An Architecture for Adaptive Web Stores

Giovanna Petrone

Dipartimento di Informatica, University of Torino
{giovanna}@di.unito.it

1 Introduction

In the last three years, we have developed SETA, a prototype toolkit for the creation of Web stores personalizing the interactions with users, focusing on the design of the front-end of on-line stores and on the development of a flexible interface. We have exploited knowledge representation techniques and agent-based technologies to improve the configurability of the toolkit and its scalability [2, 3]. Moreover, we have organized the overall architecture as a multiagent one, where specialized agents fill the main roles for the management of personalized interactions with customers [8]. An on-line demo of a prototype store created using SETA is available at the following URL: http://www.di.unito.it/~seta. This store presents telecommunication products, like phones and switchboards, and will be used throughout the rest of the paper as a concrete example to describe the functionalities of our system.

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

- Components filling different roles have to be coordinated in a single architecture and this fact may create serious organizational problems: the components may use heterogeneous knowledge sources and technologies; still, they must cooperate with one another to offer the overall service to the user.
- Some roles fit well in a traditional Object-Oriented programming paradigm [7]; however, others require that the components filling them are proactive and may initiate tasks although their methods are not explicitly invoked.
- 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 to 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.

- The exploitation of agent-based technologies facilitates the extension of complex systems with new components.

- 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.

The system architecture, described in detail in [2], is a parallel architecture where synchronous, asynchronous and multicast messages can be exchanged and handled by the agents. We described the messages as performatives in a speech-act based agent communication language [5].

We have implemented this architecture by exploiting Objectspace Voyager [6], 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 offering the services necessary to carry on the personalized interactions with customers; moreover, we have exploited Voyager’s facilities to allow the distribution of agents 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 for handling the asynchronous messages; in this way, they can also manage the active user contexts in parallel. Thus, 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.

2 Infrastructures for Developing Multi-agent Systems

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 [6], which best suited the needs of our architecture, allowing a convenient object distribution: objects created by the Voyager compiler can be remotely