditional Java method invocation, the other types of messages can be delivered by objects by invoking specific Voyager methods.

Other systems, like JAFMAS [15] and JATLite [1], would allow our agents to be distributed on a network and communicate in a standard agent communication language based on Speech Acts [38, 20, 30]. However, following KQML, they impose that the contents of the speech acts are represented, in an appropriate content language, as strings. Therefore, they make the exchange of messages containing structured information a complex task.[3] On the other hand, in the development of our system, we have experienced that a huge portion of the information which the agents need to send to each other is embedded into complex objects and a conversion of such data into very general String patterns would reduce the efficiency in agent communication.

Since we need flexibility in agent communication, we excluded other well known agent building tools, specifically focused on the management of agent mobility, which handle agent communication at a rather low level: e.g., MOLE [11], and IBM Aglets [23]. At the same time, other tools for building open multiagent systems, which also provided negotiation and coordination primitives to enable an active cooperation between the agents, proved to exceed our needs: for instance, the Agent Building Shell [10]. In fact, in our system, the agents offer fixed services, and there is no need to dynamically distribute tasks among them.

It should be noticed that the SETA architecture represents a quite well-established architecture for complex Web-based systems. In particular, the most recent research has triggered the development of other frameworks for the creation of these types of systems. For instance, the Jackal tool for the development of multiagent systems [4] supports a rich, KQML-based communication among agents; moreover, the Java 2 Enterprise Edition by Sun Microsystems provides the developer of a distributed, Web-based system with all the facilities for the management of parallel user sessions and also supports a transactional access to databases. Such frameworks were not available when we developed SETA; however, our approach is compatible with the one adopted by these tools; thus, the SETA system could be updated to exploit such environments, if needed.

7 Conclusions

In this paper, we have discussed the importance of the personalization of the interaction with customers in electronic commerce, specifically referring to the case of adaptivity in Web stores. In particular, we claim that these stores need to be adaptive and to tailor the description of the catalogs to the needs and capabilities of the specific customer. In fact, customers are heterogeneous and have different demands, ranging from the amount of information that they want

[3] Object serialization could be used to transmit messages containing object parameters, but a sensible effort is required to decode and parse the content of messages on the receiver's side.

to receive, to the content of the descriptions, that may be more or less compact, technical, or may focus on different product features, depending on the customer's interests.

In the paper, we have also discussed the importance of the exploitation of agent-based technologies in the design of adaptive Web stores: these stores requires the design of complex architectures, where differentiated roles can be identified. Agent-based technologies are very important to manage this complexity: in fact, they support the development of heterogeneous, possibly distributed systems; moreover, they offer powerful communication languages, that can be exploited to deliver different types of messages, such as synchronous and asynchronous messages. Finally, they support the interoperability with respect to the communication platform, therefore making the development of open systems an easier task.

8 Acknowledgments

This work has been developed in the project "Servizi Telematici Adattativi" (http://www.di.unito.it/~ seta), carried on at the Dipartimento di Informatica of the University of Torino within the national initiative "Cantieri Multimediali", sponsored by Telecom Italia.
Many thanks to R. Meo, C. Barbero and all the students who have contributed to the development of the SETA system.

References

1. JATLite. http://java.stanford.edu/java_agent/html.
2. AIMedia. @media, communicating on interactive media. http://www.aimedia.org, 1999.
3. M. Albers, C.M. Jonker, M. Karami, and J. Treur. An electronic market place: generic agent models, ontologies and knowledge. In *Proc. of the Agents'99 Workshop: "Agent-based decision-support for managing the Interned-enabled supply-chain"*, pages 71–80, Seattle, WA, 1999.
4. IBM alphaWorks. Jackal. http://jackal.cs.umbc.edu/, 1999.
5. L. Ardissono, C. Barbero, A. Goy, and G. Petrone. An agent architecture for personalized web stores. In *Proc. 3rd Int. Conf. on Autonomous Agents (Agents '99)*, pages 182–189, Seattle, WA, 1999.
6. L. Ardissono and A. Goy. Tailoring the interaction with users in electronic shops. In *Proc. 7th Int. Conf. on User Modeling*, pages 35–44, Banff, Canada, 1999.
7. L. Ardissono, A. Goy, R. Meo, and G. Petrone. An agent architecture for personalized interaction with customers in virtual stores. In *Proc. Int. IFIP/GI Working Conf. on Trends in Distributed Systems for Electronic Commerce (TrEC'98)*, pages 137–148. Dpunkt Verlag, Heidelberg, Germany, 1998.
8. L. Ardissono, A. Goy, R. Meo, G. Petrone, L. Console, L. Lesmo, C. Simone, and P. Torasso. A configurable system for the construction of adaptive virtual stores. *World Wide Web*, 2(3):143–159, 1999.
9. M. Balabanovič. Exploring versus exploiting when learning user models for text recommendation. *User Modeling and User-Adapted Interaction*, 8:71–102, 1998.

10. M. Barbuceanu and R. Teigen. Higher level integration by multi-agent architectures. In P. Bernus, editor, *Handbook of Information System Architectures.* Springer Verlag.
11. J. Baumann, F. Hohl, N. Radouniklis, K. Rothermel, and M. Straβer. Communication concepts for mobile agent systems. In *Proc. 1st Int. Work. on Mobile Agents (MA'97)*, 1997.
12. D. Benyon. Adaptive systems: a solution to usability problems. *User Modeling and User-Adapted Interaction*, 3:65–87, 1993.
13. D. Billsus and M. Pazzani. A personal news agent that talks, learns and explains. In *Proc. 3rd Int. Conf. on Autonomous Agents (Agents '99)*, pages 268–275, Seattle, WA, 1999.
14. P. Brusilovsky. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6(2-3):87–129, 1996.
15. D. Chauhan. *JAFMAS: A Java-based Agent Framework for Multiagent Systems Development and Implementation.* PhD thesis, University of Cincinnati, Stanford, CA, 1997.
16. Y. Chen, Y. Peng, T. Finin, Y. Labrou, S. Cost, B. Chu, J. Yao', R. Sun, and B. Wilhelm. A negotiation-based multi-agent system for supply chain management. In *Proc. of the Agents'99 Workshop "Agent-Based Decision-Support for Managing the Internet-Enabled Supply-Chain")*, pages 15–20, Seattle, WA, 1999.
17. R. Dale, S.J. Green, M. Milosavljevic, and C. Paris. Dynamic document delivery: Generating natural language texts on demand. In *Proc. 9th Int. Conf. and Workshop on Database and Expert Systems Applications (DEXA'98)*, Vienna, 1998.
18. M. Dastani, N. Jacobs, C.M. Jonker, and J. Treur. Modelling user preferences and mediating agents in electronic commerce. In *Proc. of the Agent-Mediated Electronic Commerce (AMEC) SIG-Meeting of AGENTLINK*, Barcelona, 1999.
19. O. Etzioni. Moving up the information food chain: deploying doftbots on the World Wide Web. In *Proc. 14th Conf. AAAI*, pages 1322–1326, Portland, 1996.
20. T.W. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents.* MIT Press, Cambridge, 1995.
21. J. Fink, A. Kobsa, and A. Nill. Adaptable and adaptive information for all users, including disabled and elderly people. *New review of Hypermedia and Multimedia*, 4:163–188, 1998.
22. A.R. Greenwald, , and J.O. Kephart. Shopbots and pricebots. In *Proc. 16th IJCAI*, pages 506–511, Stockholm, 1999.
23. IBM. Aglets. http://www.trl.ibm.co.jp/aglets/whitepaper.htm.
24. A. Jameson, R. Shäfer, J. Simons, and T. Weis. Adaptive provision of evaluation-oriented information: tasks and techniques. In *Proc. 14th IJCAI*, pages 1886–1893, Montreal, 1995.
25. T. Joerding. Intelligent multimedia presentations in the web: Fun without annoyance. In *Proc. of the 7th World Wide Web Conference (WWW7)*, Brisbane, Australia, 1998.
26. R. Kass and T. Finin. Modeling the user in natural language systems. *Computational Linguistics*, 14(3):5–22, 1988.
27. M.F. McTear. User modelling for adaptive computer systems: a survey of recent developments. *Artificial Intelligence Review*, 7:157–184, 1993.
28. M. Milosavljevic and J. Oberlander. Dynamic hypertext catalogues: Helping users to help themselves. In *Proc. the 9th ACM Conference on Hypertext and Hypermedia (HT'98)*, Pittsburgh, PA, 1998.
29. ObjectSpace. Voyager. http://www.objectspace.com/index.asp, 2000.

30. C.J. Petrie. Agent-based engineering, the web, and intelligence. *IEEE Expert*, December:24–29, 1996.
31. W. Pohl and A. Nick. Machine learning and knowledge representation in the LabouUr approach to user modeling. In *Proc. 7th Int. Conf. on User Modeling*, pages 179–188, Banff, Canada, 1999.
32. H. Popp and D. Lödel. Fuzzy techniques and user modeling in sales assistants. *User Modeling and User-Adapted Interaction*, 6:349–370, 1996.
33. A. Preece, K. Hui, and P. Grey. KRAFT: supporting virtual organizations through knowledge fusion. In *Proc. of the Agent-Mediated Electronic Commerce (AMEC) SIG-Meeting of AGENTLINK*, Barcelona, 1999.
34. B. Raskutti, A. Beitz, and B. Ward. A feature-based approach to recommending selections based on past preferences. *User Modeling and User-Adapted Interaction*, 7:179–218, 1997.
35. P. Resnick and H.R. Varian, editors. *Special Issue on Recommender Systems*, volume 40. Communications of the ACM, 1997.
36. E. Rich. Stereotypes and user modeling. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 35–51. Springer Verlag, Berlin, 1989.
37. N.M. Sadeh, D.W. Hildum, D. Kienstad, and A. Tseng. MASCOT: an agent-based architecture for coordinated mixed-initiative supply chain planning and scheduling. In *Proc. of the Agents'99 Workshop: "Agent-based decision-support for managing the Interned-enabled supply-chain"*, pages 133–138, Seattle, WA, 1999.
38. D. Steiner. An overview of FIPA 97. http://drogo.cselt.stet.it/fipa/#Papers, 1997.
39. K.P. Sycara, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, December:36–45, 1996.
40. P. Torasso and L. Console. *Diagnostic Problem Solving*. North Oxford Academic, 1989.
41. W. Wahlster and A. Kobsa. *User Models in Dialog Systems*. Springer Verlag, Berlin, 1989.
42. M.J. Wooldridge and N.R. Jennings. Pitfalls of agent-oriented development. In M. Wooldridge and N.R. Jennings, editors, *Proc 2nd Int. Conf. on Autonomous Agents (Agents98)*, pages 385–391. ACM Press, Minneapolis, 1998.
43. M.J. Wooldridge, N.R. Jennings, and D. Kinny. A methodology for Agent-Oriented analysis and design. In *Proc. 3rd Int. Conf. on Autonomous Agents (Agents '99)*, pages 69–76. Seattle, WA, 1999.
44. F. Ygge. Software agent for electronic power trade. In *Proc. of the Agent-Mediated Electronic Commerce (AMEC) SIG-Meeting of AGENTLINK*, Barcelona, 1999.
45. D.D. Zeng and K.P. Sycara. Agent-facilitated real-time flexible supply chain structuring. In *Proc. of the Agents'99 Workshop "Agent-Based Decision-Support for Managing the Internet-Enabled Supply-Chain")*, pages 21–28, Seattle, WA, 1999.