Building on-line sales assistance systems with ADVISOR SUITE

Dietmar Jannach and Gerold Kreutler
Institute for Business Informatics and Application Systems, University Klagenfurt
{dietmar,gerold}@ifit.uni-klu.ac.at

Abstract. Online sales advisory systems guide customers through the decision and buying process and provide added value for online customers, especially in domains where a wide range of comparable products is available and differentiation between the products requires deep technical knowledge. In this paper we present ADVISOR SUITE, an expert system that combines knowledge-based approaches for personalized product recommendation with an adaptive user interface, which is used to guide the customer through the requirements elicitation process according to his personal needs and preferences. A particular focus of the presented system lies on the reduction of costly knowledge engineering and maintenance times, which is addressed by a consistent set of graphical tools, and a personalization approach that enables the automatic generation of large parts of the user interface of the advisory application.

1. Introduction

In today's competitive markets, customers can choose among multiple suppliers for the desired goods whereby in many cases the differences between the products are only understandable for an expert with deep technical knowledge. In an electronic shop, customers can, for instance, choose among several hundreds of digital cameras; clients of a bank have a choice of hundreds of different funds or other forms of investments. In the classical sales channel, an experienced sales assistant will query his customer about his preferences and needs. Consequently, he will adapt the communication style in this dialogue depending on, e.g., the customer's (technical) knowledge or interests and then use his domain knowledge to propose a ranked list of suitable product alternatives [12, 1]. At the end of the dialogue, the sales person will give the customer an explanation for his recommendation and provide additional hints. In the online-buying channel, however, there has only been little support in that area for a long time and only in recent years the importance of the added-value of online sales assistance was recognized and suitable methods were developed [18], e.g., personalized recommender systems based on collaborative filtering [13, 16], data mining techniques, or expert systems based on decision trees or case-based reasoning techniques [14, 19, 3, 4].

The ADVISOR SUITE framework presented in this paper is a knowledge-based approach for building sales assistance systems in arbitrary domains that simulate the behaviour of an experienced sales person. Beside dialogue-based requirements elicitation and constraint-based product selection, the system also comprises a framework for rapid development of adaptive and personalized user interfaces that adapt themselves to the knowledge level of the current customer [1]. In order to reduce the typically high development and maintenance costs for such knowledge-intensive expert system applications, we developed a consistent set of graphical and intuitive knowledge engineering tools. Our experiences show that these tools can be used by the domain experts themselves after a short training phase which significantly reduces development and maintenance times for the applications.

The paper is organized as follows. After the presentation of the novel constraint-based approach for personalized product selection and ranking, we discuss how advisory dialogues can be modeled with ADVISOR SUITE and how this knowledge is then exploited by a generic and parameterizable user interface component. Then, we present the overall architecture of the system, make a comparison with related work in the field and end with a discussion of practical experiences of several industrial applications.

2. Recommendation Techniques

For determining the set of suitable products for the customer ("filtering"), we apply a constraint-based approach which exploits explicit knowledge about product features and customer requirements. Thus, it is possible to represent the domain knowledge of an experienced sales person in a declarative knowledge base. The structure of the knowledge base can be sketched as follows. Products are characterized by a fixed set of *attributes*, whereby these attributes take one or several values from a prespecified domain (like numbers, text, or predefined enumerations). Note, that in particular *multi-valued attributes* are common in practical settings, e.g., a digital camera can support multiple image formats. *Customer properties*

represent the current user's characteristics, skills and interests. The domain expert defines a set of *questions* (and possible answers) that he would pose in order to acquire the customer preferences. Together, product attributes and customer properties define the variables in the filtering mechanism. The expert rules for recommendations are denoted as *filter constraints*; typical examples for such rules within the financial domain are, for instance:

- If the customer is not interested in emerging markets or can only take low risks, we would propose conservative investments.
- In any case, only propose products where the monthly payments correspond to the customer's preferences plus/minus 10 percent.
- If the customer responded with "yes" to questions A and B, only recommend long-term investments.

The system's language for describing the expert rules therefore includes arithmetic and relational operators on customer properties and product attributes, text manipulation, if-then-else style constructs, as well as set operations for multi-valued attributes. The knowledge acquisition process for the expert rules is supported by a graphical user interface (see Figure 1) with online input assistance and a simplified user-oriented notation.

Besides explicitly questioning a customer about his preferences, the proposed framework supports the derivation of additional customer properties based on indirect questions. In particular, this is helpful in situations where the end-user of the system is no domain expert [1]. As an example, an investment advisor will ask several questions about the customer's financial or family situation in order to determine the suitable risk class of investments.

One of the major criticisms of filter-based approaches is that there might be situations where all of the products are filtered out by the constraints [4], which is undesirable in practical settings. Therefore, ADVISOR SUITE implements a filter-relaxation algorithm based on the Hierarchical Constraint Satisfaction [17] technique. The system evaluates the user inputs in the current state of the interaction and determines, which of the products should be proposed, i.e., which of the filter constraints have to be applied. When there are no products left or the number of remaining products does not reach a defined threshold, the system iteratively retracts filters until the desired number of products is reached. For that purpose, each of the filter constraints in the knowledge base is annotated with priority values which are typically defined by the domain expert; in many domains there are also strict rules that have to be obeyed, e.g., that one should not propose investments of high risk if the customer has no spare capital. Conversely, there might be non-strict expert rules, regarding, e.g., the major target industry sectors of an investment fund, which can potentially be ignored.

During the computation of the recommendation, the system keeps track both of the expert rules that are applied and those that were relaxed. When the result is finally presented to the customer, these rule sets are used in order to generate adequate explanations based on the natural-language annotations stored in the knowledge-base. Consequently, the explanations consist of a set of justifications like *"Given the information about your current financial situation, we recommend low-risk investments"* as well as explanations for relaxed rules like *"We also included products in the recommendation that do not match your wishes on the industry sectors of your investments."*.

As the initial priorities defined by the domain experts do not necessarily match the preferences of customers, the users can interactively apply or relax filter constraints dynamically. If one of the applied expert rules is not important for a certain user, he can instruct the recommender engine to ignore it or force the application of a particular rule vice versa which gives the user the required degrees of freedom in the recommendation process.

Ranking. After the application of the filters a personalized ranking of the products according to the customer's

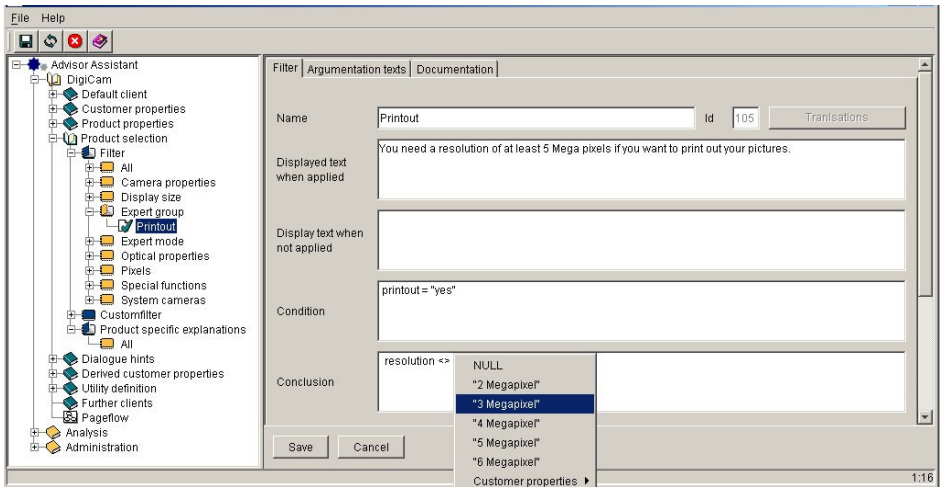

Figure 1: Graphical Knowledge Acquisition Tool

preferences is made. The initial ranking of the products is based on Multi-Attribute-Utility-Theory (MAUT) [23,1]. Similar to classical value-benefit analysis for product comparison, we can define several high-level product dimensions, like quality or economy, and define utility functions, i.e. relations from product properties to these ranking dimensions. While in classical value-benefit analysis the relative weight among the high-level dimensions is static or has to be entered manually, in the implemented MAUT approach, these relative weights are adjusted dynamically based on the customer's inputs. By using the questions about the user's preferences, the system derives the personal importance of the several product aspects for the customer and uses this data to subsequently personalize the product ranking.[1]

Nonetheless, the design of ADVISOR SUITE offers the possibility to embed external algorithms for determining the ranking based on other customer's average rating, if such information is available.

3. Personalized Dialogues

Virtual (and real-world) sales assistance and product recommendation are highly interactive processes, typically dialogues consisting of questions and answers, hints and recommendations, whereby the dialogue flow must be adapted to the customer's expertise, and preferences. Depending on the customer's answers, the system has to pose different further questions and select a suitable interaction style with the customer. A typical example is the choice of technicality of the questions, depending e.g., on the user's self assessment of his expertise.

Therefore, in ADVISOR SUITE, the way how the system interacts with the user is regarded as another important piece of domain specific knowledge of a sales expert beside the core recommendation knowledge. Consequently, the knowledge-based approach is extended with a *conceptual model* for a declarative definition of web-based sales assistance dialogues:

- A recommendation dialogue consists of a set of *pages*; each one of them contains one or more questions, where the possible answers are presented in a given layout style, for instance as a radio button.
- The dialogue can optionally be organized in *phases*, in order to give the user an overview of the progress of the recommendation session.
- Within the application there exists a set of *special* pages, like result presentation, explanations, or additional hints.
- For each page we can define in a declarative way where to proceed, i.e., which page to display next, whereby this decision depends on the user inputs. The evaluation of these conditions happens at runtime, when a *controller* page is invoked.

We intentionally used a conceptual model with a strong relation to the final web application to narrow the gap between the design model and the resulting application.

Dialogue design. Figure 2 depicts the graphical knowledge acquisition tool for defining the page flow of the sales assistance dialogue: On the left hand side, the designer of the application defines the individual phases and pages as well as the questions that have to be displayed on pages. In a detailed view, the designer can determine several parameters for each page, for instance, the style in which the question should be displayed.

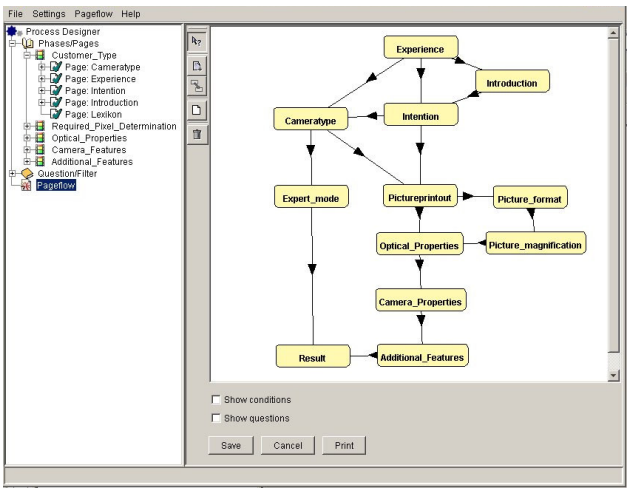

Figure 2: Modeling the interaction flow

On the right hand side, all possible paths through the interactive dialogue can be defined in a graphical way: vertices represent individual pages, edges link possible successor pages, whereby the links are annotated with transition conditions. These conditions are entered using the high-level language that is also used for the definition of filter constraints, e.g. *''Follow this link, if the user stated to have low experience.''*. When using this style of page flows that depend on direct or indirectly derived inputs, it is possible to design a personalized interaction flow that immediately reacts on the user's behaviour and adapts the dialogue to the current situation. For modeling the flow of interaction, we do not directly rely on standard techniques for modeling dynamics in user interfaces, like State-Diagrams, Petri Nets, or UML-like extensions for modeling e-Commerce applications [15, 8], because our experiences have shown that domain experts and even web-site developers have significant problems in understanding these technical concepts. Instead, we decided to use a simplified, less technical notation similar to state diagrams, which ensures that this concept is understandable for domain experts, but still has a formal semantics.

[1] Further details on using MAUT for personalized user interaction can be found, e.g., in [1].

Hints. There are situations in real sales dialogues, where the sales advisor actively interrupts the dialogue. First, there might be conflicting answers, e.g., some questions of the dialogue may be used to cross-check the plausibility of previous answers. If such a conflict arises, the sales assistant will suggest his client to reconsider his answers. Second, another point of dialogue interruption can occur when the sales assistant offers additional hints or explanations; in real-life sales conversations these interruptions are also often used for cross- or up-selling purposes.

ADVISOR SUITE allows the designer of advisory applications to model interruptions of those two kinds, i.e., hints that relate to conflicting user input as well as hints that represent additional information for the customer. The moment in time when such a hint has to be displayed during the dialogue is modeled as a condition in the graphical knowledge acquisition tool, by using either the tool's standard constraint language or explicit tables of compatible and incompatible user input combinations

In general, the ADVISOR SUITE framework supports acquisition and maintenance arbitrary text fragments, e.g. for explanations. Each of these knowledge chunks can be maintained in different personalized variants, i.e., be annotated with a condition on the customer properties. At run-time, the correct personalized version of these informative texts is automatically selected by the system. Thus, maintenance of domain-specific texts can be carried out without working at the HTML-code level.

User interface generation. In order to accelerate the development process of the graphical user interface of an sales assistance system, a framework for automatic generation of dynamic HTML pages from the declarative definitions of the interaction flow was developed. This framework follows the Model-View-Controller[2] approach which strictly separates interaction control from presentation issues and the underlying repository content.

The web-based advisory application consists of generated *Java Server Pages*[3] which are used for displaying questions, answers and informative texts and provide standard navigation which enables the customer to move freely through the dialogue. Additionally, the application includes a generic interaction module that handles user inputs (e.g. manages revisions of previously given answers), evaluates whether additional hints have to be given, and steers the interaction flow based on the definitions from the knowledge base. Furthermore, we make extensive use of *Custom Tags*[4] such that the display of questions or possible answers can be performed in HTML like style without scripting code which in turn shortens the page adaptation process.

[2] http://java.sun.com/blueprints/patterns/MVC.html.
[3] Java Server Pages, see http://java.sun.com.
[4] Java Server Pages - Tag libraries, see http://java.sun.com.

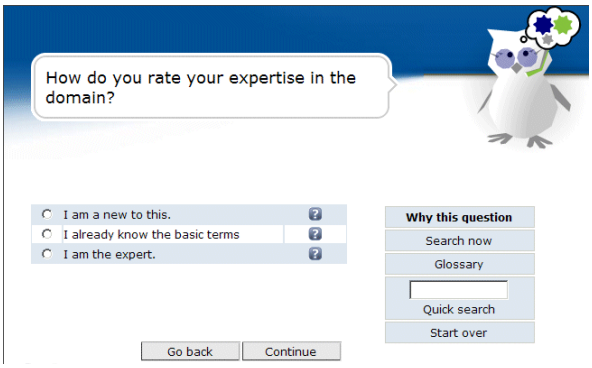

Figure 3: An automatically generated page

Figure 3 shows such a generated page that is built from predefined and easy-to-maintain templates. The problem of changes in the generated code is taken into account by the integration of an elaborated template mechanism for page generation. Additionally, hooks are integrated in the generated pages where custom code can be incorporated by the web developer, whereby this code will be unchanged in cases the pages are re-generated after maintenance activities, e.g., after a new page was introduced in the dialogue.

4. Architecture and Implementation

The overall system architecture is depicted in Figure 4. The complete knowledge for recommendation and personalization is maintained using graphical knowledge acquisition tools and stored in a central repository which is built on top of a relational database system. After generation of the user screens, the advisory application runs as application on a Web server, whereby for each customer an *Interaction Agent* manages the user interaction and performs the required personalization steps.

Integration with existing systems (like a Web-store or other enterprise applications) and extensibility are key issues for successful deployment of e-Commerce applications. Therefore, the system was built using Java-based technology as well as XML-based interfaces for data exchange. We did not rely on available expert system shells or special programming environments like *Prolog* in order to minimize the need for specialist developer knowledge. Our experiences show that this consistent use of state-of-the-art technologies significantly reduces the development and maintenance costs for expert systems with a web-based user interface. The usage of non-standard technology, for instance as a backend reasoner, would typically require programming experts and the implementation of additional interfaces between the knowledge-base, reasoner, and the user interaction components.

In the proposed system, performance issues commonly related with Java technology are addressed by ex-

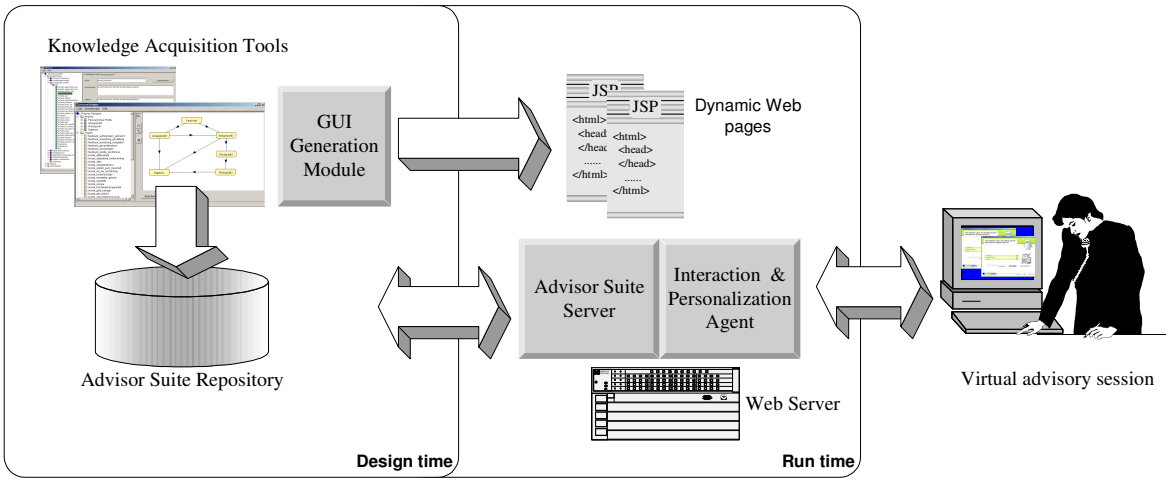

Figure 4 Overall architecture

tensive caching as well as pre-compilation of expert rules into an optimized internal representation.

5. Comparison

Recommendation technique. Over the last years, several techniques for personalized or non-personalized product recommendations have been successfully applied, whereby the most prominent examples are probably Amazon.com's[5] or CDNow's[6] online stores. From the perspective of the recommendation techniques, we can distinguish the following – often conjointly applied – approaches, compared to, e.g., [11, 5]:

- *Collaborative* These systems exploit explicit ratings on the available items made by the users. They also try to identify users that are similar to the current user and consequently extrapolate the items that are recommended.
- *Content-based* Compared to pure collaborative approaches, these systems utilize features of the items, like e.g., the genre of a book for user classification and for improving recommendation results.
- *Demographic* Beside the user ratings these systems also collect demographical and social information about the user to estimate the user's preferences [2].
- *Utility-based* They use information about item features and utility functions over the items that describe the user preferences.
- *Knowledge-based* These systems exploit explicitly acquired user needs and knowledge about feature items and about how these items match the user's preferences to infer recommendations.

All of these techniques have their specific advantages and problems. Collaborative techniques, for instance, are very well studied and have shown to be able to compute recommendations that are appreciated by the users. Moreover, this technique does not require information about the products, i.e., no knowledge engineering or maintenance activities are required. However, a typical drawback of this technique is that no good recommendations can be made for new users or new items where no ratings exist. On the other hand, knowledge-based approaches make the expert knowledge explicit and allow the recommender system to generate plausible explanations of its recommendations, whereby for these systems possibly costly knowledge-engineering processes are required for acquiring both the expert and the product knowledge.

As a consequence, hybrid approaches are used to overcome the drawbacks of individual techniques [5]. In ADVISOR SUITE, such a hybrid approach was adopted. We exploit explicit expert knowledge for inferring user preferences by using direct and indirect questions and by filtering the available items based on a constraint-based technique. The remaining items are then ranked based on the expected utility for the user. In order to reduce the costs for knowledge elicitation, significant efforts have been made to simplify this process, e.g., by defining a high-level business rules language or by providing a convenient graphical user interface.

The main reasons for this choice lie in the targeted application domains, namely complex products or services. In these domains, e.g., electronic goods like digital cameras or investment decisions, the user typically would need deep domain knowledge about the items in order to select a product that matches his real preferences and demands. Even more, recommendations in these domains require good explanations for the suggested items to increase the customer's confidence in his buying decision.

User interaction. For a long time, research on recommender systems was mostly focused on the underlying algorithms that steer the product selection and recommendation processes. However, in particular when using content-based approaches that exploit knowledge

[5] www.amazon.com
[6] www.cdnow.com

about user preferences and product properties, more complex user interaction is needed. In turn, this makes dialogue design and dialogue efficiency important topics in the field [14, 19, 22, 20, 21, 9, 1].

Dialogue efficiency is directly associated to the dialogue's length, i.e., shortening the dialogue makes it more efficient and increases user satisfaction. Recent work on recommender systems based on case-based reasoning techniques for instance aim at improving the incremental requirements elicitation process in different ways. [14] describes an approach for a product attribute selection strategy that allows the system to terminate the dialogue prematurely without loss of solution quality. [19] proposes an adaptive selection strategy that can re-focus its recommendations based on user inputs dynamically.

Another aspect of recommender systems is personalization. Following the classification from [12], ADVISOR SUITE applications both adapt the content, the structure and the presentation of the sales dialogue. Beside the computation of personalized product proposals, the knowledge-based approach for adapting the dialogue flow according to the customer's answers allows us to personalize the communication by taking the customer's knowledge and preferences into account.

From our perspective, the final goal for online sales assistance dialogues is to provide the customer a natural language conversational user interface enhanced with a virtual character with which the user can establish an emotional relation. First ideas and approaches are for instance described in [21, 9] or [22]. In our view, the complexity of natural-language interfaces still hampers the application of such systems on a broad basis because of the high costs and set-up times. Therefore, we currently rely on adaptive, form-based user interaction with the possibility of including pages and hints which makes the dialogue more natural. Further work, however, will focus on incorporating mechanisms in this direction.

Application Design and Development. The ADVISOR SUITE framework enables the domain expert to completely design the recommendation dialogue using graphical tools and to automatically generate a web-based application. This allows us to bridge the gap between requirements elicitation and the subsequent development phases. Existing model-based approaches like, for instance WebML or OO-*H*Method [6, 7, 10, 15], aim at providing general methodologies for designing web applications on a conceptual level. Compared to that work, ADVISOR SUITE is limited to web-based sales advisory applications. This allows us to use a specific, simplified modelling notation which also allows domain experts to specify the dialogue flow. In addition, the designer only has to model one single dialogue flow (steered by the users' inputs) instead of multiple flows for different contexts, e.g. for different users or user groups [7], which also improves simplicity and clarity. In contrast to general conceptual modeling approaches, the automatic generation of the recommender application in our system allows us to immediately test and use of the application, even during the prototyping phase, whereas purely conceptual approaches potentially exhibit a gap between the design model and the final web application.

6. Experiences from Practical Settings

Up to now, several instances of advisory applications built with ADVISOR SUITE system were deployed in various domains like in the financial sector, for "technical" goods like digital cameras or skis, as well as for "quality-and-taste" domains like cigars. From the development perspective we encountered that the development times for the core application are in fact very short and the basic knowledge-base could be developed in a few workshop days. Note that the initial knowledge bases were typically not created by the domain expert alone, but together with a knowledge engineer. After this phase, however, the expert was able to carry out the necessary maintenance tasks on his own requiring the engineer's help only in a few cases. Quite interestingly, the number of business rules defined by the experts is rather small, i.e., only a few dozen rules, which is a promising small number with respect to overall knowledge acquisition and maintenance costs. In this context, our experiences have shown that the modeling language and the graphical notation are easy to comprehend for the domain expert. Additionally, the automatic generation of the application is very helpful especially in the set-up phase of the knowledgebase, where the impact of changes can be tested immediately. Furthermore, the structure and the layout of the templates and the generated pages have shown to be simple enough to be adapted by a web developer in order to fit the corporate design of a company.

One of the key factors of the acceptance of advisory applications by end users lies in the quality, up-to-dateness, and completeness of the underlying product data. In most cases, a major part of the data is already available in electronic form but had to be manually enriched, e.g., with additional properties. Both the real-time access of product data in existing databases and the periodical data import via the provided XML interface ensure a satisfying quality of the data. Regarding user acceptance, we measured that in a project on one of Austria's largest e-Commerce sites with respect to daily users, over eighty percent of the sessions were successful, i.e., the customers stepped through the whole dialogue to the result page showing the proposals.

As a side effect, companies offering online advisory systems do not only profit from increased customer satisfaction by the value adding online service, but also from the new information provided by the system. Since all information about user interactions are stored and can be

evaluated online, they can learn about their customers in the sense of improved Customer Relationship Management. For instance, knowing whether one's online customers rate themselves to be experts, can serve as an aid for the company to tailor the online service to this target group. On the other hand, the evaluation of click-behaviour of the online users can help us improving the advisory application itself, e.g., we can determine when the dialogue is terminated by the user prematurely. Future work will therefore include advanced web mining techniques for automatic identification of such situations.

7. Conclusion

Product recommendation and virtual sales assistance are areas that can create significant customer benefit and improve customer relations on the online channel. We have presented a software framework for rapid development of personalized, interactive advisory systems for arbitrary domains following a content- and knowledge-based approach. Our practical experiences have shown that a knowledge-intensive approach can be successful, if adequate graphical knowledge acquisition and maintenance tools are available and the underlying concepts are presented to the domain expert using an easy-to-understand terminology. Finally the usage of a common development technology throughout the system allows us to apply standard industrial software engineering practices for the development of an expert system that has to be tightly integrated with a web-based user interface.

References

[1] L. Ardissono, A. Felfernig, G. Friedrich, D. Jannach, M. Zanker, and R. Schäfer. A framework for the development of personalized, distributed web-based configuration systems. AI Magazine, 24(3):97-110, 2003.

[2] C. Basu, H. Hirsh, and W. W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In AAAI/IAAI, pp. 714-720, 1998.

[3] R. Bergmann, R. Traphoner, R. Schmitt, P. Cunningham, B., and Smyth. Knowledge-intensive product search and customization in electronic commerce. In E-Business Applications. Springer Verlag, 2003.

[4] D. Bridge. Product recommendation systems: A new direction. In R.Weber and C. Wangenheim, editors, Procs. of the Workshop Programme at the Fourth International Conference on Case-Based Reasoning, pp. 79-86, 2001.

[5] R. Burke. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, (4):331-370, 2002.

[6] S. Ceri, P. Fraternali, and A. Bongio. Web Modeling Language (WebML): a modelling language for designing Web sites. Computer Networks 33:137-157, 2000.

[7] J. Gómez, C. Cachero and O. Pastor. Extending a Conceptual Modeling Approach to Web Application Design. In Proceedings of the 1st International Workshop on Web-Oriented Software Technology, Valencia, Spain, June 2001.

[8] M. Green. A Survey of Three Dialogue Models. In: ACM Transactions on Graphics, 5(3):244-275, 1986.

[9] T. Gurzki, P. Schweizer, and C.-T. Eberhardt. A virtual-sales-assitant architecture for e-business environments. In E-Business Applications, pp. 77-86. Springer Verlag, 2003.

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

[11] A. Jameson, J. Konstan, and J. Riedl. AI Techniques for Personalized Recommendation. Tutorial Notes. In 18th International Conference on Artificial Intelligence, Acapulco, Mexico, August 2003.

[12] A. Kobsa, J. Koenemann, and W. Pohl. Personalized hypermedia presentation techniques for improving online customer relationships. The Knowledge Engineering Review, 16(2):11-155, 2001.

[13] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. Communications of the ACM, 40(3):77-87, 1997.

[14] D. McSherry. Increasing dialogue efficiency in case-based reasoning without loss of solution quality. In 18th International Joint Conference on Artificial Intelligence, pp. 121-126, Acapulco, Mexico, August 2003. Morgan Kauffman.

[15] O. Pastor, S. Abrahao, and J. Fons. Building e-commerce applications from object-oriented conceptual models. SIGecom Exchanges, 4(2):28-36, 2001.

[16] P. Resnick and H. R. Varian. Recommender systems. Communications of the ACM, 40(3):56-58, 1997.

[17] T. Schiex, H. Fargier, and G. Verfaille. Valued constraint satisfaction problems: Hard and easy problems. In International Joint Conference on Artificial Intelligence, pp. 631-639, Montreal, Canada, 1995.

[18] J. J. B. Shafer, J. Konstan. Recommender systems in e-commerce. In ACM Conference on Electronic Commerce (EC-99), pp. 158-166, 2001.

[19] B. Smyth and L. McGinty. The power of suggestion. In 18th International Joint Conference on Artificial Intelligence, pp. 127-132, Acapulco, Mexico, August 2003. Morgan Kauffman.

[20] K. Swearingen and R. Sinha. Beyond algorithms: An HCI perspective on recommender systems. In ACM SIGIR 2001 Workshop on Recommender Systems, New Orleans, Lousiana, 2001.

[21] U. Thiel, M. Abbate, A. Paradiso, A. Stein, G. Semeraro, and F. Abbattista. Intelligent e-commerce with guiding agents based on personalized interaction tools. In E-Business Applications, pages 61-76. Springer Verlag, 2003.

[22] C. A. Thompson, M. Goeker, and P. Langley. A personalized system for conversational recommendations. In Technical Report UUCS-02-013, School of Computer Science, University of Utah, Salt Lake City, 2002.

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