Koba4MS: Selling Complex Products and Services using Knowledge-based Recommender Technologies

Alexander Felfernig
Computer Science and Manufacturing, University Klagenfurt
Universitätsstraße 65-67, A-9020 Klagenfurt, Austria
phone:++43/463/2700/3754
alexander.felfernig@uni-klu.ac.at

Abstract

The increasing complexity of products and services offered by online stores and electronic marketplaces makes the identification of appropriate solutions a challenging task. Customers can differ greatly in their expertise and level of knowledge w.r.t. such product assortments. Consequently, intelligent sales assistance systems are required which support customers with intuitive and personalized dialogs. Knowledge-based recommender systems meet these requirements by allowing a flexible mapping of product, marketing and sales knowledge to the formal representation of a knowledge base. This paper presents the domain-independent knowledge-based recommender system Koba4MS which assists customers and sales representatives by guaranteeing the consistency and appropriateness of proposed solutions, identifying additional selling opportunities and by providing intelligent explanations for identified results. Using examples from the financial services domain we show how constraint satisfaction, model-based diagnosis, personalization and intuitive knowledge acquisition techniques support the effective implementation of customer-oriented sales dialogs. Finally, we present experiences gained from commercial projects.

1 Introduction

Electronic commerce provides convenient mechanisms for buying and selling products and services. However, buying complex products and services (e.g. financial services, computers, etc.) is still a challenging task since many organizations offer simple query interfaces under the assumption that customers know the technical details of the offered set of products [1, 25]. *Recommender technologies* [1, 2, 3, 10, 14, 18, 19] improve this situation by providing solution alternatives for the customer which are automatically derived from a set of customer requirements. There are three basic approaches to the implementation of recommender applications. *Collaborative Filtering* [10, 18, 19] is based on the concept of storing preferences of a large set of customers. Based on the assumption that human preferences are correlated, recommendations given to a customer are derived from preferences of a group of customers with similar interests, i.e. no deep knowledge about product properties is needed. Similarly, using *Content-based Filtering* [3, 14], products are described by a set of keywords (categories) which are stored in a customer profile in the case that a customer buys a certain product. The next time, the customer enters the system, the stored preferences are used for identifying additional products which are assigned to similar categories. Finally, *Knowledge-based Recommender* applications (advisors) [1, 2] exploit deep knowledge about the product domain in order to determine solutions exactly fitting to the wishes and needs of the customer. When selling complex products such as financial services, a customer's taste is not of primary concern - primarily solutions and explanations must be correct in every case. This requirement can only be met by explicitly representing product, marketing, and sales knowledge [7, 13], i.e. *Knowledge-based Recommender* applications (advisors) are the natural choice in this context. In the following we give an overview of the major technologies implemented within the *Koba4MS*[1] environment, a domain-independent tool designed for the development of knowledge-based advisors. The major difference between *Koba4MS* and other knowledge-based recommender systems [2] is the inclusion of model-based diagnosis [6, 17] and personalization techniques [1] which improve the effectiveness of advisor development as well as the interaction with the advisor. For example, a graphical development and test environment makes the implementation of advisors

[1] *Knowledge-based Advisors for Marketing and Sales* is a project funded by the Austrian Research Fund (agreement number FFF-808479).

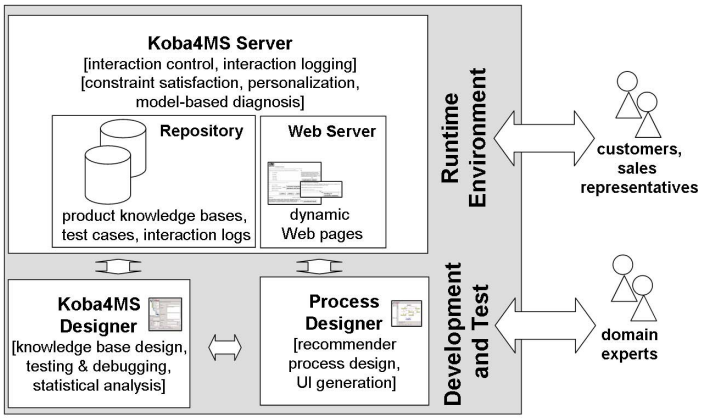

Figure 1. Overall architecture.

feasible for non-programmers, furthermore intelligent diagnosis and repair techniques actively support customers in situations where no solution could be found. *Koba4MS* can be applied in the following scenarios.

- Direct customer support. Similar to traditional sales channels, improved sales assistance generates added value for customers. On the one hand knowledge-based advisors allow an intuitive access to complex products for customers, on the other hand sales representatives are relieved from routine advisory jobs.

- Support of sales representatives. Sales representatives interact with advisors when talking with the customer, where guided dialogs provide questions and explanations focusing on the customers wishes and needs, i.e. support a customer-oriented sales dialog.

Throughout the paper we provide real-world examples from the financial services domain which is our leading application domain. Financial service advisory is a knowledge-intensive task which in many cases overwhelms customers as well as sales representatives. Therefore financial service providers ask for tools providing an intuitive access to their product assortment.[2]

The remainder of the paper is organized as follows. In Section 2 we present the *Koba4MS* recommender environment. In Section 3 we present examples for the usage of AI technologies which allow the implementation of knowledge-based advisors (constraint satisfaction, model-based diagnosis and test, personalization, knowledge acquisition). Finally, Section 4 presents experiences from commercial advisor projects.

[2]Products of financial service providers cover different areas of interest such as investment decisions, financing, pension, life insurance, etc.

2 Koba4MS Environment

The *Koba4MS* toolsuite (see Figure 1) provides comprehensive assistance for customers and sales representatives by supporting guided and personalized dialogs allowing an intuitive access to an assortment of complex products and services. In this context *Koba4MS* can be used for the following purposes.

- supporting the formalization of product, marketing and sales knowledge by non-programmers.

- testing and debugging of knowledge bases in order to identify faulty constraint definitions.

- checking customer requirements for consistency and (in the case of inconsistencies) supporting a corresponding error handling.

- matching customer requirements to a set of product properties, i.e. calculating a solution.

- diagnosing and repairing a set of inconsistent customer requirements, i.e. proposing minimal changes which allow the retrieval of a solution.

- explaining solutions in order to increase the confidence of the customer.

Koba4MS technologies are used in application domains such as financial services, digital cameras, cigars, computers, services in public administration etc. In the financial services domain *Koba4MS* technologies are applicable for the following reasons.

- Solutions must be objective, correct and explainable which makes approaches such as *Collaborative Filtering* or *Content-based Filtering* not the best choice.

- Typically, financial service providers want to develop advisors autonomously, i.e. knowledge representation formalisms are needed which allow the development of recommender knowledge bases for non-programmers (this is supported by graphical knowledge acquisition, model-based debugging and testing).

- Intelligent explanation, debugging, and repair mechanisms as well as automated test case generation are using model-based knowledge representations, i.e. deep knowledge about the application domain must be available (which is not available in *Collaborative Filtering* or *Content-based Filtering* approaches).

- Financial services recommendation is a complex task with a large number of constraints and possible solutions. In this context, knowledge-based approaches can significantly reduce efforts related to advisor development and maintenance.

Similar reasons motivate the application of knowledge-based advisors in other application domains such as online-selling of computers, digital cameras, etc.

2.1 Overall Architecture

Koba4MS product knowledge bases and process definitions are developed and maintained using a *Development and Test* environment (*Koba4MS Designer* and *Process Designer*). Products are defined within *Koba4MS Designer* or imported from external systems using an XML interface (for details see [1]). In the following advisors are automatically generated and made available for customers (e.g. online-stores, e-marketplaces, etc.) and sales representatives (e.g. intranet applications or installations on notebooks of sales representatives), where *Koba4MS Server* supports the execution of advisory sessions (*Runtime Environment*).

2.2 Development & Test Environment

Koba4MS Designer. *Koba4MS Designer* is a graphical development environment for knowledge-based recommenders. It is based on Java Web Start which provides a browser-independent architecture for deploying Java-2 based applications on a client. The concepts implemented in *Koba4MS* are based on long-term AI research in the area of knowledge-based configuration and personalization [1, 5, 6, 7, 9]. *Koba4MS Designer* supports the design of advisors, where the relevant set of product- and customer properties is identified and transformed into a *recommender knowledge base* [7, 13]. Such a knowledge base consists of the following parts (see Figure 2).

- *product properties* are structural descriptions of the provided products (e.g. life insurances can be characterised by the *possible length of life assurance policies*, *premiums of life assurance policies*, *links to additional product documentation*, etc.).

- *customer properties* are descriptions of possible customer requirements (e.g. within an investment advisory process the question *under the assumption that your investment of 10.000 EUROS decreases in value, at which value would you sell your investment*? is related to the *willingness to take risks*).

- *constraints* are restricting the combinations of customer requirements and product properties, e.g. *return rates above 9 percent require the willingness to take risks*. Constraints can be defined on the graphical level as well as on the textual level.

In order to support the analysis of advisors, *Koba4MS Designer* provides a statistical analysis component which operates on interaction logs of advisory sessions conducted by online customers or sales representatives.

Process Designer. A *recommender process* represents possible navigation paths which define the way the system adapts its dialog style to the knowledge level and interests of the customer. Such process definitions are based on a predicate augmented finite state recognizer (PFSR) [24] (constraints describe transitions between different states of a recommender process) which represents allowed navigation paths within an advisor (see Figure 2). Transition conditions between states of a recommender process are evaluated using the *Koba4MS* constraint engine. Based on a layout template definition, knowledge bases and process definitions can be automatically (no programming is needed) translated into an executable advisor (see e.g. Figure 4), where each state of the process definition corresponds to a Web-page in the generated application.

Testing & Debugging Knowledge Bases. The increasing size and complexity of recommender knowledge bases makes testing a critical task [12, 16] in the context of successfully deploying and maintaining recommender applications. Figure 3 depicts the basic process for validating solutions calculated by *Koba4MS*. Process definitions (see e.g. Figure 2) are the basis for automatically generating test cases. Solutions (results calculated by the knowledge base) for generated test cases are presented to the domain expert who decides on their validity (*Result Validation*). Correct results are marked as checked by the domain expert, faulty results are used by a diagnosis component (*Knowledge Base Design&Debugging*) for identifying the corresponding faulty constraints in the knowledge base [6]. Test

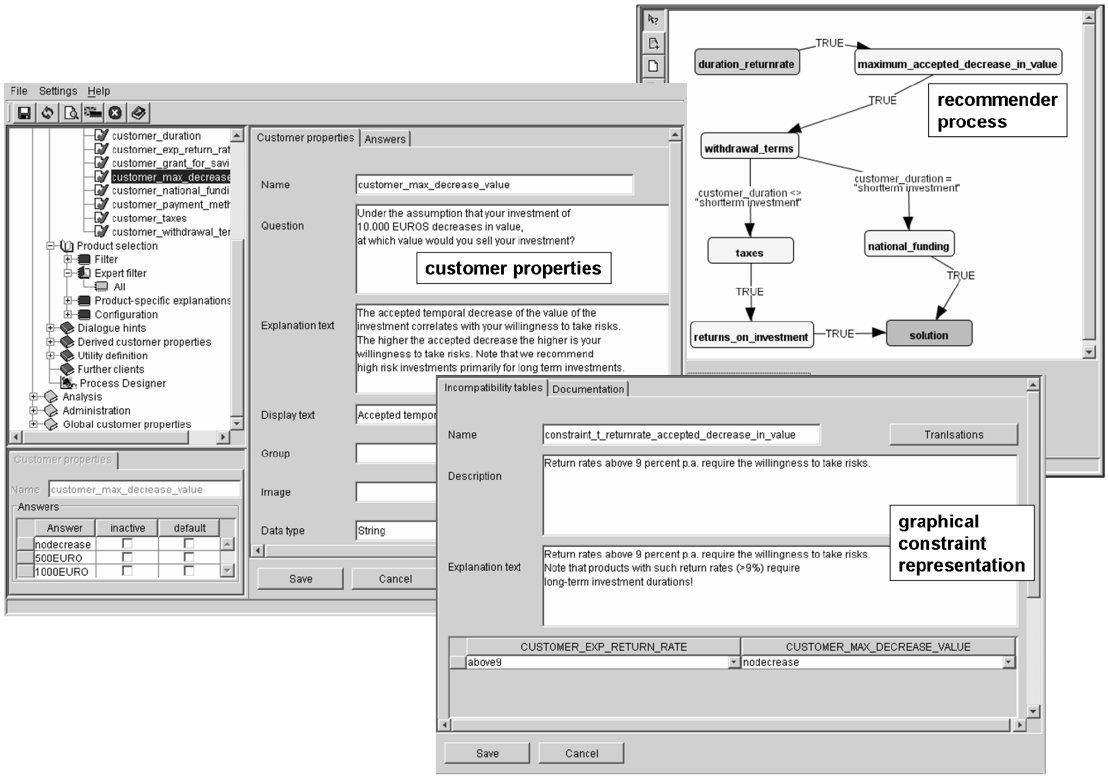

Figure 2. Definition of customer properties, constraints, recommender processes.

cases deemed as correct by the domain expert are used for regression tests [8]. Test case generation in *Koba4MS* follows a path-oriented approach (the test cases for each path are derived from the set of solutions to a corresponding constraint satisfaction problem) which allows a high degree of coverage [4]. The disposable time for testing is restricted, consequently mechanisms are provided which reduce the amount of tests without reducing the coverage of the overall test suite (except for random selections). Typically, domain experts agree with accepting efforts related to quality assurance since solution quality is of serious concern. We can calculate a complete set of test cases which includes all possible transitions of a process definition, but this is only feasible for small and strongly constrained recommendation tasks. The following approaches are reducing the number of test cases within *Koba4MS*.

- Using *equivalence partitioning*, variable domains can be split up into equivalence classes out of which we can select a representative subset of test cases. A person's age can be split up into a set of equivalence classes, e.g. the age under 13, between 13 and 16 years, etc. Depending on the equivalence class, different legal regulations restrict the set of possible financial services which can be recommended, e.g. certain types of building society savings can only be offered to customers between 19 and 21 years, i.e. we can select 20 years as representative value for the equivalence class.
- Test cases including combinations of customer requirements which are inconsistent with the knowledge base can be neglected by *certifying* the corresponding constraints as valid. If such a constraint is certified, we can neglect all test cases with the corresponding assignments e.g. if we certify the incompatibility between no readiness to take risks and high return rates, test cases containing this assignment combination can be neglected.
- Sometimes advisors pose questions which have *no influence* on the solution (marketing questions, where no constraints are defined on the corresponding variable), e.g. when recommending pension products, the customer can be asked to make a decision concerning returns on investment (singular, annuity payment). Since pension products allow a decision to be taken at the end of the investment period, the customer's answer doesn't influence the calculation of the solution.
- Confronted with large variable domains and lengthy

processes, *random selections* are a means to reduce the set of test cases. Different facets of random selection are possible, e.g. path selection or assignment selection (reduction of a variable domain using a statistical distribution).

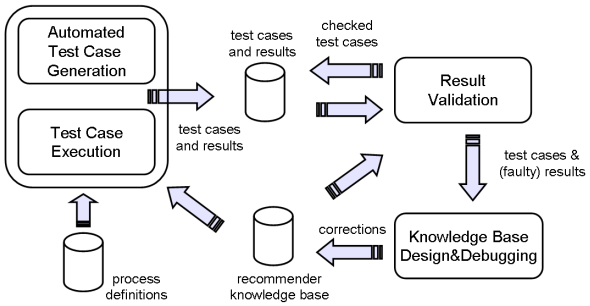

Figure 3. Koba4MS validation process.

2.3 Runtime Environment

Koba4MS Server. The calculation of solutions for a recommendation task is based on constraint satisfaction problem solving [23]. Customer properties as well as product properties are represented as constraint variables. A solution for a given recommendation task (constraint satisfaction problem) is found if all constraints are satisfied. For an example screenshot of an interface see Figure 4.

3 Used Technologies

Compared to *Knowledge-based Recommender* applications [1, 2], *Collaborative Filtering* [10, 19] and *Content-based Filtering* [3, 14] do not exploit deep knowledge about the domain in order to determine solutions fitting to the wishes and needs of the customer. Using knowledge-based approaches, the relationship between customer requirements and financial services can be explicitly modelled in an underlying knowledge base [7]. Such model-based representations are the precondition for applying diagnosis and testing techniques.

3.1 Constraint Satisfaction

Search for Solutions. As already mentioned, *Koba4MS* problem solving is based on constraint satisfaction problem solving. A Constraint Satisfaction Problem (CSP) (C,V,D) [23] is defined by a set V of variables x_i, a set C of constraints c_j and a set D of domains d_i which defines for each variable the set of possible values. A CSP is solved if there exists a set of instantiations of the variables x_1, x_2, ..., x_n s.t. all constraints contained in C are satisfied. A *recommendation task* can be defined as a CSP (C, V_{SRS}, V_{PROD}, D_{SRS}, D_{PROD}), where V is divided into V_{SRS} (set of variables describing customer requirements) and V_{PROD} (set of variables describing product properties). This definition is similar to the concept of functional architectures describing customer requirements and related components representing product properties [13]. If no solution can be found by the search engine, constraints are relaxed starting with constraints with lowest priority. If nothing but non-relaxable constraints (priority = 0) remain and no solution was found, a repair mechanism is activated. In addition to constraints, *Koba4MS* supports *tips*, i.e. constraints representing e.g. cross-selling opportunities which are presented to the customer without interrupting the recommender process. An example for such a tip is: *long-term investments reduce risks*, *i.e. allow higher return rates than short-term investments without taking high risks*.

Diagnosis and Repair of Requirements. If the result set is empty, conventional recommenders tell the user (customer) that no solution was found, i.e. no clear explanation for the reasons for such a situation is given. *Koba4MS* supports the calculation of repair actions for customer requirements (a minimal set of changes allowing the calculation of a solution). If $\Sigma = \{x_1 = a_1, x_2 = a_2, ..., x_n = a_n\}$ is a set of customer requirements ($\Sigma \cup C$ has no solution), a repair is a minimal set of changes to Σ (resulting in Σ') s.t. Σ' $\cup$ C has a solution. The computation of repair actions [6] is based on the Hitting Set algorithm [17] which exploits minimal conflict sets (minimal sets $\Pi \subseteq \Sigma$ of variable instantiations triggering an inconsistency with C) provided by the constraint solver in order to determine minimal diagnoses and corresponding repair actions (see Figure 5 for the representation of repair alternatives).

Execution of recommender processes. The execution of recommender processes, i.e. the interpretation of a recommender process definition, is based on the evaluation of transition conditions of process definitions (see Figure 2). After each user input, transition conditions following the actual state in the process definition are evaluated in order to determine the following state. The result page is the final page (state), where a solution is presented.

Test Case Generation. Automated test case generation (for more details see Section 2.2) is based on the definition of a constraint satisfaction problem. For this purpose a (complete) set of possible paths through a recommender process is determined. For each path a corresponding CSP is generated and executed - identified solutions represent test cases, i.e. possible settings of customer requirements.

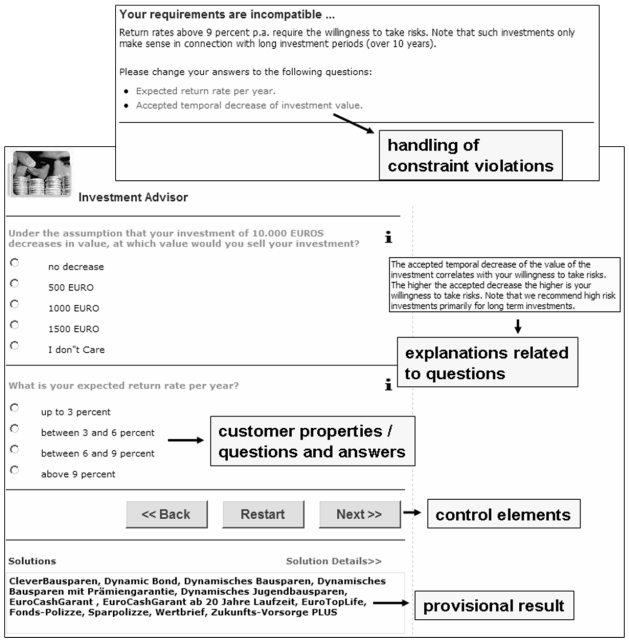

Figure 4. Example user interface.

3.2 Personalization Concepts

Dialog Style. Customers have different approaches to specify their requirements ranging from the direct specification of product parameters (e.g. a certain savings account running for 3 years) to a general specification of their personal goals (e.g. financing their children's education). An adaptation of the interaction style can significantly contribute to an improved approximation to the behavior of a human sales expert (an experienced sales assistant adapts his dialog style to the skill level and interests of customers). Depending on answers already provided by a customer, the dialog style can be personalized as follows.

- Alternative formulation of questions, e.g. questions posed to expert users can be differentiated from those posed to customers with less knowledge about the product/service domain.
- Rule-based formulation of default-answers, e.g. if the goal of the customer is to *put money by 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, e.g. if the customer is a novice, a very general explanation about changes in the pension law is given, more detailed information can be included for experts.

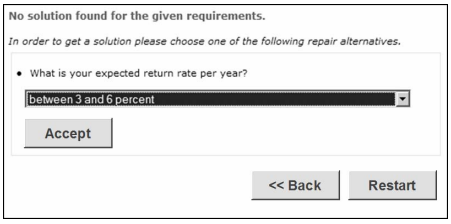

Figure 5. Repair support.

Utility of Repair Proposals. If no solution can be found for a given set of customer requirements, *Koba4MS* provides a set of possible (minimal) repair actions which allow the calculation of a solution. Different customer properties have an assigned priority which indicates the importance of the variable for the customer. The *lower* the priority of the variable the higher the probability is that the variable is considered as focus of repair actions, e.g. if the type of returns on investment (at the end of the investment period, dividend payout) is not important for a customer, this property is primarily considered as a potential candidate for repair actions. More formally, the personalization of repair proposals is based on the formula

$$f(x_1, x_2, ..., x_m) = \sum_{j=1}^{m} p(x_j),$$

where $f(x_1, x_2, ..., x_m)$ represents the utility of repair actions related to the variables $x_1, x_2, ..., x_m$ and $p(x_j)$ denotes the customer-specific priority of variable x_j. Customer-specific priorities can be either defined statically or by a customer within the scope of an advisory session.

Utility of Solutions. A solution for a recommendation task is a set (portfolio) of financial services. The order of solutions should strictly correspond to the degree a solution contributes to the wishes of a customer. *Koba4MS* supports multi-attribute object rating [1], where each solution entry is evaluated w.r.t. to a predefined set of dimensions. *Profit*, *availability* and *risk* are examples for such abstract dimensions. Depending on the weighting of the dimensions for a specific customer (e.g. a customer is strongly interested in products with high return rates, i.e. compared to availability and risk, profit is a very important dimension) the set of solutions 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 solution x, e_i represents the interest of the customer in dimension i, and s_i is the contribution of solution x to dimension i. Using this formula, different financial services can be personalized for a specific customer as follows (see Table 1).[3] For *customer* 1 the value of

[3] Scores are taken from a scale between 0 and 10.

product	profit	avail.	risk
savings	1	8	8
bonds	4	2	2
equity funds	9	2	0

dim.	cust1	cust2
profit	9	6
avail.	4	5
risk	7	1

Table 1. Utility of solutions (*lhs*: object ratings, *rhs*: customer preferences).

$g(savings)$ = 9*1 + 4*8 + 7*8 = 97, whereas for *customer* 2 the value of $g(savings)$ = 6*1 + 5*8 + 1*8 = 54, i.e. savings better fit to *customer* 1 (the utility of *savings* is higher for *customer* 1).

Presentation of Solutions. For each solution a set of *immediate explanations* [9] is calculated, i.e. a set of explanations which are derived from variable assignments directly dependent on selections already made during search. Furthermore, *solution-specific explanations* are supported, e.g. if the customer is strongly interested in high return rates and a solution shows a remarkable return rate, this fact is explicitly mentioned when the solution is presented to the customer. In contrast to *immediate explanations* (derived in the search process), *solution-specific explanations* are related to explicitly defined explanation constraints.

Handling of Profiles. *Koba4MS* includes mechanisms allowing the adaptation of the dialog style to the user's skills and needs [1]. The user interface relies on the management of a user model that describes capabilities and preferences of individual customers. Some of these properties are directly provided by the user (e.g. *name or personal goals*, or self-estimates such as *knowledge about financial services*), other properties are derived using personalization rules and scoring mechanisms which relate user answers to abstract dimensions [1] such as *preparedness to take risks* or *interest in high profits* (dimensions describing the users interests) and *knowledge about funds, etc.* (dimensions describing the users knowledge about the domain). Initial values for the customer profile are collected in a requirements analysis phase where best-matching stereotypes are applied to complete profiles.

3.3 Knowledge Acquisition

Graphical Design Environment. One major requirement imposed by financial service providers is the availability of a graphical design environment for recommender knowledge bases. *Koba4MS Designer* and *Process Designer* allow the design and maintenance of recommender knowledge bases and process definitions by non-programmers. Constraint schemes (see e.g. Figure 2), i.e. graphical interfaces for defining constraints, alleviate the management of recommender knowledge bases. Incompatibility and requirement relationships are currently available as constraint schemes within the *Koba4MS* environment.

Contextual constraint representation. In many cases constraints are defined within a specific *context*, e.g. constraints related to customers interested in long-term investments. Having defined a context *long-term investments*, the condition *customer_duration_of_investment = longterm* can be omitted when defining context-related constraints. Contexts can be defined using an environment for the textual definition of constraints.

Knowledge Base Debugging. Effective debugging support for the implementation of recommender knowledge bases is a critical issue for a successful development and maintenance of advisors. In *Koba4MS* we have implemented model-based diagnosis algorithms [6, 17] supporting the identification of minimal sources of inconsistencies in recommender knowledge bases. Similar to the diagnosis and repair of customer requirements, we apply model-based diagnosis techniques in order to identify a minimal set of constraints $\in C$ which - when deleted from the knowledge base - allow consistency restoration.

4 Experiences from Projects

A graphical development environment guaranteeing the maintainability of applications is a major prerequisite for successfully implementing a knowledge-based advisor. In the financial services domain the implementation and maintenance of knowledge bases must be supported for non-programmers, i.e. the knowledge acquisition component must provide intuitive modelling concepts. The correctness of solutions plays a vital role for the acceptance of the system by sales representatives applying the system while communicating with the customer. Domain experts do not use formal knowledge representation formalisms on a daily basis, i.e. effective *test and debugging* support is extremely useful and significantly improves the effectiveness of the overall advisor development process. Experiences from projects indicate a reduction of efforts related to knowledge base development of about 30-50 percent. Specifying and executing test cases helps domain experts to better understand and debug knowledge bases. The complete set of possible test cases for a knowledge base with 20 customer properties with a domain of cardinality 5 would comprise about 5^{20} test cases which is definitely infeasible for a domain expert. Reducing the input space to 20 possible paths each path defined by 7 variables and 5 possible values per variable reduces the number of potential test cases to 1.5

mio which is still unfeasible. By applying additional restrictions the number of test cases can be reduced to about 500-1000. The following conclusions can be drawn from the actual projects based on *Koba4MS* technologies.

- Knowledge Acquisition. Experiences from projects[4] show that graphical knowledge acquisition is a major precondition for enabling the design and maintenance of recommender knowledge bases and significantly reduces the knowledge acquisition bottleneck between domain experts and knowledge engineers.

- Cross Selling. *Koba4MS* indicates cross-selling opportunities with a corresponding set of explanations as to why a solution is useful for the customer. The analysis of sales records e.g. in the digital camera domain shows significant improvements in the sales of add-on and niche products which were neglected previously.

- Routine advisory tasks. Effort reductions related to routine advisory tasks are reported, e.g. financial services advisory provided on the homepage relieves sales representatives from routine advisory jobs.

- Documentation. Added value is provided by explanations for calculated service portfolios which are used as starting point for future advisory sessions. Furthermore, legal regulations can force companies to provide intelligent reporting for the customer, e.g. due to regulations of the European Union, financial service providers are forced to improve the documentation of advisory sessions - intelligent reporting is required which includes explanations as to why certain products were offered to the customer.

- *Koba4MS* knowledge bases are developed and tested by marketing and sales experts. Sales representatives can rely on the solutions calculated by the financial advisor and can provide qualified explanations.

- A set of applications has been implemented on the basis of the recommender technologies presented in this paper, e.g. the digital camera advisor PIXLA which was implemented for the largest Austrian online product platform (www.geizhals.at). This application exhibits about 10.000 successful advisory sessions per month. Users of www.geizhals.at were interviewed before and after the introduction of PIXLA. The major result of the study was a statistically significant increase of customer satisfaction (related to dimensions such as easiness to find products etc.).

[4] See e.g. *www.hypo-alpe-adria.at* (investment advisor) or *www.geizhals.at* (digital camera advisor deployed on the largest Austrian online product platform).

5 Related Work

Basically there are three approaches to the implementation of recommender applications. *Collaborative Filtering* [10, 19] and *Content-based Filtering* [3, 14] do not exploit deep knowledge about the product domain. Collaborative Filtering is based on the assumption that customer preferences are correlated, i.e. similar products are recommended to customers with similar interest profiles. Content-based filtering focuses on the analysis of a given set of products already ordered by a customer. Based on this information, products are recommended which resemble products already ordered (products related to similar categories). Additionally, there exist a number of approaches combining these basic approaches in order to gain an improved quality of the resulting solutions (see e.g. [20]). Using *knowledge-based* approaches, the relationship between customer requirements and offered products is explicitly modelled [7]. Such model-based knowledge representations are the major precondition for the application of model-based diagnosis and testing techniques.

Within the context of knowledge based systems development, validation technologies in many cases have not been adopted by practitioners, ad hoc techniques still dominate [16, 11]. Although testing is considered the most pragmatic and successful technique in quality assurance, the research field is still insufficiently explored. The literature on test case generation in knowledge-based systems development is still largely directed at rule-based types of knowledge-based systems [16], i.e. not directly applicable to model-based approaches such as knowledge-based recommender systems. [22] present an approach to the testing of configurator applications, where 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, [22] directly deal with the generation of test cases for partonomies representing product structures, whereas the approach presented in this paper presents test case generation as the task of finding solutions for a given Constraint Satisfaction Problem (CSP). [4] presents an approach to test data generation based on the analysis of program flowgraphs, where test sets are generated by the execution of the flow graph, i.e. the interpretation of different path predicates which restrict the number of paths and possible test sets. Compared to our approach, the work of [4] is based on flowgraph execution whereas our work focuses on solving a test case generation CSP.

The increasing size and complexity of knowledge bases motivated the application of model-based diagnosis (MBD) [17] in knowledge-based systems development. [6] concentrate on the identification of errors in a given configuration

knowledge base. In cases where either positive examples (test cases) are not consistent with the configuration knowledge base or negative examples are accepted by the knowledge base, a debugging process of the configuration knowledge base is initiated (if negative examples are accepted by the knowledge base, it is the task of the user to find adequate extensions (constraints) triggering the rejection of those examples). Practical experiences related to the implementation of the concepts presented in [6] are discussed in [8]. An introduction to the concepts of MBD diagnosis and an overview of existing applications can be found in [15]. An overview on the application of model-based diagnosis techniques in software debugging can be found in [21].

6 Conclusions

In this paper we have presented the *Koba4MS* toolsuite which supports the implementation of knowledge-based recommender applications (advisors). The toolsuite is based on innovative AI technologies (model-based diagnosis, personalization, constraint satisfaction) which provide an intuitive access to complex products and services for customers as well as for sales representatives. *Koba4MS* includes a graphical development-, test- and debugging-environment which allows the development and maintenance of recommender knowledge bases for non-programmers. The applicability of the presented concepts has been shown within the context of commercial projects.

References

[1] L. Ardissono, A. Felfernig, G. Friedrich, D. Jannach, G. Petrone, R. Schaefer, and M. Zanker. A Framework for the development of personalized, distributed web-based configuration systems. *AI Magazine*, 24(3):93–108, 2003.

[2] R. Burke. Knowledge-based Recommender Systems. *Encyclopedia of Library & Information Syst.*, 69(32), 2000.

[3] R. Burke. Hybrid Recommender Systems: Survery and Experiments. *User Modeling and User-Adapted Interaction*, pages 331–370, 2002.

[4] J. Edvardson. A Survery on Automatic Test Data Generation. In 2^{nd} *Conference on Computer Science and Engineering CCSSe'99*, 1999.

[5] A. Felfernig, G. Friedrich, and D. Jannach. UML as domain specific language for the construction of knowledge-based configuration systems. *IJSEKE*, 10(4):449–469, 2000.

[6] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner. Consistency-based Diagnosis of Configuration Knowledge Bases. *AI Journal*, 2(152):213–234, 2004.

[7] A. Felfernig, G. Friedrich, D. Jannach, M. Stumptner, and M. Zanker. Configuration knowledge representations for Semantic Web applications. *AI Engineering Design, Analysis and Manufacturing Journal*, 17:31–50, 2003.

[8] G. Fleischanderl. Suggestions from the software engineering practice for applying consistency-based diagnosis to configuration knowledge bases. In 13^{th} *International Workshop on Principles of Diagnosis (DX-02)*, 2002.

[9] G. Friedrich. Elimination of Spurious Explanations. In 16^{th} *European Conference on Artificial Intelligence (ECAI 2004)*, pages 813–817, 2004.

[10] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. on Information Systems*, 22(1):5–53, 2004.

[11] P. Jorgensen. *Software Testing, A Craftsman's Approach*. CRC Press, 2002.

[12] S. H. Kirani, I. A. Zualkernan, and W. T. Tsai. Evaluation of Expert System Testing Methods. *Communiations of the ACM*, 37(11), 1994.

[13] S. Mittal and F. Frayman. Towards a Generic Model of Configuration Tasks. In *Proceedings* 11^{th} *International Joint Conf. on AI*, pages 1395–1401, Detroit, MI, 1989.

[14] M. Pazzani. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.

[15] B. Peischl and F. Wotawa. Model-Based Diagnosis or Reasoning from First Principles. *IEEE Intelligent Systems*, 18(3):32–37, 2003.

[16] A. Preece, S. Talbot, and L. Vignollet. Evaluation of Verification Tools for Knowledge-Based Systems. *International Journal of Human-Computer Studies*, 47:629–658, 1997.

[17] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 23(1):57–95, 1987.

[18] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.

[19] B. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Item-based collaborative filtering recommendation algorithms. In 10^{th} *Int. World Wide Web Conf.*, pages 285–295, 2001.

[20] Y.-Y. Shih and D.-R. Liu. Hybrid recommendation approaches: collaborative filtering via valuable content information. In *38th Hawaii International Conference on System Sciences (HICSS'05)*, page 217b, Big Island, Hawaii, 2005.

[21] M. Stumptner and F. Wotawa. A Survey of Intelligent Debugging. *European Journal on Artificial Intelligence (AICOM)*, 11(1):35–51, 1998.

[22] J. Tiihonen, T. Soininen, I. Niemelä, and R. Sulonen. Empirical Testing of a Weight Constraint Rule Based Configurator. In *ECAI 2002 Workshop on Configuration*, Lyon, France, 2002.

[23] E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, 1993.

[24] G. v.Noord and D. Gerdemann. Finite State Transducers with Predicates and Identities. *Grammars*, 4(3):263–286, 2001.

[25] B. Xiao, E. Aimeur, and J. Fernandez. PCFinder: An Intelligent Product Recommendation Agent for E-Commerce. In *IEEE International Conference on E-Commerce (CEC'03)*, pages 181–188, 2003.