ADVISOR SUITE: A Tool for Rapid Development of Maintainable Online Sales Advisory Systems

Dietmar Jannach and Gerold Kreutler

University Klagenfurt
Universitätsstraße 65
9020 Klagenfurt, Austria
0043 463 2700 3757
{dietmar.jannach, gerold.kreutler}@uni-klu.ac.at

Abstract. A sales advisory system is a tool supporting customers in the decision-making and buying process by interactive and personalized requirements elicitation and the provision of comprehensible product proposals and explanations. The particular challenges when building such systems lie in the strong interdependencies between the recommendation and personalization logic and the corresponding adaptive, web-based user interface. The ADVISOR SUITE tool described in this paper is a system that follows a consistent knowledge-based approach for all tasks that are required to build such intelligent sales advisory systems for arbitrary domains. The development of advisory applications is based on a conceptual model of online sales dialogues, a "Model-View-Controller" application architecture, a generic controller component, as well as (semi-)automatic, template-based web page generation. Experiences from various real-world applications show that the knowledge-based approach and the corresponding graphical tools of ADVISOR SUITE significantly accelerate the development and maintenance process for such applications.

1 Introduction and Overview

Online customers are increasingly overwhelmed by the variety of comparable products or services available on the online channel. Web-based sales assistance and recommendation systems are a means for supporting online customers in their product selection and decision-making processes. These systems provide the best value for the customers when they simulate the behavior of a real sales assistant. Therefore, they acquire the customer's real needs in a personalized dialogue ([1], [2]), and come up with a suitable set of proposals and provide adequate explanations for these proposals, which is required to increase the customer's confidence in his buying decision.

The purpose of the ADVISOR SUITE framework described in this paper is to provide a domain-independent tool for integrated development and maintenance of such web-based sales advisory applications. At the core, ADVISOR SUITE is an expert system where the knowledge of the domain expert is made explicit in a declarative knowledge base. This knowledge both comprises the recommendation guidelines of the domain, as well as information about *how* the real customer requirements have to be elicited, i.e., knowledge about efficient sales dialogues. The main challenge of such an approach is that there are strong interrelations between these two types of knowledge. Typically, the user's preferences are acquired by asking questions in an interactive dialogue. The user's answers (i.e., his/her profile) obviously influence the personalized set of products to be recommended, but also determine the further dialogue flow which should be adapted, e.g., to the user's skill level.

Consequently, the web pages used in the online dialogue must be extremely flexible and dynamic such that changes in the knowledge base are immediately taken into account and do not require manual adaptation of the dynamic HTML code. Nonetheless, the web pages have to be comprehensible and editable by a Web developer who aligns the pages' style to the corporate layout or integrates the application into an existing web site.

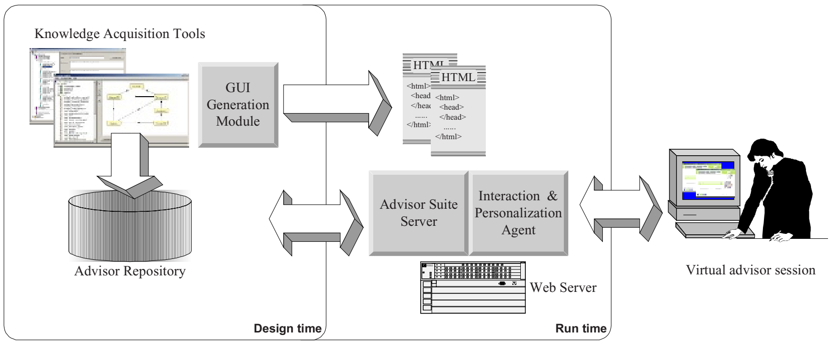

Fig. 1. Architecture of the framework

Fig. 1 shows the overall architecture of the ADVISOR SUITE framework. The required knowledge is acquired using graphical knowledge acquisition tools and stored in a common repository. The *server* component utilizes that knowledge and creates *interaction and personalization agents* that manage the user input for each advisory session. Before the system is started, the needed web pages for the dialogue and a generic controller component implementing the personalization logic are generated.

2 Combining the Recommendation Logic and Adaptation Logic

In our approach, the *customer properties* are the glue between the recommendation logic and the personalized adaptation of the user interface and the dialogue. First, these properties, i.e., the user's interests and preferences, determine the products to be proposed and their degree of fit with the requirements. The possible values (answers) for the properties are typically finite and pre-defined. The computation of suitable products is based on a priority-based filtering technique similar to [3] and declarative rules like,

"If the customer's experience in the domain is low and he has limited financial reserves, we propose low-risk investments."[1]

[1] Simplified example taken from the domain of online investment advisory.

Such advisory rules are modeled in a graphical tool and are expressed in a high-level, end-user oriented language; the individual ranking of the remaining products is based on a standard personalization technique [4] and the evaluation of the products' utilities for the customer. More details on the knowledge-based recommendation approach can be found in [5].

On the other hand, the customer properties also steer the interaction process, i.e., the sequence of the dialogue pages. In many cases, for instance, the interaction style depends on the user's self estimate of his knowledge level in the domain. Expert users can be asked fewer, but more complex and more technical questions, novice users might need more help and a different form of explanations. Within the ADVISOR SUITE framework, this adaptation and personalization knowledge that an experienced sales agent will have is also made explicit. Again, the personalization process is driven by the customer properties and made explicit with rules like,

"If the user has limited knowledge on the domain, proceed to a page where he is asked if he wants to have a look on more introductory material."

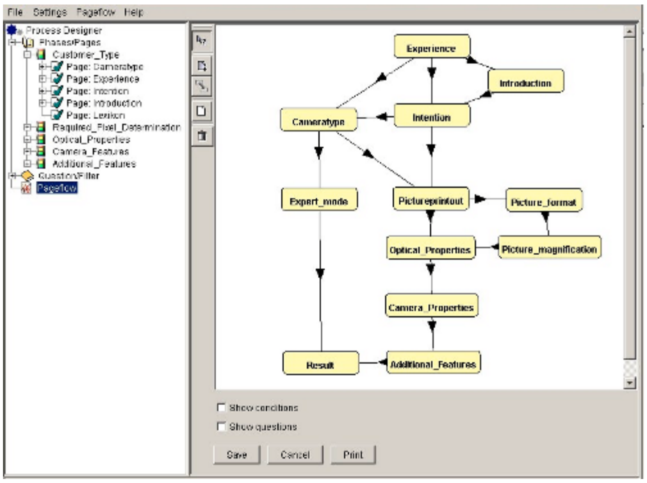

Fig. 2. Modeling the dialogue

These rules are maintained with the help of a special development tool which is depicted in Fig. 2. In particular, the used modeling approach is based on a simple but general *conceptual model* of a sales advisory dialogue, where the major concepts are chosen in a form such that they are close to the resulting web application and use a non-technical representation: Basically, sales assistance dialogues consist of dialogue steps (pages) that contain one or more questions on customer preferences or desired product properties. Each dialogue step can have a number of possible successor pages, whereby the actual successor page is determined dynamically based on the personalization rules described above. A dialogue can have "special steps", like hints on conflicting requirements or additional information or result and explanation pages. Our experiences from several practical applications show that domain experts are able to cope with this level of complexity and can describe the structure of a good dialogue using these concepts after a very short training phase. An important factor for user acceptance also lies in the conceptual integration of this knowledge with the recommendation logic, e.g., the same customer properties used for product filtering now appear as questions in the dialogue; the language for expressing complex conditions is the same as for page successor relations and for filtering rules.

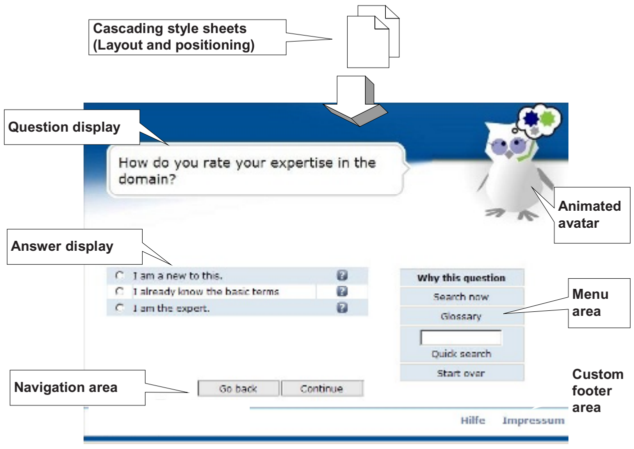

Fig. 3. Structure of a dialogue page

3 User Interface Generation

The dynamic HTML pages of the final application have to be very flexible and must immediately reflect changes in the knowledge base, e.g., when a new question is defined. Moreover, they have to be simple enough such that they can be easily adapted by a Web developer who for instance wants to change the layout of the pages. ADVISOR SUITE uses the following basic techniques to deal with that problem. First, we make extensive use of *Custom Tags* [6] which are syntactically similar to standard HTML tags, but implement application-specific functionality. The usage of these tags helps us both to avoid the problematic mixture of static HTML-code with procedural scripting code and also leads to a more clear and legible page source code. On the other hand, we also rely on automatic web page generation based on an elaborated template mechanism. Each dialogue page is actually built up from different, predefined areas and like question or explanation areas, headers and footers. The *GUI Generation Module* in Fig. 1 automatically assembles and parameterizes the needed web pages from small, pre-defined templates such that the basic dialogue can be generated (in a rapid prototyping process) without programming. Fig. 3 shows the layout and the different areas of such a generated page.

4 Conclusions

Over the last years, several approaches (e.g., [7], [8], [9], [10], and [11]) have been presented that aim at applying state-of-the-art Software Engineering practices to the specific requirements of the development of web applications. To some extent, our work can be seen as an implementation of best-practices from these approaches for a specific application domain: The design process is based on a generic, conceptual model of a sales advisory application; the separation between the business logic, controller, and presentation layers is very rigid. As such, our system shows the practical applicability of the basic idea of these approaches.

However, our work differs from most of the above-mentioned approaches as we want to support the full development process up to the automatic generation of the web pages; although we limit ourselves to a specific class of web-based systems, a broader analysis of the applicability of the presented ideas will be part of our future work. Another difference lies in the choice of the modeling notation, where we deliberately did not use a standard technique, e.g., based on UML (Unified Modeling Language). With the goal of short training times for the domain expert, we opted for a proprietary, end-user oriented notation with a defined semantics that is needed for automated application generation. Our future work will include a detailed evaluation of the applicability of standard modeling techniques from the field of Software Engineering for domain experts with limited background in that area.

References

[1] Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R., and Zanker, M.: A Framework for the Development of Personalized, Distributed Web-Based Configuration Systems, *AI Magazine*, Vol. 24(3) Fall 2003, 93-110.

[2] Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schäfer, R., and Zanker, M.: Personalizing on-line configuration of products and services, *Proceedings 15th European Conference on Artificial Intelligence*, Lyon, France, IOS Press, 2000.

[3] Schiex, T., Fargier, H., Verfaille, G.: Valued Constraint Satisfaction Problems: Hard and Easy Problems, *International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995, 631-639.

[4] von Winterfeldt, D., Edwards, W.: *Decision Analysis and Behavioral Research*, Cambridge University Press, Cambridge, UK, 1986.

[5] D. Jannach and G. Kreutler, Building on-line sales assistance systems with ADVISOR SUITE, *Proc. of 16th Intl. Conference on Software Engineering and Knowledge Engineering* (SEKE'04), Banff, CAN, 2004.

[6] Goodwill, J.: *Mastering JSP Custom Tags and Tag Libraries*, Wiley Publishers, 2002.

[7] Ceri, S., Fraternali, P., and Matera, M.: Conceptual Modeling of Data-Intensive Web Applications, *IEEE Internet Computing*, Vol. 6 , No. 4, pp. 20-30.

[8] Conallen, J.: Building Web Applications with UML, Addison Wesley, Reading, MA, 2000

[9] Rossi, G., Schwabe, D., Esmeraldo, L., Lyardet, F.: Engineering Web Applications for Reuse, *IEEE Multimedia* 8(1), 2001, pp. 20-31.

[10] Jacyntho, M. D., Schwabe, D., Rossi, G.: A Software Architecture for Structuring complex Web Applications, *Journal of Web Engineering*, 1 (1), October, 2002, pp. 37-60.

[11] Gomez, J., Cachero, C., Pastor, O.: Extending a Conceptual Modelling Approach to Web Application Design, *Proc. of the 1st International Workshop on Web-Oriented Software Technology*, Valencia, Spain, June 2001.