

An Integrated Environment for the Development of Knowledge-Based Recommender Applications

Abstract. The complexity of product assortments offered by online selling platforms makes the selection of appropriate items a challenging task. Customers can differ significantly in their expertise and level of knowledge regarding such product assortments. Consequently, intelligent recommender systems are required which provide personalized dialogues supporting the customer in the product selection process. In this paper we present the domain-independent, knowledge-based recommender environment CWAdvisor which assists users by guaranteeing the consistency and appropriateness of solutions, by identifying additional selling opportunities, and by providing explanations for solutions. Using examples from different application domains, we show how model-based diagnosis, personalization, and intuitive knowledge acquisition techniques support the effective implementation of customer-oriented sales dialogues. In this context, we report our experiences gained in industrial projects and present an evaluation of successfully deployed recommender applications.

Keywords: *Recommender Systems, Personalization, Knowledge Acquisition, End-user Programming, Knowledge-based Recommenders, Testing Recommender Knowledge Bases, Model-based Diagnosis.*

1 Introduction

The selection of products from a complex assortment such as financial services or digital cameras is still a challenging task for customers interacting with online selling environments. In many cases, those environments only offer simple query interfaces based on the assumption that customers know technical product details. Recommender technologies [1] [29] [34] [36] [37] improve this situation by generating product proposals which are automatically derived from a set of given customer requirements. The three most well known approaches to the implementation of recommender applications are the following: *collaborative filtering* [20] [33] relies on product preferences of a large set of customers. Recommendations are derived from preferences of a group of customers with similar purchasing patterns. Consequently, no deep knowledge about the product domain is needed in this context. When using *content-based filtering* [30],

recommendations are based on similarities between product descriptions and the preferences of the current customer. When a customer interacts with the recommender application, products are proposed that are similar to those the customer has liked in the past. Finally, *knowledge-based recommender applications (advisors)* [4] [5] [13] [24] [39] exploit deep knowledge about the product domain in order to determine recommendations. When selling, for instance, financial services, recommendations must adhere to legal regulations and suit the customers' financial restrictions as well as their needs and wishes. Compared to customers purchasing simple products such as books or compact disks, customers purchasing financial services are much more in the need of information and in the need of intelligent interaction mechanisms which support the selection of appropriate solutions. Therefore, an *explicit* representation of product, marketing and sales knowledge [13] is needed. Such deep knowledge allows us (a) to calculate solutions which adhere to legal regulations, which are in the line with a company's marketing and sales strategy, and which suit the requirements of the customer, (b) to explain solutions to a customer, and (c) to support customers in situations in which no solution can be found. Primarily, knowledge-based recommender technologies can provide the formalisms that are needed in this context.

In this paper, we present the knowledge-based recommender environment CWAdvisor. The major innovations of this environment compared to other knowledge-based approaches [4] [5] [24] [39] are the following:

- *Graphical knowledge acquisition.* CWAdvisor provides a graphical development and test environment for knowledge-based recommender applications [13] [14]. This environment supports the design of knowledge bases (product, marketing and sales knowledge) and process definitions (intended behavior of the recommender user interface) on a graphical level (see Section 2.2). Such an approach allows rapid prototyping processes by automatically translating graphical models into an executable application.

- *Relaxations of filter constraints and repairs of inconsistent customer requirements.* Model-based diagnosis techniques [11] [32] can be used to actively support customers in situations where no solution could be found (see Section 3.1): either by identifying minimal relaxations of filter constraints or by the determination of repair actions for customer requirements. Both can guarantee the identification of at least one solution.
- *Personalization of sales dialogues and recommendation results.* Recommender dialogues in CWAdvisor are based on a finite state model [14] which describes possible interaction sequences of a recommender application on a graphical level. Using such representations, the formulation of questions, answer alternatives, and explanations can be automatically adapted to the domain knowledge level and preferences of a customer. Furthermore, recommendations can be ranked using concepts of multi-attribute object rating [2].

CWAdvisor has been successfully applied in domains such as financial services or electrical equipment. All these applications are based on one of the following basic scenarios:

- *Consumer/Customer support.* Similar to traditional sales channels, improved sales assistance generates added value for online customers. On the one hand, knowledge-based advisors allow an intuitive access to products for customers. On the other hand, sales representatives are relieved from routine advisory tasks which are taken over by recommender applications.
- *Support of sales representatives.* Sales representatives interact with advisors when preparing a sales dialogue or when talking to a customer. In this context, they are supported by guided dialogues which consist of questions and explanations focusing on the customer's needs and wishes. Consequently, customer-oriented sales dialogues are supported in this context.

Within the scope of these scenarios, we present *two example showcases* (see Section 4) which clearly demonstrate the utility of our approach. First, *a digital camera advisor* has been deployed for the largest Austrian online price comparison platform. Effective preference elicitation is a

challenge in this context since customers can differ significantly in their product domain knowledge and in their expectations. Second, financial service advisors have been deployed for an Austrian financial service provider. *Financial service advisory* is a knowledge-intensive task which in many cases overwhelms customers as well as sales representatives. Since the product assortment is quite manifold (investment, financing, pension, life insurance, etc.), many sales representatives focus on selling only a restricted set of products which leads to sub-optimal offers. The goal here is to identify a solution which suits the needs, wishes, and financial restrictions of the customer and which is in the line with the sales strategy of the company.

The remainder of this paper is organized as follows. In Section 2 we present the CWAdvisor environment. In Section 3 we give an overview of Artificial Intelligence (AI) technologies which ease the development of and improve the interaction with knowledge-based recommender applications. Section 4 presents an evaluation of successfully deployed recommender applications. Thus, we provide an overview of the CWAdvisor environment which is the result of long-term research in the areas of knowledge-based configuration [2] [12] [17], model-based diagnosis [11] [14] and knowledge-based recommender systems [13] [14] [22]. We focus our discussions (a) on basic principles of testing knowledge-based recommender applications, (b) on mechanisms supporting users in situations where no solution can be found, and (c) on experiences gained from commercial recommender application development projects.

2 CWAdvisor Environment

CWAdvisor technologies can be used for a variety of tasks: the *formalization of product, marketing and sales knowledge* by non-programmers; the *test of recommender knowledge bases*; the *mapping of customer requirements to a set of product properties*; the *repair of inconsistent customer requirements*, and the *explanation of solutions* in order to increase a customer's trust.

The major reasons for the application of CWAdvisor technologies in the *financial services* domain are the following:

- Solutions must be objective, correct, and explainable. This requirement makes approaches such as collaborative or content-based filtering not the best choice since, explanations in the form of repair proposals, for example, rely on the existence of a knowledge base.
- Typically, financial service providers want to develop and test recommender applications autonomously. Therefore, knowledge representations are needed which allow both an effective knowledge base design and the validation of calculated results by domain experts.
- Financial services recommendation is a complex task with a large number of constraints and possible solutions [13]. In such scenarios, a knowledge-based approach can significantly reduce the development efforts for recommender applications [16].

Similar reasons motivate the application of CWAdvisor technologies in other domains such as *digital cameras* where the following additional aspects have to be taken into account:

- In the consumer electronics market we observe a fast technological progress. Knowledge bases have to be updated on a regular basis when new features have to be considered in the recommendation process, or when increased performance characteristics become standard. Such change requests related to the knowledge base happen very frequently and must be fulfilled fast and without extensive programming efforts.
- Personal preferences that cannot be captured in advance play an important role in the domain of digital cameras. In this context, recommender applications should allow customers to experiment with decision alternatives (undoing choices, sorting proposals by different criteria, changing the priority of preferences, etc.).

2.1 CWAdvisor Architecture

The overall CWAdvisor architecture is shown in Figure 1. This architecture consists of two major parts: the *development and test environment*, which supports the graphical design of recommender applications, and the *runtime environment*, which is responsible for controlling the execution of recommendation processes. CWAdvisor knowledge bases and process definitions are developed and maintained using the development and test environment (*CWAdvisor Designer*, *Process Designer*, and *Test Designer*) and are stored in an underlying relational database (*Repository*). Product structures and instances are defined within CWAdvisor Designer or imported from external systems using an XML interface. CWAdvisor Server supports the execution of advisory processes. Based on given user inputs, the server determines and executes the personalized dialogue flow, initiates result computations and retrieves explanations, such as explanations as to why a certain product suits the needs and wishes of the customer.

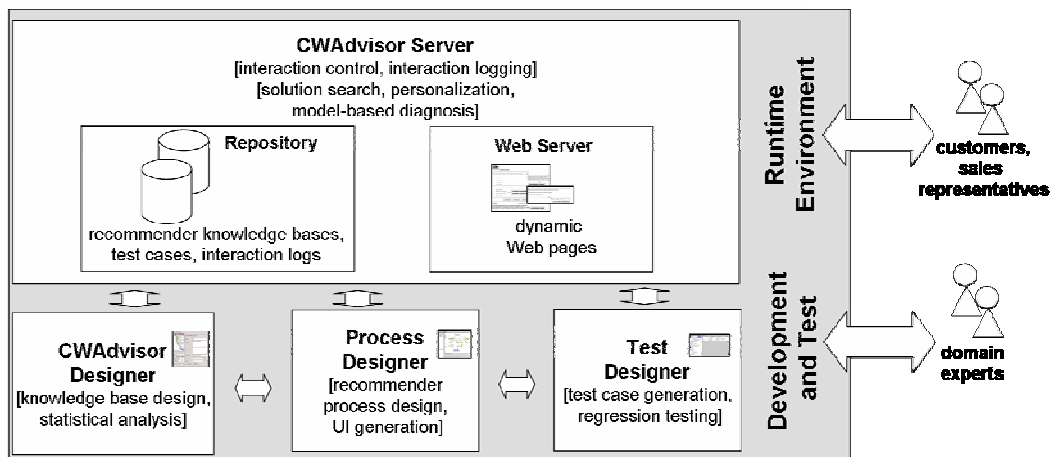


Figure 1: CWAdvisor architecture.

2.2 CWAdvisor Designer and Process Designer

CWAdvisor Designer is a graphical development environment for knowledge-based recommender applications (see Figure 2). It is based on Java Web Start technology which provides an environment for deploying Java-based applications on a Web server and executing

those applications on a corresponding client. Using CWAdvisor Designer, the relevant set of product and customer properties and constraints is identified and transformed into the formal representation of a *recommender knowledge base* [12] [13].

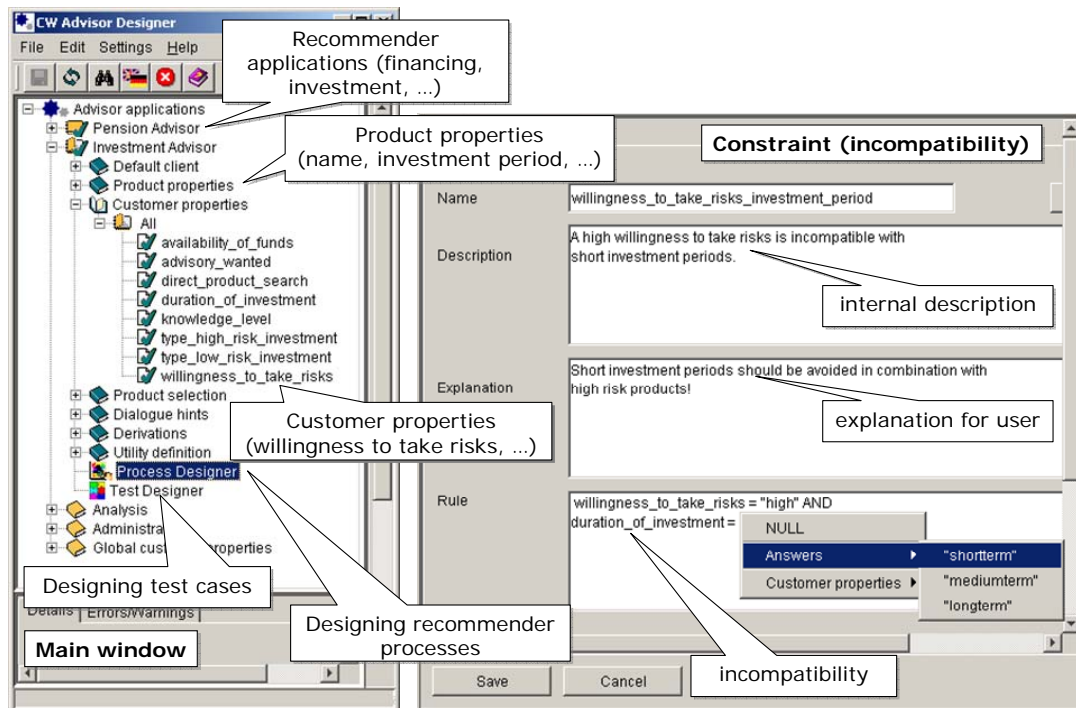


Figure 2: Definition of customer properties and constraints in CWAdvisor Designer.

A *recommender knowledge base* consists of the following parts:

1. *Customer properties* describe possible customer requirements. Customer requirements are collected by posing questions to the customer. The question *under the assumption that your investment of 10,000 EURO decreases in value, at which value would you sell your investment?* is related to the customer property *willingness to take risks* ('low', 'medium', 'high'). Examples from the digital camera domain are *preferred motif* ('landscapes', 'portraits', 'sport scenes') or *maximum price*.
2. *Product properties* describe possible product instances. Life insurances, for instance, can be characterized by the *possible length of life insurance policies*, *premiums of life insurance*

policies, or *product type*. Furthermore, digital cameras can be described by their *resolution*, *brand name*, or *weight of the camera*.

3. *Compatibility constraints* are restricting possible combinations of customer requirements: *return rates above 9% require the willingness to take risks* or *high resolution cameras cannot be provided in a low-price segment*. Furthermore, *filter constraints* define the relationship between customer properties and product properties: *customers without experiences in the financial services domain should not receive recommendations which include high risk products*. It is important to note that compatibility as well as filter constraints can also be represented on a tabular level.

Apart from the description of the offered products, customer properties and constraints, a *process definition* is needed to complete the model of a recommender application (see Figure 3).

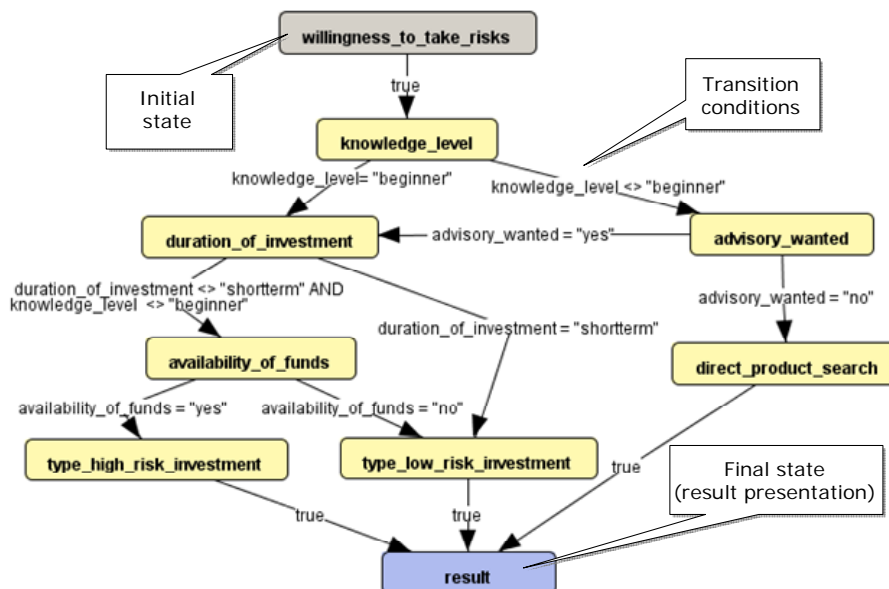


Figure 3: Recommender process definition.

A *recommender process definition* represents possible navigation paths which define the way the system adapts the formulation of questions, explanations, error messages, etc. to the knowledge level and interests of the customer. Such process definitions are based on the formalism of a predicate augmented finite state recognizer (PFSR) [14]. In such a PFSR, constraints describe

transitions between different states (see Figure 3). Based on layout template definitions [22], knowledge bases and process definitions can be automatically translated into an executable recommender application. Figure 4 depicts example screenshots of a financial service recommender application. First, the advisor poses questions to the customer (a). Answers provided by the customer (requirements) serve as input for the calculation of a solution (b). If no solution can be found, alternative repair actions are presented to the customer (c). Selecting such a repair action guarantees the identification of a solution (see Section 3.1).

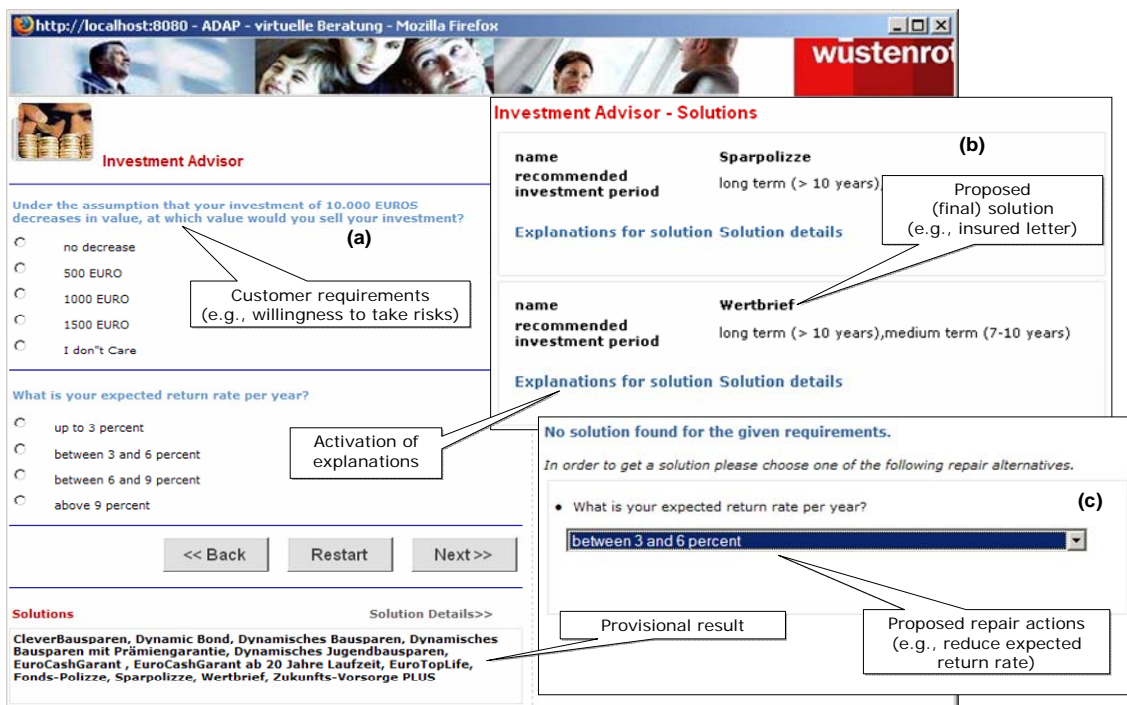


Figure 4: Example user interface of a financial services recommender application.

2.3 CWAdvisor Test Designer

The increasing complexity of recommender knowledge bases makes quality assurance a critical task [26] [31]. Mechanisms have to be provided which allow us to guarantee the correctness of calculated recommendations. In order to avoid situations in which inconsistencies are detected by a customer who interacts with the recommender application, the recommender development environment must indicate potential sources of inconsistencies as early as possible. Therefore,

automated testing functionalities have been integrated into the CWAdvisor development environment. In Test Designer, process definitions (see Section 2.2) are the basis for automatically generating *test cases* for recommender knowledge bases. A *test case* is the combination of an *input sequence* and a set of corresponding *results* calculated by the current version of the recommender knowledge base. An input sequence represents a set of customer requirements (see Section 2.2) for which a corresponding result has to be calculated by the knowledge base. Typically, domain experts are responsible for checking the correctness of results calculated for a given set of input sequences. This *validation* step is necessary in order to make the test cases applicable for regression testing. Figure 5 depicts a simple example of the generation of input sequences. The basis for the generation of input sequences is a set of paths which have been selected from a given recommender process definition (in our case only one path is selected). For each selected path, we can derive a query which generates a set of input sequences. For the purpose of automated input sequence generation, customer properties are represented as tables, their possible instantiations are represented as tuples of the table (see Figure 5). If we execute the query of Figure 5 by only using path transition conditions as selection criteria, we receive 12 different input sequences as query result (see Figure 5). The following concepts allow us to further reduce the number of input sequences:

- *Equivalence partitioning*. Variable domains can be split up into *equivalence classes* out of which we can select a representative subset of input values. When selecting equivalence classes, the interchangeability of the values within a class needs to be taken into account because different variable values inside a class should not change a corresponding result.¹ In the example of Figure 5 we assume that the domain expert selects *knowledge level* ('average',

¹ Currently, the selection of equivalence classes is the task of domain experts. In future versions of the test environment we will include mechanisms supporting the automated derivation of equivalence classes.

'expert') as single equivalence class and *knowledge level* = 'average' as representative input value. This input value is included as additional restriction criterion in the input sequence generation query of Figure 5.

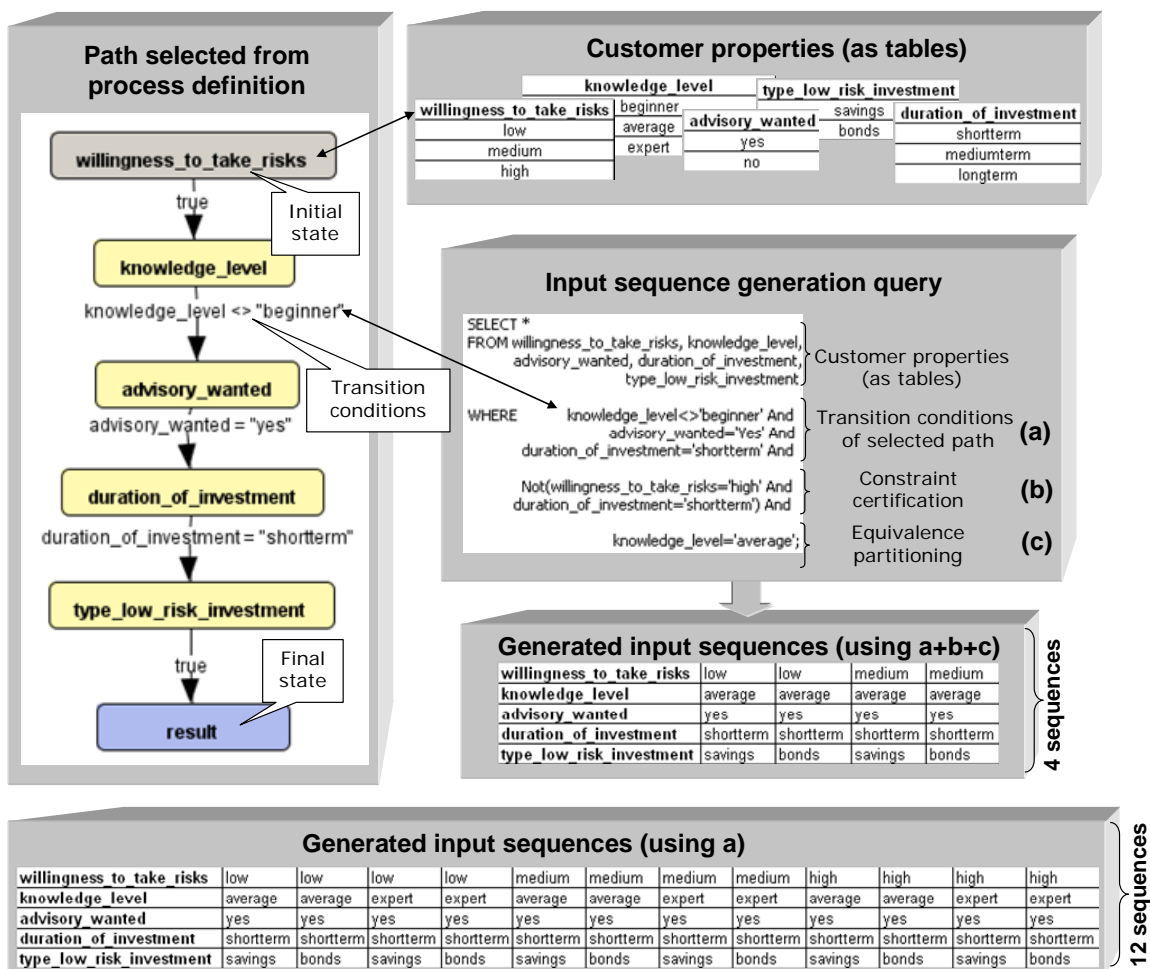


Figure 5: Generation of input sequences (for test cases).

- *Constraint Certification.* By the inclusion of incompatibility constraints which have been considered as correct by domain experts (certified constraints), additional input sequences can be eliminated. Those sequences which satisfy the negated incompatibility constraint are filtered out. If the constraint *a high willingness to take risks is incompatible with short term investment periods* is certified, we can filter out all sequences with related assignments ($willingness\ to\ take\ risks = 'high' \wedge duration\ of\ investment = 'shortterm'$). If we apply the

concepts of constraint certification and equivalence partitioning to our example, the number of relevant input sequences can be reduced to 4 (see Figure 5).

- *Variables with no effects.* In some situations, the advisor asks questions which have no influence on the recommendation. When recommending pension products, a customer may be asked to make a decision concerning *returns on investment (annuity payment, singular payment)*. Since all pension products allow a decision to be made at the end of the investment period, the given answer does not have any effect on the recommendation. Therefore, all values of the corresponding customer property can be assigned to a single equivalence class.

The example depicted in Figure 5 is a simple one. A typical recommender application such as an investment advisor comprises about 20 possible paths. Each path is defined by about 7 variables and 5 possible values per variable. Without including any additional restrictions, the input sequence generation queries for such a setting would generate 1.5mio sequences. This does not allow an effective result validation by domain experts. However, additional restrictions (transition conditions of paths, equivalence partitioning, constraint certification, variables with no effects) can significantly reduce the number of test cases: our experiences from the financial services domain indicate a potential reduction of 1.5mio test cases to the number of about 500-1000.

3 AI Technologies in CWAdvisor

Using a knowledge-based approach, the relationship between customer requirements and products can be *explicitly* modeled [12] [13]. Such a representation is the precondition for applying the technologies discussed in the following subsections.

3.1 Representing Recommendation Knowledge and Calculating Solutions

The first step when building an advisor is the construction of a *recommender knowledge base* which consists of two variable sets (V_C, V_{PROD}) and three sets of constraints (C_R, C_F, C_{PROD}).

- Customer Properties (V_C) describe possible customer requirements, such as *willingness to take risks* ('low', 'medium', 'high') or *knowledge level* ('beginner', 'average', 'expert').
- Compatibility Constraints (C_R) are restricting the possible combinations of customer requirements: $\text{not}(\text{willingness to take risks} = \text{'low'} \wedge \text{expected return rate} = \text{'>9\%'})$.
- Product Properties (V_{PROD}) describe possible product instances: *product type* ('savings', 'bond', 'equity fund') or *risk level* ('low', 'medium', 'high').
- Filter Constraints (C_F) establish the relationship between customer properties and product properties: $\text{knowledge level} = \text{'beginner'} \rightarrow \text{risk level} \diamond \text{'high'}$.
- Allowed instantiations of product properties (offered set of products) are represented by constraints (C_{PROD}) which define restrictions on the possible instantiations of variables in V_{PROD} .

In order to calculate recommendations (solutions), we have implemented a relational query-based approach², in which a set of customer requirements makes a *conjunctive query*. The query is constructed from the consequent part of those filter constraints of C_F whose condition is consistent with the given customer requirements (active filter constraints). The consequent part of our example filter constraint $\text{knowledge level} = \text{'beginner'} \rightarrow \text{risk level} \diamond \text{'high'}$ is translated into the expression $\text{risk level} \diamond \text{'high'}$ as part of the corresponding conjunctive query (in the case of $\text{knowledge level} = \text{'beginner'}$ being a customer requirement). Accordingly, V_{PROD} is represented by a set of table attribute definitions (the product table) and C_{PROD} is represented by tuples whose values represent instantiations of the attributes defined in V_{PROD} . Furthermore, customer properties (V_C) are represented as input variables where the compatibility (C_R) of the corresponding instantiations is ensured by a consistency checker. The execution of the conjunctive query on a product table results in a set of recommendations which are presented to the customer.

² A recommendation task can be represented as a Constraint Satisfaction Problem (CSP) as well [13] [39].

Relaxing Filter Constraints. Filter-based approaches have well-known limitations. In the case of inconsistent customer requirements, all products in the catalog are filtered out and no recommendation can be given. This problem is addressed in CWAdvisor using the concept of query relaxations in the sense of [18] [27] [28]. If *none* of the available products fulfills the conjunctive query that is constructed from *active filter constraints*, we trigger the calculation of a Maximum Succeeding Subquery (XSS): individual atoms are deleted from the conjunctive query in order to identify products that fulfill as many of the active filter constraints as possible. CWAdvisor implements a conflict-directed approach to the calculation of *minimal* relaxations of a given query. The computation of conflicts is based on the QuickXPlain algorithm [25] which solves a conflict detection problem using a *divide and conquer* strategy. Preferences related to the elimination of filter constraints (priorities of filter constraints) can be defined a priori by domain experts. They know that most digital camera users would rather settle on a camera of a brand that is not their first choice than accept a camera with a different resolution. The second approach for determining preferences is to directly ask the customer during an advisory session. XSS calculation is based on the resolution of minimal conflicts [25] detected in a conjunctive query (implemented as a version of the standard algorithm for calculating hitting sets [32]). All of our advisory applications follow the strategy of immediately computing an (optimal) relaxation when no product fulfills all customer requirements. An optimal relaxation can be determined by selecting those filter constraints with the *minimum relaxation costs*. Relaxation costs are determined by a corresponding cost function [23]. A simple cost function is, for example, the sum of filter priorities in a relaxation candidate, where higher values represent higher priorities.

Repairing Customer Requirements. There are domains in which filter relaxation is not possible or desirable. In many cases, filter constraints in the financial services domain cannot be relaxed for reasons such as legal regulations or quality aspects. Simply reporting retrieval failures without

making further suggestions of how to recover from such a situation is not acceptable, however. Therefore, we need to find out which requirements the customer is willing to change. We aim to find possible compromises (repair actions) that can be presented to the customer. Similar to the computation of filter relaxations, the identification of a minimal set of repair actions is based on the calculation of hitting sets [11] [32]. The goal is to identify a *minimal* set of variable settings in a set of requirements that should be changed in order to find a solution. A simple example of the calculation of repair actions is shown in Figure 6.

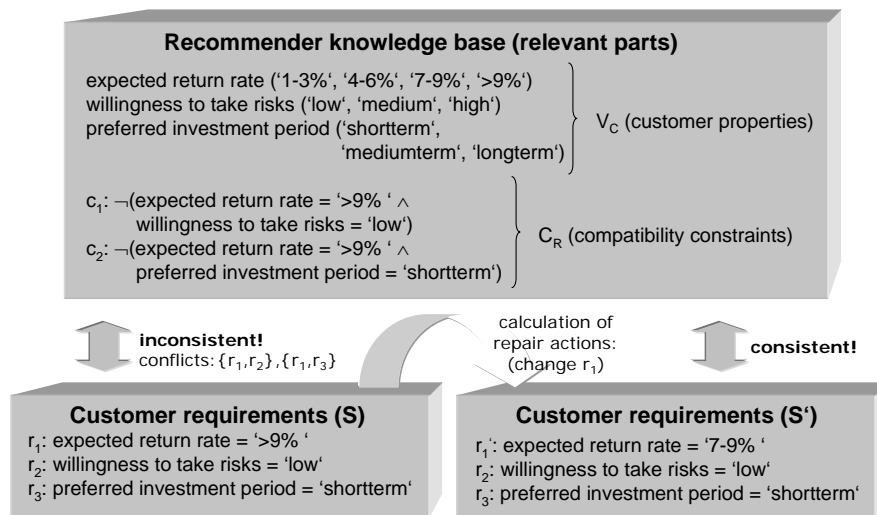


Figure 6: Simple example of the repair of customer requirements.

In this example, $S \cup C_R$ has no solution because $\{r_1, r_2\} \cup C_R$ and $\{r_1, r_3\} \cup C_R$ are inconsistent and therefore both $\{r_1, r_2\}$ and $\{r_1, r_3\}$ induce a conflict [25] with the given compatibility constraints. Based on the hitting set algorithm [11] [32], we have to resolve each of the given conflicts: in our example we simply have to change the setting of the *expected return rate*. Consequently, one possible repair for S is to change the requirement r_1 : *expected return rate*='>9%' to r_1 : *expected return rate*='7-9%'. This makes $S' \cup C_R$ consistent ($S' = \{r_1$: *expected return rate*='7-9%', r_2 : *willingness to take risks*='low', r_3 : *preferred investment period*='shortterm'}). Another repair alternative would be $S' = \{r_1$: *expected return rate*='>9%', r_2 : *willingness to take risks*='medium', r_3 : *preferred investment period*='mediumterm'}.

3.2 Personalization

Dialogue Control. Customers have different preferences of how to specify their requirements: from the direct specification of product parameters (such as a certain savings account running for three years) to a general specification of their personal goals (for example, financing their children's education). Depending on the answers provided, the dialogue with the customer can be adapted as follows (the knowledge level of a customer is derived either from answers to a set of test questions or from a self-assessment at the beginning of an advisory session):

- *Alternative formulation of questions.* Questions posed to expert users can be differentiated from those posed to customers with less knowledge about the product domain. Questions are defined during design time; they are not dynamically composed during program execution. The selection of questions strictly depends on the process definition (see Section 2.2).
- *Rule-based formulation of default answers.* If the goal of the customer is to *put money away for a rainy day* the default answer to a question related to the maximum accepted decrease in value of the investment is *no value decrease accepted*.
- *Alternative explanations for constraint violations.* If the customer has no detailed knowledge about the product domain (novice), a very general explanation about changes in the pension law is given. More detailed information can be included for experts.

Utility/Ranking of Solutions. A solution for a recommendation task is a set of products or services. The order of products within a solution should strictly reflect the degree the products correspond to the wishes of a customer. CWAdvisor supports multi-attribute object rating [2], where each product is evaluated according to a predefined set of interest dimensions. *Profit*, *availability* and *risk* are examples for such dimensions in the financial services domain. If a customer is strongly interested in products with high return rates, compared to availability and risk, profit is a very important dimension. Consequently, the order of products depends on the

importance the individual dimensions have for the customer. The set of products that are part of a solution is ordered using the formula $g(x) = \sum_{(i=1..n)} e_i s_i(x)$, where n denotes the number of dimensions, $g(x)$ represents the utility of a product x , e_i represents the interest of the customer in dimension i , and $s_i(x)$ is the contribution of product x to dimension i . Values for e_i can be automatically derived from customer requirements on the basis of scoring rules. The score 9 for the dimension *profit* in Table 1 can be derived from the customer requirement *expected return rate* = '>9%' where the corresponding scoring rule for the dimension *profit* can be defined as *return rate* = '>9%' → 9).³ Another alternative is to weigh the interest dimensions conforming to an assessment provided by the customer. Values for $s_i(x)$ have to be defined at design time and manually specified for each product. Financial services can be ranked as follows (see Table 1)⁴: for *customer*₁, $g(savings) = 9*1 + 4*8 + 7*8 = 97$, whereas for *customer*₂, $g(savings) = 6*1 + 5*8 + 1*8 = 54$. The utility of *savings* is higher for *customer*₁ than for *customer*₂.

product	profit	availability	risk	dimension	customer ₁	customer ₂
savings	1	8	8	profit	9	6
bonds	4	2	2	availability	4	5
equity funds	9	2	0	risk	7	1

Table 1: Utility of solutions (object ratings and customer preferences).

Presentation of Solutions. For each solution, a set of *immediate explanations* [17] is calculated. These explanations are derived from *active* filter constraints. An explanation related to our example filter condition *knowledge level* = 'beginner' → *risk level* <> 'high' could be *this product assures adequate return rates with a lower level of related risks*. Consequently, explanations related to filter conditions are predefined at design time.

Product Comparison. Product comparisons provide basic mechanisms to compare different products that are part of a recommendation result. Product comparison is based on the definition of rules stating under which conditions an argumentation for/against a certain product should be

³ Note that the score related to an interest dimension (e.g., profit) is not necessarily based on a single requirement.

⁴ In this example, scores are taken from a scale between 0 and 10.

displayed. If the price of product A is significantly higher than the price of product B, the product comparison component should display a corresponding hint. In this context, the level of significance has to be defined by the domain expert. Product comparisons are currently used in our digital camera recommender application of geizhals.at (see Section 4).

Handling of Profiles. The user interface relies on a user model that describes capabilities and preferences of individual customers. Some of these properties are directly provided by the customer (e.g., *name* or *personal goals*). Other properties are derived using scoring mechanisms which relate user answers to abstract *dimensions* [2] such as *preparedness to take risks* or *knowledge about financial services*. This information is stored in the profile and can be exploited in future advisory sessions.

4 Experiences from Industrial Projects

4.1 Digital Camera Advisor

In the domain of consumer electronics, customers are typically confronted with a large variety of products. In our case (digital cameras of the price comparison platform geizhals.at), customers can choose among more than 600 different models. The platform geizhals.at informs approximately one million unique clients per month about the best price of products from domains such as computer hardware, consumer electronics or household machines (about 85,000 products offered by more than 700 online shops). Customers can navigate in a product catalog which is organized in hierarchical categories. For each product, an ordered list of prices and additional user ratings of the corresponding online shops are presented to the customer. The business model of geizhals.at is built on paid advertisements and pay-per-click commissions of the enlisted online shops. The digital camera advisor project was not only initiated to provide a better service for experts, but also to support new user groups with less product domain

knowledge. Therefore, the dialogue includes interaction paths that allow the specification of technical features such as *optical zoom* as well as answers to problem-oriented questions such as *what is your preferred motif?* The digicam advisor was introduced in 2003. Since then, more than 200,000 users have successfully completed their advisory sessions.

Our evaluation of the digital camera advisor consisted of $n=1,253$ online users. An announced lottery ensured that participants identified themselves with their genuine names and addresses and no duplicate questionnaires were counted. The sample consisted of arbitrary online users of the digital camera section of *geizhals.at*, who did not necessarily buy a digital camera. The goal of the questionnaire was to evaluate to what extent the advisor helped users to find what they were looking for and which of the provided advisory features were most useful. Therefore, the key hypothesis of the evaluation was that *advisor applications help users to better orientate themselves when being confronted with a large product assortment*. We asked users whether they had noticed and used the advisor, and whether they had found the product they were looking for. The answers verified our hypothesis: a significantly higher share of users had successfully completed their product search among those who employed the advisor ($\chi^2(1) = 9.39$; $p < 0.01$) which demonstrates an interdependence between the observed variables (see Table 2). Furthermore, we wanted to know which features of a recommender application users especially liked. Interviewees were asked to rate their subjective benefit from the different features of the application. *Product comparisons* on the result page and the *easy and quick way to access the right products* were rated highest (see Table 3). None of the results changed significantly when allocating the sample by gender, age or technical expertise. A study on consumer behavior in the interaction with knowledge-based recommender applications confirmed above results (for details

see [15]). This study was conducted within the scope of the COHAVE⁵ project. Examples of results of this study are the following:

- Advisors supporting *product comparisons* clearly outperform advisors without this functionality (*trust in recommended products* was one of the relevant features herein). The reason for this lies in the lower mental workload of users when differences between alternatives are clearly presented.
- Knowledge-based advisors clearly outperform simple product lists regarding dimensions such as overall *satisfaction with the advisory support* or *trust in recommended products*.
- Finally, explanations of recommendation results significantly contribute to an increased perceived *conformance between expected and recommended products*.

Found the right products	Did use the digicam advisor	
	Yes	No
Yes	276 (76%)	604 (67%)
No	85 (24%)	288 (33%)
Total	361	892

Table 2: Feedback of users accessing the digital camera section of geizhals.at.

Application features	(<i>very</i>) <i>high</i> usefulness
Product comparisons	74.4%
Easy and quick access to the <i>right</i> products	72.7%
Explanation of product properties	57.6%

Table 3: CWAdvisor functionalities rated *high* or *very high* regarding usefulness.

4.2 Financial Services Advisor

CWAdvisor has been deployed for 1,400 sales representatives (insurance brokers and employees of the company) of the *Wüstenrot* building and loan association [13].⁶ In this context, financial service recommender applications have been integrated into an existing Customer Relationship Management (CRM) environment that supports sales representatives by providing functionalities such as calculations, presentation of technical product details, quotation generation or scheduling

⁵ Consumer Buying Behavior & Decision Modeling for Recommender Systems (Austrian Research Fund 810996).

⁶ See Figure 4 for an example screenshot of a deployed investment recommender application.

of meetings. The recommender applications support sales representatives in the preparation, conduction, and summary of sales dialogues. In this context, they help explain recommendation results. The motivation for the development of knowledge-based financial services recommender applications was twofold. First, time effort related to the preparation, conduction and completion of sales dialogues should be reduced. Second, an automated explanation of advisory results should be supported in order to take into account regulations of the European Union [10] relating to an improved documentation of advisory sessions in the financial services domain. On average, a sales representative sells about 60-70 products per year, experts sell up to 500. One and a half years after the initial deployment of the advisor applications we interviewed sales representatives (n=52) with the goal to evaluate the acceptance of the provided advisory applications and to detect potentials for further improvements. The most important topics/questions of the survey are contained in Table 4.

Expertise level. About 23.1% of the interviewees were beginners (focus is on selling 1-3 different types of products), 28.9% were advanced sales representatives (selling more than 3 different types of products) and about 48.0% were experts (selling all product types).

Satisfaction with advisory support. In order to evaluate the sales representatives' satisfaction with the provided advisor functionalities, we compared evaluations related to the Wüstenrot CRM environment without advisory support with evaluations related to the environment with a corresponding advisory support (two separate questions). A one-sample z-test [7] resulted in a significant increase of the average level of satisfaction with the provided advisory functionality ($z = 10.83, p < 0.01$). Furthermore, there is a positive correlation between the estimated time savings related to the advisory support and the satisfaction with the provided advisory functionalities (Pearson's $r = 0.49$) [7].

Importance of advisory support. The interviewees strongly agreed on the importance of the provided advisory functionality (67.3% high importance, 21.1% average importance). Major arguments for the application of knowledge-based recommender technologies were (a) automated consistency checks which improve the correctness of offers (this was significantly more important for beginners and representatives selling less than 50 products per year), and (b) the automated generation of advisory protocols which allow for time savings in the introductory dialogue as well as in the quotation phase.

Major application area. Most of the sales representatives (81.8%) reported that they used CWAdvisor functionalities throughout the sales dialogue or for the generation of documentations for completed advisory dialogues (protocols). Each such protocol includes a summary of customer preferences, a listing of recommended products, an argumentation as to why the recommendations suit the needs and wishes of the customer (explanations), and a listing of further issues to be discussed with the customer. Furthermore, 53.8% of the sales representatives reported that they also used the advisors at home in order to prepare for a sales dialogue.

Time savings. On average, the time savings caused by the application of financial service advisors were specified with 11.73 minutes per advisory session. Most savings are explained by the generation of advisory protocols at the end of a session and available summaries of previous dialogues at the beginning of a new session. Assuming that an average sales representative conducts about 170 advisory sessions per year, time savings can amount up to 33 hours per year for each person. For a sales expert, time savings can amount up to 100 hours per year.

Importance for new representatives. Most of the sales representatives (97%) definitely agreed on the potential of knowledge-based recommender technologies to provide e-learning support. Due to this feedback, a corresponding project has already been initiated which exploits recommender

technologies in order to support learning processes for new sales representatives. The software will be applied in the context of sales courses.

Topic	Description
Expertise level	Expertise level of the sales representative
Satisfaction with advisory support	Overall satisfaction with the advisory support of the CRM environment (with/without CWAdvisor)
Importance of advisor support	Importance of CWAdvisor for sales dialogues
Major application area	Major application area for CWAdvisor
Time savings	Time savings in sales dialogues directly related to the application of CWAdvisor
Importance for new representatives	Importance of CWAdvisor for educating (new) sales representatives

Table 4: Major topics/questions of the questionnaire.

4.3 The Role of Knowledge Acquisition in Recommender Projects

Due to a lack of programming skills [21], there is a significant discrepancy between technical experts who are able to create software artifacts and domain experts who are not. Consequently, domain experts without technical knowledge are in the position of being solely responsible for providing domain knowledge and knowledge engineers transform this knowledge into the formal representation of an underlying knowledge base. This process is quite error-prone and is also known as the *knowledge acquisition bottleneck*. In order to reduce this bottleneck, our goal was to empower autonomous development and maintenance processes for end users. Such an integration of end users has great potential to significantly improve the effectiveness of recommender application development. Within the scope of the development of the CWAdvisor environment we took into account the following principles which are crucial for supporting effective knowledge acquisition and maintenance processes [6] [8]:

- Rapid prototyping follows the principle of *concreteness*. The user immediately sees the effects of changing explanation texts, product properties, images, transition conditions, etc. in the recommender knowledge base or process definition. In CWAdvisor, such changes are performed on a graphical level (see Section 2.2); the corresponding recommender

applications can be automatically generated from graphical representations. Such an automated application generation is extremely important to make the development of advisors feasible for domain experts (end users) without a special IT education [3].

- A flexible and simple graphical design of the intended behavior of the user interface protects the user from programming details. CWAdvisor provides simple modeling concepts that allow the explicit specification of the intended behavior of the recommender user interface (principle of *explicitness*). This approach also follows the principle of separating application logic design from implementation details.
- Effective test support follows the principle of *immediate feedback*. Potential sources of inconsistencies in the knowledge base caused by the creation of a new knowledge base version are indicated promptly. Especially automated test case generation and regression testing have been identified as prerequisites for effective end-user debugging [35]. Following this approach, errors in the recommender knowledge base can be detected immediately and are not forwarded to the productive environment where, in the worst case, those errors are detected by customers. Thus, domain experts can establish a higher level of trust in recommendations, which is extremely important for our projects.

Our experiences show that all these principles are important for the development and maintenance of recommender knowledge bases. We compared time efforts needed for the development of advisors before and after CWAdvisor was available. Due to the graphical development support, the overall development efforts were reduced significantly. In the case of the Wüstenrot project, three domain experts are responsible for the development of knowledge bases (investment & financing, pension & life insurance, property insurances). The first knowledge base versions were developed in cooperation with knowledge engineers. Now, most of the maintenance activities, including updates of products, (in)compatibility tables, filter

conditions, and process definitions, are conducted by domain experts. The time efforts related to knowledge base maintenance done by knowledge engineers are now negligible. Changes to current knowledge base versions are conducted in a test environment. After the completion of changes in the knowledge base, a set of pre-generated test cases (see Section 2.3) is used for regression testing. Thereafter, the new version of the knowledge base is synchronized with the productive environment (synchronization with the client applications of sales representatives). In the case of geizhals.at, knowledge engineers who are knowledgeable about digital cameras are responsible for maintaining the knowledge base. In this context, knowledge bases must be designed as generic as possible in order to be stable regarding the frequent changes in the product assortment (technical information such as memory size or resolution is constantly changing).

5 Related Work

Recommender Technologies. An overview of different applications of recommender technologies can be found in [29] [36], overviews of different technological approaches to the implementation of recommender applications can be found in [1] [5] [38]. Basically, there are three widely known technological approaches to the implementation of recommender applications. *Collaborative filtering systems* [20] automate the recommendation process on the basis of user opinions about items and generate new recommendations based on inter-user comparisons. Because recommendations follow from a comparison between the target user and other users, a user with few ratings becomes difficult to categorize which is also known as the *new user* problem. Similarly, a new item that does not have many ratings also cannot be easily recommended – this is known as the *new item* problem. *Content-based filtering* [30] is a special type of information filtering that uses features of items the user has liked in the past to infer new recommendations. In the case of collaborative filtering as well as in the case of content-based

filtering, user profiles are long-term models. Both approaches [20] [30] do not exploit deep knowledge about the product domain and therefore are excellent techniques supporting recommendation processes for simple products such as books or movies. A major strength of these approaches is that no additional knowledge acquisition and maintenance efforts are needed. *Knowledge-based recommender technologies* receive increasing attention in research [4] [5] [13] [22] [24] [39]. In contrast to collaborative and content-based filtering approaches, knowledge-based approaches are based on an explicit representation of product, marketing and sales knowledge. Using such representations, new item and new user problems are avoided since recommendations are directly derived from user preferences identified within the scope of the requirements elicitation phase. The main reason for choosing a knowledge-based recommendation approach stems from the requirements of domains such as financial services where deep product knowledge is needed in order to identify and *explain* solutions. Jiang et al. [24] present an approach to multimedia-enhanced product recommendation where basic recommendation technologies are additionally equipped with a component supporting the visualization of results. Thompson et al. [39] focus on the integration of conversational natural language interfaces with personalized recommender systems with the major goal of reducing system-user interactions. Long-term user preferences are obtained in the course of normal recommendation dialogues and are used to adapt future dialogs with the same user [39]. A study conducted in the restaurant recommendation domain found that users needed fewer interactions to find a good restaurant when using a personalized conversational recommender than when using a non-personalized one [39]. Both natural language interaction [39] and result visualization [24] are not provided by CWAdvisor but are within the scope of future work. Compared to the approaches of [4] [5] [24] [39], CWAdvisor additionally provides a graphical recommender application development and test environment and supports the calculation of repair actions for customer

requirements in situations in which no solution can be found. Both functionalities are extremely important when deploying knowledge-based advisors in industrial environments.

Knowledge Base Validation. Within the context of knowledge-based systems development, validation technologies have, in many cases, not been adopted by practitioners; ad hoc techniques are still dominating [31]. The literature on test case generation in knowledge-based systems development is still largely directed at rule-based types of systems [31]. It is not directly applicable to knowledge-based recommendation. Tiihonen et al. [40] present an approach to the testing of configurator applications, in which the configuration model is considered as consisting of a set of local requirement groups representing a set of potential inputs provided by the user. A test case is represented by a group of requirement items, and test case generation is based on randomly selecting requirement groups. In contrast to our work, [40] directly deal with the generation of test cases from partonomies, whereas our approach defines test cases on the basis of results of conjunctive queries that are related to paths in recommender process definitions. Furthermore, no mechanisms for reducing the number of test cases are presented in [40]. This is, however, crucial for the validation of knowledge bases by domain experts.

6 Conclusions

Effective knowledge acquisition processes and intuitive interaction mechanisms that support the user in the product and service selection process are major preconditions for successful recommender applications. Both aspects are supported by the recommender technologies discussed in this paper. First, the CWAdvisor knowledge acquisition environment provides innovative development, maintenance and testing techniques for recommender knowledge bases. Second, model-based diagnosis techniques actively support customers in situations in which no solution could be found. Our goal for the future is to further increase the acceptance of

knowledge-based recommender technologies. In order to reach this goal, we will integrate psychological theories from the area of consumer buying behavior into the design of our recommender environment. Following this strategy, CWAdvisor technologies continuously improve recommender application development processes and contribute to an improved accessibility of complex product and service assortments.

7 Acknowledgements

Parts of the work presented in this paper have been done within the scope of the project Koba4MS (Knowledge-based Advisors for Marketing and Sales – FFG-808479).

8 References

1. Adomavicius, G. and Tuzhilin, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17, 6 (2005), 734–749.
2. Ardissono, L.; Felfernig, A.; Friedrich, G.; Jannach, D.; Petrone, G.; Schäfer, R.; and Zanker, M. A Framework for the development of personalized, distributed web-based configuration systems. *AI Magazine*, 24, 3 (2003), 93–108.
3. Bhargava, H.; Sridhar, S.; and Herrick, C. Beyond spreadsheets: tools for building decision support systems. *IEEE Computer*, 32, 3 (1999), 31–39.
4. Burke, R.; Hammond, K.; and Young, B. The FindMe approach to assisted browsing. *IEEE Expert*, 12, 4 (1997), 32–40.
5. Burke, R. Knowledge-based recommender systems. *Encyclopedia of Library & Information Systems*, 69, 32 (2000).
6. Burnett, M. HCI research regarding end-user requirement specification: a tutorial. *Knowledge-Based Systems*, 16 (2003), 341–349.
7. Dalgaard, P. *Introductory Statistics with R*. Springer, 2004.
8. DiGiano, C.; Kahn, K.; Cypher, A.; and Smith, D. Integrating learning supports into the design of visual programming systems. *Visual Languages & Computing*, 12 (2001), 501–524.
9. Edvardson, J. *A Survey on Automatic Test Data Generation*. 2nd Conference on Computer Science and Engineering CCSSe'99, Linköping, Sweden, 1999, pp. 21–28.
10. EU (2002). *Richtlinie 2002/92/EG des Europäischen Parlaments und des Rates vom 9. Dezember 2002 über Versicherungsvermittlung*. Amtsblatt der EU, 2002.

11. Felfernig, A.; Friedrich, G.; Jannach, D.; and Stumptner, M. Consistency-based diagnosis of configuration knowledge bases. *AI Journal*, 2, 152 (2004), 213–234.
12. Felfernig, A.; Friedrich, G.; Jannach, D.; Stumptner, M.; and Zanker, M. Configuration knowledge representations for Semantic Web applications. *AI Engineering Design, Analysis and Manufacturing Journal*, 17 (2003), 31–50.
13. Felfernig, A. and Kiener, A. *Knowledge-based Interactive Selling of Financial Services using FSAdvisor*. *17th Innovative Applications of Artificial Intelligence Conference (IAAI'05)*, Pittsburgh, Pennsylvania: AAAI Press, 2005, pp. 1475–1482.
14. Felfernig, A. and Shchekotykhin, K. *Debugging User Interface Descriptions of Knowledge-based Recommender Applications*. *ACM Conference on Intelligent User Interfaces*, Sydney, Australia: ACM Press, 2006, pp. 234–241.
15. Felfernig, A. and Gula, B. *An Empirical Study on Consumer Behavior in the Interaction with Knowledge-based Recommender Applications (to appear)*. *IEEE Conference on e-Commerce Technology (CEC'06)*, San Francisco: IEEE, 2006.
16. Fleischanderl, G.; Friedrich, G.; Haselböck, A.; Schreiner, H.; and Stumptner, M. Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems*, 13, 4 (1998), 59–68.
17. Friedrich, G. *Elimination of Spurious Explanations*. *16th European Conference on Artificial Intelligence (ECAI 2004)*, Valencia, Spain: IOS Press, 2004, pp. 813–817.
18. Godfrey, P. Minimization in cooperative response to failing database queries. *International Journal of Cooperative Information Systems* 6, 2 (1997), 95–149.
19. Goodwill, J. *Mastering JSP Custom Tags and Tag Libraries*. Wiley Publishers, 2002.
20. Herlocker, J.; Konstan, J.; Terveen, L.; and Riedl, J. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22, 1 (2004), 5–53.
21. Ioannidou, A. *Programmorphism: a Knowledge-Based Approach to End-User Programming*. *INTERACT'03*, Zürich, Switzerland: IOS Press, 2003, pp. 152–159.
22. Jannach, D. and Kreutler, G. *A Knowledge-Based Framework for the Rapid Development of Conversational Recommenders*. Zhou, X.; Su, S.; Papazoglou, M.; Orłowska, M.; and Jeffery, K. (eds.): *WISE'04*, Brisbane, Australia. *LNCS 3306*, 2004, pp. 390–402.
23. Jannach, D. and Liegl, J. *Conflict-directed Relaxation of Constraints in Content-based Recommender Systems (to appear)*. *IEA/AIE 2006*, Annency, France, 2006.
24. Jiang, B.; Wang, W.; and Benbasat, I. Multimedia-based interactive advising technology for online consumer decision support. *Communications of the ACM*, 48, 9 (2005), 93–98.
25. Junker, U. *QuickXPlain: Preferred Explanations and Relaxations for Over-Constrained Problems*. *AAAI'04*, San Jose: AAAI Press, 2004, pp. 167–172.
26. Kirani, S.; Zualkernan, I.; and Tsai, T. Evaluation of expert system testing methods. *Communications of the ACM*, 37, 11 (1994), 71–81.

27. McSherry, D. *Maximally Successful Relaxations of Unsuccessful Queries*. 15th Conference on Artificial Intelligence and Cognitive Science, Galway, Ireland, (2004), pp. 127–136.
28. Mirzadeh, N.; Ricci, F.; and Bansal, M. *Supporting User Query Relaxation in a Recommender System*, Zaragoza, Spain: LNCS 3182, 2004, pp. 31–40.
29. Montaner, M; Lopez, B.; and De la Rose, J. A taxonomy of recommender agents on the internet. *Artificial Intelligence Review* 19 (2003), 285–330.
30. Pazzani, M. and Billsus, D. Learning and revising user profiles: the identification of interesting web sites. *Machine Learning* 27 (1997), 313–331.
31. Preece, A.; Talbot, S.; and Vignollet, L. Evaluation of Verification Tools for Knowledge-Based Systems. *International Journal of Human-Computer Studies*, 47 (1997), 629–658.
32. Reiter, R. A theory of diagnosis from first principles. *AI Journal*, 23, 1 (1987), 57–95.
33. Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. *GroupLens: An Open Architecture for Collaborative Filtering of Netnews*. ACM Conference on Computer Supported Cooperative Work, Chapel Hill, NC: ACM, 1994, pp. 175–186.
34. Ricci, F.; Venturini, A.; Cavada, D.; Mirzadeh, N.; Blaas, D.; and Nones, M. *Product Recommendation with Interactive Query Management and Twofold Similarity*. 5th Intl. Conference on Case-Based Reasoning. Trondheim, Norway, 2003, pp. 479–493.
35. Ruthruff, J. and Burnett, M. *Six Challenges in Supporting End-User Debugging*. Workshop on End-User Software Engineering, Saint Louis, Missouri, 2005.
36. Schafer, J.; Konstan, J.; and Riedl, J. Electronic Commerce recommender applications. *Journal of Data Mining and Knowledge Discovery*, 5, 1/2 (2000), 115–152.
37. Smyth, B.; Balfe, E.; Boydell, O.; Bradley, K.; Briggs, P.; Coyle, M.; and Freyne, J. *A Live User Evaluation of Collaborative Web Search*, 19th International Joint Conference on Artificial Intelligence. Edinburgh, Scotland: IJCAI, 2005, pp.1419–1424.
38. Terveen, L. and Hill, W. *Beyond recommender systems: Helping people help each other*. HCI in the New Millennium, Addison Wesley, 2001.
39. Thompson, C.; Göker, M.; and Langley, P. A Personalized System for Conversational Recommendations. *Journal of Artificial Intelligence Research* 21 (2004), 393–428.
40. Tiihonen, J.; Soinen, T.; Niemelä, I.; and Sulonen R. *Empirical Testing of a Weight Constraint Rule Based Configurator*. ECAI Configuration Workshop, 2002, pp. 17–22.