Designing a metamodel-based recommender system

Sven Radde[1], Bettina Zach[2], Burkhard Freitag[1]

[1] Institute for Information Systems and Software Technology
University of Passau
D-94030 Passau, Germany
http://www.ifis.uni-passau.de/
sven.radde@uni-passau.de; burkhard.freitag@uni-passau.de
[2] :a:k:t: Informationssysteme AG
Dr.-Emil-Brichta-Straße 7
D-94036 Passau, Germany
http://www.akt-infosys.de/
bettina.zach@akt-infosys.de

Abstract. Current recommender systems have to cope with a certain reservation because they are considered to be hard to maintain and to give rather schematic advice. This paper presents an approach to increase maintainability by generating essential parts of the recommender system based on thorough metamodeling. Moreover, preferences are elicited on the basis of user needs rather than product features thus leading to a more user-oriented behavior. The metamodel-based design allows to efficiently adapt all domain-dependent parts of the system.

Key words: conversational recommender system, metamodelling, industrial application

1 Introduction

High quality assistance in complex product domains requires recommender systems to move away from strictly feature-centric recommendation approaches towards customer-oriented models. Good salespersons try to elicit the customer's needs and expectations about a product rather than asking overly technical questions. The matching of the customer's preferences with the technical attributes of a product is then left to the salesperson's experience and technical expertise. Furthermore, e-commerce activities gain more and more importance but, on the other hand, cannot be mediated by human sales experts. Particularly if the product domain is highly complex or changes frequently, however, customers expect a quality of recommendation when shopping online that is comparable to visiting a store. Hence, digital recommender systems must surpass the common simple "configurators" in functionality to increase acceptance of online sales.
The main contribution of this paper is an industrial-strength architecture for a conversational recommender system based on an elaborate customer- and

product-metamodel. The metamodel is coupled with a Bayesian inference engine which is automatically derived from an instance of the customer model. This approach allows the recommender system to classify users with respect to different stereotypes, to assess their needs and, finally, to obtain information about the most recommendable products by inferring likelihoods for the different possible technical characteristics of the matching products. Based on the inferred results, the recommender system adapts its dialogue dynamically and obtains product recommendations by querying a ranking-enabled database that contains the product catalogue.
The solution presented in this paper is the result of a joint R&D effort with contributions from both science and business. This ensures that the developed metamodel closely resembles the natural recommendation procedures in the target market of mobile telecommunications. In addition, we found that a wide range of business domains share the basic assumptions of our metamodel, thus enabling the re-use of our approach by simply re-instantiating the metamodel. The latter effectively hides the mathematical complexity of the backend and therefore allows for a less involved, intuitive maintenance process. Business people have often argued that this could be a crucial point for recommender systems, particulary in fast changing domains.
In summary, we present:

- a metamodel-based approach for the efficient maintenance of a recommender system;
- an innovative industrial application of recommendation technology based on this approach;
- some experiences from a real-life implementation of the described principles;

The rest of the paper is organized as follows: In section 2 we describe our use case in the mobile telecommunications domain, identifying the need for support by recommender systems. Section 3 introduces the metamodel, while section 4 gives an overview of the proposed architecture. We detail experiences from our prototypical implementation in section 5, with particular attention on question selection, generation of recommendations and model maintenance. In section 6 we review some related work, before concluding with an outlook in section 7.

2 Use Case and Market Situation

Today's market for mobile phones is characterized by a huge number of choices. It is not uncommon to see more than 100 different items in shops, many of them differing only in details. The technology itself progresses rather fast, with permanently increasing computing power, new hardware options like GPS receivers, or integration of online services. Customers need qualified consultation to match their (often vague) preferences with these complex product models. However, when visiting the web sites of major mobile service providers, one finds most of them lacking sophisticated recommendation functionality. Commonly, these websites merely allow customers to restrict the catalogue of available cellphones by specifying purely technical constraints.

The course of action of specialized dealers is notably different: Initially the customer is classified according to a set of broad stereotypes, such as "young", "business customer", or the anticipated amount of money he or she would be willing to spend. This classification determines the broad course of the further dialogue. Based on their professional experience, sales people then try to assess the customer's needs and expectations which are finally mapped onto suggestions for technical features and, consequently, available products. We argue that emphasising *soft criteria* like "elegant phone" is one of the key differences between a natural sales dialogue and the common technology-oriented online phone configurators.
In addition to its complexity, the product domain changes frequently and often radically. Updates of the product domain may stem from a temporary marketing campaign, the introduction of new mobile phone variants, or technical innovations. In particular the latter often requires significant adjustments to a recommendation dialogue to accommodate previously unknown functionality. It is commonly not sufficient to change some parameters and therefore necessary to involve programmers thus leading to correspondingly high delays and costs.
Also, the constant requirement to train shop assistants for these adaptations or to modify online recommender applications, however, is counteracted by the need to cut costs, because average monthly revenues per customer decline, forcing retailers to deal with smaller profit margins. In this context, particularly when looking at retail chains with their even smaller margins, providing salespersons with a suitably powerful electronic recommender system is a way to keep costs low. For retail chains and e-commerce use in general, such a system will allow for a high quality recommendation process, while independent retailers and specialized dealers will be able to keep up their high standards of recommendation quality with a significantly reduced training overhead.

3 Metamodel Design

The use-case described in the previous section has been investigated in a joint R&D project conducted in cooperation of academic researchers and an industry partner having long-term expertise in the field. One of the primary goals is to achieve a separation between domain-dependent parts of the system which have to be adapted to changes on the one hand and a domain-neutral core that can remain stable even in face of significant changes within the domain on the other hand. To this end, we propose a domain metamodel basically expressing our assumption that stereotypes, needs and technical attributes form a network of mutual influences (see Fig. 1 for a UML representation). Instances of the metamodel must be created by a suitably experienced domain expert.
For presentation reasons, the product model has been simplified by not including those parts that enable the representation of "bundled" articles (i.e., the combination of two distinct articles to form the eventually sold item as is the case, e.g., with mobile phones and the corresponding contracts). It is noteworthy that the presented metamodel is not capable to capture configurable products

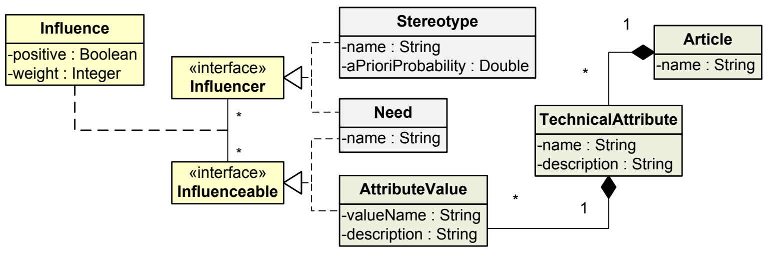


Fig. 1. UML diagram of the domain metamodel

with implicit relations (e.g., "Chosen AC adapter must be able to power all other components.") without further ado. Explicit relations, such as "allowed bundles", can be modelled and used to construct valid recommendations.
First of all, *Stereotypes* are broad categories used to classify customers into classes that are considered representative for the particular application domain. They can be seen as a set of labels, a subset of which applies to any particular customer, forming his/her "composite" stereotype. *Stereotypes* have an *a priori probability* assigned that models statistical demographic knowledge about the prospective customers. Whenever the recommender system does not have positive knowledge about a *Stereotype* (e.g., the customer refuses to answer a corresponding question), it may use the modelled probability (in order to avoid "stalling" the dialogue). Obviously, this modelling is ideally suited to our use of Bayesian networks (cf. section 4), but other formalisms may also choose to exploit this information appropriately. Some *Stereotypes* will be independent of each other, while others may exclude or imply another *Stereotype*.

Example 1. As a brief example within our telecommunications domain, assume that the following *Stereotypes* are defined: "Age below 21", "Age between 21 and 55", "Age greater than 55", "Business customer" and "Fun-oriented user". Obviously, a customer would be assigned one of the "Age" *Stereotypes* and one, both or neither of the other two *Stereotypes*.

The customer's composite stereotype influences the *Needs* that he or she is likely to have. As opposed to *Stereotypes*, having or not having certain *Needs* is rarely a purely boolean decision. Rather, customers are able to choose from a set of values ranging from "Not at all." to "Yes, absolutely." when answering inquiries about *Needs*. *Influences* between *Needs* and *Stereotypes* represent the fact that being of a certain *Stereotype* changes the likelihood of having certain *Needs*.
Accordingly, *Influences* have a type (positive/negative) indicating the kind, i.e., strengthening or weakening, of their effect and a numerical weight expressing their relative importance. The semantic of *Influences* is dependent on their type: Positive *Influences* increase the likelihood that a customer will feel a *Need* when he or she belongs to the corresponding *Stereotype* and decrease it when this is not the case. Negative *Influences* have the opposite meaning. It is worth noting that a *Need* is both an *Influencer* and an *Influenceable* in our metamodel, thus enabling "recursion" to model more complex hierarchies of *Needs*.

Stereotype	Need	Type	Weight
Age below 21	Multimedia use	positive	1
Age between 21 and 55	Office use	positive	1
Business customer	Office use	positive	3
Fun-oriented customer	Multimedia use	positive	2
Age below 21	Office use	negative	1

Table 1. Representative *Influences* for example 2

Example 2. Extending the set of *Stereotypes* from example 1, assume that a domain expert defines the *Needs* "Office use" and "Multimedia use". To integrate them with the *Stereotypes*, some representative *Influences* as shown in Table 1 are defined.

In the domain metamodel, an *Article* is seen as having a number of *Technical-Attributes* which, in turn, have discrete ranges of possible *AttributeValues*, modelled as a hierarchy of compositions in Fig. 1. Often, a *TechnicalAttribute* will have a boolean value range, indicating the presence or absence of a certain property. Domain experts may also use this feature to extend technical information with subjective annotations, such as "fancy casing", "trendy" or similar.
It is important to establish the notion that the product part of the domain model for our sample domain does not contain information about concrete products, i.e., concrete cellphones. Rather, it abstracts from this point of view by providing a more general description of how "cellphones as a whole" may look like by enumerating the characteristics of their technical properties. Finally, product catalogues or databases can be seen as instances of our domain model, i.e., they form the domain itself. The concrete cellphones available for sale are represented at this level, described by the terms defined in the domain model.
We note that restricting our domain metamodel to cover only discrete value ranges is not a significant loss: Even attributes that would normally be considered to have continuous value ranges often follow quite regular patterns, e.g., "price" is usually defined in terms of discrete "price bands" anyway. If there are too many discrete values, one can define buckets and classify the actual values accordingly.
Again, *Influences* link *AttributeValues* to *Stereotypes* and *Needs*, based on the notion that, given a certain *TechnicalAttribute*, some of its *AttributeValues* are more likely so satisfy the customer's *Needs* than others. See Fig. 2 for a simplified instance of the domain metamodel, extending examples 1 and 2.
While aiming primarily at the telecommunications market, the underlying assumptions of the metamodel apply to a wide range of similarly structured business domains. A technology prototype of our system was used in a completely different context where it was used to recommend a program of study to potential beginners, thus demonstrating the general adaptability of the metamodel.

4 Preference Elicitation and Domain Model Usage

All domain-dependent parts of the recommender system described in this paper can be generated from instances of the metamodel shown in section 3 and are

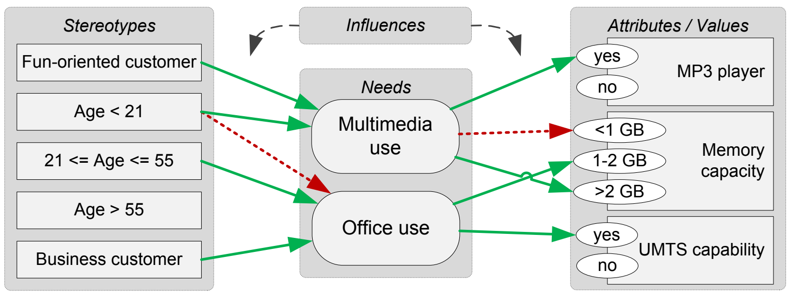


Fig. 2. Exemplary domain model. Solid lines denote positive *Influences*, dotted lines negative ones.

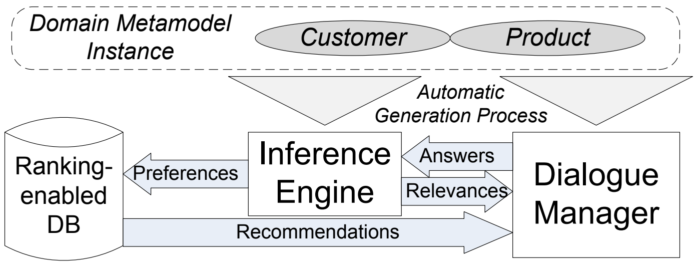


Fig. 3. System architecture overview

combined with a domain-independent system core. In [1], we presented a dialogue structuring method, which acts as the underlying controller component for the course of the dialogue. [2] presents the Bayesian inference engine and particularly its generation-method in more detail. Fig. 3 illustrates how these components are integrated with a ranking-enabled database [3] and the metamodel instance for a given domain to form our recommender system architecture.
As illustrated by Fig. 3, the recommendation dialogue is generated automatically from the current domain model after a maintenance iteration has been finished. The dialogue itself consists of a series of steps, each of which contains a variable number of questions. The generation step creates one question for each stereotype and need defined in the domain model. Another set of questions to cover the technical attributes of the domain model is generated as well, so that the dialogue may transition to this fine-grained level of detail if necessary.
A question is designed in a way that its answer expresses the customer's preferences with respect to the corresponding model element, e.g., the need "Office use". The *Dialogue Manager* component, which uses statecharts (see [4, 5]) as its internal dialogue structuring model, is responsible for the selection of the most appropriate question and its presentation.
The initial strategy of the dialogue is to attempt to obtain evidences for the stereotype nodes by asking the corresponding questions first, thus providing a quick personalization of the network's state. Afterwards, the customer's needs are elicited, leading to further individualization. Answers are passed to the *Inference Engine*, which treats them as evidences for its internal Bayesian network (see,

Domain metamodel element	Generated element(s)
Stereotype	Question & Boolean node in the Bayes net
Need	Question & "Graded" node in the Bayes net
Technical attribute	Question
Attribute value	Boolean node in the Bayes net
Influence	Edge in the Bayes net & Defines the CPTs

Table 2. Generated system elements

e.g., chapter 14 of [6]). The latter is also based on the domain model and is constructed during the generation step.

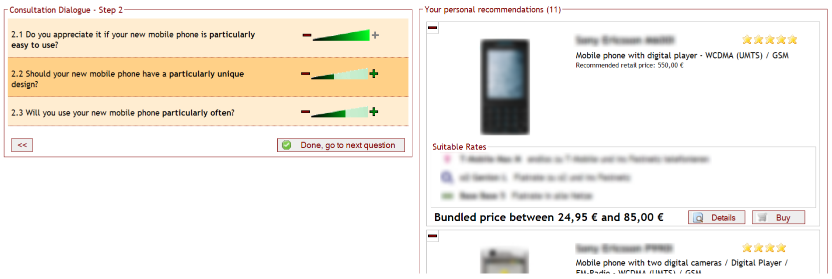
Table 2 gives a summary of the system elements that are generated from the domain model. Stereotypes are represented as chance nodes with a boolean value range (and a-priori probabilities are set according to the domain model), whereas needs are modeled as chance nodes with a value range to reflect the possible "graded" answers mentioned in section 3. Product attributes are included in the network by creating one boolean node for each element of their respective ranges of attribute values. Finally, the modeled influences are represented as edges between the corresponding nodes. The conditional probability tables (CPTs) of the influenced nodes are designed to reflect the causal effects implied by the influences of the domain model (cf. [2] for details): 1) Belonging to certain stereotypes implies that some needs are more likely than others; 2) Having a certain need implies that a product with appropriate technical properties will be more useful than another.

As discussed in subsection 5.1 below, the *Inference Engine* computes estimations of the importance of each question which the *Dialogue Manager*, in turn, uses to select the most relevant question to ask next. By tracking the causal influences mentioned above, the *Inference Engine* also provides estimations of the usefulness of technical attributes. Based on the results, a personalized multi-attribute utility function is constructed which is then used by the *Ranking Database* to sort the products in the catalogue and to determine the current, personalized recommendation. Conceptionally, the whole product catalogue is ranked, but Top-K operators may be used to limit the size of the recommendation (e.g., to match the capabilities of the user interfaces at hand).

To summarize, a complete dialogue step contains the following actions:

1. Present the current questions to the customer and receive answers;
2. Update the evidences in the Bayesian network according to the answers;
3. Calculate the new posteriori probability distributions;
4. Construct utility function and execute ranking query to create new recommendations;
5. Determine next questions based on predicted answers;

Dialogue steps are iterated until the customer decides to buy one of the recommended products or the system runs out of questions. For the latter case, questions about technical attributes are asked which should quickly narrow down the available choices and therefore eventually lead to the purchasing decision. In addition, customers are *always* able to supply technical requirements which are taken into account for ranking purposes. Alternatively these requirements may


Consultation Dialogue - Step 2

2.1 Do you appreciate it if your new mobile phone is **particularly easy to use?**

2.2 Should your new mobile phone have a **particularly unique design?**

2.3 Will you use your new mobile phone **particularly often?**

Your personal recommendations (11)

Mobile phone with digital player - WCDMA (UMTS) / GSM
Recommended retail price: 550,00 €

Suitable Rates

Bundled price between 24,95 € and 85,00 €

Mobile phone with two digital cameras / Digital Player /

Fig. 4. Application screenshots showing questions (*left*) and recommendations (*right*)

be treated as *hard* constraints. Having hard constraints in the system, however, leads to the possibility of empty recommendations and therefore may require constraint relaxation, whereas representing technical requirements as soft constraints implicitly ensures that always at least one recommendation is available. Hence, hard constraints are only enabled at the explicit request of the customer. The current recommendation is displayed in parallel to the questions of the current dialogue step and the customer is of course free to end the dialogue at any time (either by buying or by leaving the website).

5 Experiences

A recommender system based on the approach described above has been implemented as an ASP.net web application (cf. Fig. 4). Our prototype exhibited reasonable performance and will be integrated into an established order management system. The system has been demonstrated to selected customers and will continue to be presented to interested parties. We have already gathered valuable feedback from these initial evaluations. In addition, we are planning to conduct a large-scale field-test to obtain statistical results about the quality of recommendations and intuitiveness of dialogue management.

5.1 Selecting the next question

The dialogue manager selects questions based on the probabilities that the Bayesian network predicts for the possible answers (cf. [2]). In our current prototype, the most relevant questions are those with particularly confident predictions. The basic assumption is that the customer has a clear opinion about these questions and that, because of this, they should be asked first. Also, our prototype can display "empathic" behaviour by pre-selecting the most probable answer option for these questions in the GUI. Our experiences with the prototype show that customers generally agree that the questions shown this way are very relevant for the recommendation process and that the preselected answers resembled more or less exactly the choice the user would have made.
However, the approach also exhibited a few shortcomings during our tests. Due to the described way of choosing questions, the early phases of the dialogue mostly

consist of confirming the predictions of the inference engine predictions. As a consequence, the Bayesian network does not acquire much "new" knowledge quickly, meaning that its calculated a posteriori probabilities do not change much. As the latter are also used to obtain the recommended products, this leads to the following observed behaviour:

1. Defining the stereotypes at the beginning of the dialogue has a significant influence on the generated recommendations;
2. Confirming the confidently predicted questions during the early dialogue does not change the recommendations significantly;
3. Only later on, when other questions are answered, do the recommendations stabilize towards the eventually preferred products;

To improve this behaviour, the dialogue manager will be modified to prefer exactly those questions that have a particularly uncertain answer prediction. Initial evaluations show that this leads to more significant early changes within the Bayesian network that tend to stabilize later on. In other words, the Bayesian network obtains the most important knowledge earlier and only confirms its predictions afterwards. We will enter our field-test with both variants to investigate the user-friendliness of the approaches but we expect that the tendency towards shorter, more "efficient" dialogues will prove superior to our previous strategy.
Also, it is desirable to exploit possible semantic connections between individual needs and their corresponding questions. A question might, for instance, be a logical successor of another, or answering a given question in a certain way may mean that some other questions are useless to ask. Therefore, the dialogue manager will be extended to infer these relationships from the structure of the Bayesian network as far as possible. The goal is to form sensible chains of dialogue steps without having to extend the metamodel with capabilities to explicitly define "cliques" of questions.

5.2 Recommendations

During our evaluations we observed that users commonly assign a higher subjective importance to their most recent answers. Consequently, products satisfying the corresponding needs were expected to be displayed as a recommendation in the next dialogue step. Recommendations were regarded as incomplete if the top products displayed on screen do not reflect these answers distinctly.
In contrast, our ranking method always uses all available preferences and does not specifically consider the "age" of any given answer. This means that in some cases the most recent questions do not have an obvious effect on the displayed recommendations. Researching measures to extend the ranking mechanism by a notion of "temporal" importance of questions is part of our ongoing work.

5.3 Model Maintenance

To qualify for being integrated into the industrial sales/distribution processes, any approach must minimize the required maintenance overheads. The mere existence of a model already reduces maintenance efforts for the most common case:

Adding new products is fully transparent, as long as they can be described in terms of the established domain model. Conventional approaches would require adaptation to integrate the new products. The product model itself needs adaptation only if technical innovations cause the addition of new *TechnicalAttributes* or an extension of the value ranges of existing attributes.
Considering the maintainability of the customer model, the *Stereotypes* exhibited reasonable stability over time, and, to a certain extent, even across different business domains. This is obviously due to the fact that they represent sociodemographic knowledge about the customers. Domain experts primarily have to define the *Needs* and *Influences* of the domain model. As the *Influences* are defined with respect to *AttributeValues* of the domain model (as compared to concrete products), these associations remain stable when new products are added. Once changes have been applied to the model, the system backend is re-generated automatically (see section 4), completely hiding the mathematical and informatical complexities from the model designer.
In addition, integrating the maintenance of the domain model into the existing business processes of catalogue creation and maintenance provides potential for even further optimizations. As the model contains a significant amount of marketing knowledge, it can be re-used, e.g., to simplify the creation of printed marketing materials by automatically annotating products with information about which needs they satisfy particularly well.

6 Related Work

Our recommendation approach is notably different from collaborative filtering methods (cf., e.g., [7, 8]) by not requiring item ratings. Assuming that the same user will not interact with the system very frequently (typically, in our domain, a user will buy a single new cellphone about every two years), we cannot rely on buying histories to build our model of the user. While eliciting explicit ratings may be acceptable in an online context, it is not an adequate form of interaction between salespersons and customers. Hence, our user-model builds on information that can be elicited during the course of a natural sales dialogue.
Ardissono et al. [9, 10] present a personalized recommender system for configurable products. Their approach involves preference elicitation techniques that employ reasoning about customer profiles to tailor the dialogue to a particular customer by providing explanations and smartly chosen default values wherever possible. The customer preferences learned this way are then used as constraints in the configuration problem at hand to generate the recommended product configuration, which might result in empty recommendations (i.e., the specified constraints are not satisfyable), requiring repair actions. Our approach does not directly exploit the elicited preferences as constraints but rather uses them as an input to ranking database queries which return a list of product recommendations ordered according to the customer's preferences. To suggest personalized default answers to questions, our approach does not need to rely on a set of business rules as appears to be the case in [9].

In [11], the dialogue-oriented recommender suite CWAdvisor is presented. Their knowledge-base is similar to ours but it includes an explicit representation of the "recommender process definition", i.e. all possible dialogue paths in a tree-like structure. While obviously able to specify a fine-grained dialogue, the achievable level of detail is limited by the complexity of the dialogue specification. Our approach generates the (equally complex) dialogue specification from a much more compact model and is more flexible by incorporating mixed-initiative selection of questions, easy belief revision and adaptive personalization.
An adaptive approach to select technical questions is presented in [12] that is able to suggest "tightenings" (i.e. further questions) to reduce recommendation size based on a previously learned probability model. Their approach includes the possibility to re-learn the model when more dialogue histories are available. In contrast, our approach does not include a learning step but delegates that task to a domain expert. Also, our model is not concerned with direct connections between dialogue elements as is the case in [12] but rather specifies a more abstract view on the product domain from which the dialogue structure is inferred.
An approach similar to ours is presented in [13]. However, the utility estimations (the "value tree") of Jameson et al. do not seem to be built on an explicit model of the currently served customer but rather on an average user of their system. Hence, the recommendations are not personalized as strongly as in our approach which allows an adaption even to atypical customers by setting the appropriate stereotypes. Also, as the value tree is a strictly hierarchical structure, it cannot capture the fact that a technical attribute may be influenced by more than a single need. Furthermore, it is not completely clear how informal statements (e.g., "I am a law student.") can be interpreted as relevant knowledge (e.g., an increased interest in politics) by the system apart from the possibility that a domain expert models this association directly within the Bayesian network.
A domain model based on dynamic logic programming was introduced by Leite and Babini in [14]. Both customer and user model are represented using a massive set of declarative rules which allows a detailed and powerful specification of the business domain – possibly even extended by user-supplied rules. However, the complex formal models appear expensive to maintain when confronted with domain changes. Furthermore, it seems unlikely that domain experts, much less customers, are able to express their knowledge by logic rules, whereas intuitiveness and maintainability of the model are two key points of our approach.

7 Conclusion

This paper presents an approach for a dialogue-based recommender system, taking industrial maintainability requirements into consideration. The focus of our prototypical implementation is the telecommunications market, but the system has been designed with domain-independence in mind and therefore establishes a strict separation of domain-dependent and domain-independent parts. This is achieved by using a domain metamodel explicitly designed to allow efficient maintenance. All domain-specific parts of the system are generated from in-

stances of the metamodel. The overall architecture comprises a dynamic dialogue management, a preference elicitation component based on Bayesian networks, and a ranking-based retrieval from the product database.
The evaluation of a prototypical implementation proved the general usability of our approach and also yielded valuable lessons for further technical improvements: Determining questions' relevances and aligning the recommendations with the customers' expectations are the crucial points for our ongoing research. We will refine our approach based on the first evaluation and a thorough field-test relating the computed outcome of the recommender system with the experience and expectations of users and sales experts.

References

1. Radde, S., Beck, M., Freitag, B.: Generating recommendation dialogues from product models. In: Proc. of the AAAI-07 Workshop on Recommender Systems in E-Commerce, AAAI Press (2007)
2. Radde, S., Kaiser, A., Freitag, B.: A model-based customer inference engine. In: Proc. of the ECAI-08 Workshop on Recommender Systems. (2008)
3. Beck, M., Freitag, B.: Weighted boolean conditions for ranking. In: Proc. of the IEEE 24th International Conference on Data Engineering (ICDE-08) – 2nd International Workshop on Ranking in Databases (DBRank08). (2008)
4. Harel, D.: Statecharts: A Visual Formalism for Complex Systems. Science of Computer Programming **8**(3) (1987) 231–274
5. Wieringa, R.: Design Methods for Reactive Systems. Morgan Kaufmann (2003)
6. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall International Editions (1995)
7. Jin, R., Si, L., Zhang, C.: A study of mixture models for collaborative filtering. Information Retrieval **9**(3) (2006) 357–382
8. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Trans. on Information Systems **22**(1) (2004) 5–53
9. Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Meyer, M., Petrone, G., Schaefer, R., Schuetz, W., Zanker, M.: Personalizing online configuration of products and services. In: Proc. of the 15th European Conference on Artificial Intelligence (ECAI-02). (2002)
10. Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schaefer, R., Zanker, M.: A framework for the development of personalized, distributed web-based configuration systems. AI Magazine **24**(3) (2003) 93–110
11. Felfernig, A., Friedrich, G., Jannach, D., Zanker, M.: An integrated environment for the development of knowledge-based recommender applications. Intl. Journal of Electronic Commerce **11**(2) (2006) 11–34
12. Mahmood, T., Ricci, F.: Learning and adaptivity in interactive recommender systems. In: Proc. of the 9th Intl. Conference on Electronic Commerce, ACM (2007)
13. Jameson, A., Schaefer, R., Simons, J., Weis, T.: Adaptive provision of evaluation-oriented information: Tasks and techniques. In: Proc. of the 14th Intl. Joint Conference on Artificial Intelligence (IJCAI-95). (1995)
14. Leite, J., Babini, M.: Dynamic knowledge based user modeling for recommender systems. In: Proc. of the ECAI-06 Workshop on Recommender Systems. (2006)