A Wiki-based Environment for Constraint-based Recommender Systems Applied in the E-Government Domain

Stefan Reiterer¹, Alexander Felfernig¹, Michael Jeran¹, Martin Stettinger¹, Manfred Wundara², and Wolfgang Eixelsberger³

 ¹ Institute for Software Technology, A-8010 Graz, Inffeldgasse 16b firstname.lastname@ist.tugraz.at
 ² City of Villach, A-9500 Villach, Rathausplatz 1 manfred.wundara@villach.at
 ³ Carinthia University of Applied Sciences, A-9524 Villach, Europastraße 4 w.eixelsberger@fh-kaernten.at

Abstract. Constraint-based recommenders support customers in identifying relevant items from complex item assortments. In this paper we present WEEVIS, a constraint-based environment that can be applied in different scenarios in the e-government domain. WEEVIS supports collaborative knowledge acquisition for recommender applications in a MediaWiki-based context. This paper shows how Wiki pages can be extended with recommender applications and how the environment uses intelligent mechanisms to support users in identifying the optimal solutions to their needs. An evaluation shows a performance overview with different knowledge bases.

1 Introduction

Constraint-based recommender applications help users navigating in complex product and service assortments like digital cameras, computers, financial services and municipality services. The calculation of the recommendations is based on a knowledge base of explicitly defined rules. The engineering of the rules for recommender knowledge bases (for constraint-based recommenders) is typically done by knowledge engineers, mostly computer scientists [1]. For building high quality knowledge bases there are domain experts involved who serve the knowledge engineers with deep domain knowledge [1]. Graphical knowledge engineering interfaces like [2] improved the maintainability and accessability and moved the field one step further.

Other recommendation approaches like collaborative filtering use information about the rating behavior of other users to identify recommendations [3,4]. Content-based filtering [5] exploits features of items for the determination of recommendations. Compared to these approaches, constraint-based recommenders are more applicable for complex products and services due to their explicit knowledge representation. In the line of Wikipedia⁴ where users build and maintain Wiki pages collaboratively we introduce WEEVIS⁵. WEEVIS is a MediaWiki⁶ based environment that exploits the properties of MediaWiki and enables community based development and maintenance of knowledge bases for constraint-based recommenders. WEEVIS is freely available as a platform and successfully applied by four Austrian universities (in lectures about recommender systems), in the financial services domain and in e-government .

In the e-government domain officials as well as the community residents can take numerous advantages of knowledge-based recommenders:

- WEEVIS can be used as an online advisory service for citizens for example for documents that are necessary to apply for a private construction project. The online recommendation of necessary documents in advance to on-site appointments can lead to a time reduction for community residents and community officials.
- WEEVIS can be used for modeling internal processes like the signing of travel applications for example a community official wants to visit a conference, based on different parameters like the conference type, or if it's abroad or in the domestic area, different officials have to sign the travel request. In WEEVIS the appropriate rules for such internal processes can be mapped and especially for new employees WEEVIS recommenders can provide substantial assistance.
- WEEVIS can be used as an information platform for example with integrated knowledge-based recommenders for community residents e.g. to identify the optimal waste disposal strategy for a household (this example is used as a running example in this paper, see Section 2). Instead of providing plain text information, like common municipality web pages, the knowledge representation as a recommender provides an easier way for community members to identify the optimal solution for their situation.

A recommender development environment for single users is introduced in [2]. This work is based on a Java platform and focuses on constraint-based recommender applications for online selling. Compared to [2], WEEVIS provides a wiki-based user interface that allows user communities to develop recommender applications collaboratively. Instead of an incremental dialog, where the user answers one question after the other, like [2], WEEVIS provides an integrated interface where the user is free to answer questions in any order.

The WEEVIS interface also provides intelligent mechanisms for an instant presentation of alternative solutions in situations where it is not possible to find a solution for a given set of user (customer) requirements, i.e., the requirements are inconsistent with the recommendation knowledge base and the user is in the need for repair proposals to find a way out from the *no solution could be found dilemma*. Model-based diagnosis [6] can be applied for the identification

⁴ www.wikipedia.org

⁵ www.weevis.org

⁶ www.mediawiki.org

of faulty constraints in a given set of customer requirements. In this context efficient divide-and-conquer based algorithms [7] can be applied to the diagnosis and repair of inconsistent requirements. The environment supports the user with integrated model-based diagnosis techniques [6,8]. A first approach to a conflict-directed search for hitting sets in inconsistent CSP definitions was introduced by [9]. With regard to diagnosis techniques, WEEVIS is based on more efficient techniques that make the environment applicable in interactive settings [8, 10].

A Semantic Wiki-based approach to knowledge acquisition for collaborative ontology development is introduced, for example, in [11]. Compared to [11], WEEVIS is based on a recommendation domain specific knowledge representation (in contrast to ontology representation languages) which makes the definition of domain knowledge more accessible also for domain experts.

The remainder of this paper is organized as follows. In Section 2 we present an overview of the recommendation environment WEEVIS and it's application in the e-government domain. In Section 3 we present results of a performance evaluation that illustrates the performance of the integrated diagnosis technologies. With Section 4 we conclude the paper.

2 WeeVis Overview

Since WEEVIS is based on the MediaWiki platform, it can be installed on freely available web servers. On the website www.weevis.org a selection of different WEEVIS recommenders is publicly available. For internal processes WEEVIS can be deployed in the local intranet. Standard wiki pages can be complemented easily by recommender knowledge bases. Currently, WEEVIS calculates recommendations based on previously entered requirements. If the requirements would result in a no solution could be found message WEEVIS calculates alternative solutions based on diagnoses (see Section 2.4). In line with the Wiki idea, WEEVIS provides the ability to build knowledge bases collaboratively, a valuable feature in e-government domain, because depending on the community department multiple people are responsible for data management and administration. Furthermore, WEEVIS exploits the basic functionalities provided by MediaWiki and allows rapid prototyping processes where the result of a change can immediately be seen by simply switching from the *edit mode* to the corresponding *read mode*. This approach allows an easy understanding of the WEEVIS tags and also of the semantics of the provided WEEVIS language.

2.1 WeeVis User Interface

Since WEEVIS is a MediaWiki-based environment the user interface relies on the common Wiki principle of the *read* mode (see Figure 1) for executing a recommender and the *write* mode (see Figure 2) for defining a recommender knowledge base. The development and maintenance of a knowledge base is supported a textual fashion with a syntax that is similar to the standard Wiki syntax (see Figure

Waste Disposal Strategy



Fig. 1. Waste Disposal Strategy a simple recommender knowledge base from the *e*-government domain (WEEVIS read mode).

2). In the following we will present the concepts integrated in the WEEVIS environment on the basis of a working example from the e-government domain. More specifically we present a recommender that supports households in identifying their optimal waste disposal strategy. In this recommendation scenario, a user has to specify his/her requirements regarding, for example, the number of persons living in the household or how frequently the containers should be emptied. A corresponding WEEVIS user interface is depicted in Figure 1. Requirements are specified on the left and the corresponding recommendations for the optimal waste disposal plan are displayed in the right hand side.

For each solution, a so-called support score is determined. If a solution fulfills all requirements, this score is 100%, otherwise it is lower and, when clicking on the score value, a corresponding repair action is displayed on the left-hand side (see Figure 1). Due to the automated alternative determination, WEEVIS is always able to present a solution and users are never ending up in the *no* solution could be found dilemma (see Figure 1).

An example of the definition of a (simplified) e-government recommender knowledge base is depicted in Figure 2. The definition of a recommender knowledge base is supported in a textual fashion on the basis of a syntax similar to MediaWiki. Basic syntactical elements provided in WEEVIS will be introduced in the next subsection.

2.2 WeeVis Syntax

A WEEVIS recommender consists of three necessary aspects, the definition of questions and possible answers, items and their properties, and constraints (see Figure 2).

4

Editing "Waste Disposal Strategy"

```
F K Ab 🗞 A = 📐 🐼 Cu —
A very simple "Waste Disposal Strategy" recommender.
<recommender>
  & PRODUCTS
    {!name!sizep!emptyingp!#pricep!
     |Small Plan | 60 | monthly | 300 |
     |Medium Plan|120|monthly|300|
     |Large Plan|120|weekly|400|
  &OUESTIONS
    { |persons?(one to two, three to four, more than four) |
     |maxprice?#(200,800,200, Euro)|
     |emptying?(monthly, weekly)|
     |container size? (120, 60)|]
  &CONSTRAINTS
     |persons? = more than four &INCOMPATIBLEWITH container size? = 60|
     |persons? = more than four &INCOMPATIBLEWITH emptying? = monthly|
     |maxprice? >= #pricep|
     [&IF container size? = 120 &THEN sizep = 120]
[&IF container size? = 60 &THEN sizep = 60]
     [&IF container size? = 120 &THEN sizep <> 60]
     |&IF emptying? = monthly &THEN emptyingp = monthly|
     |&IF emptying? = weekly &THEN emptyingp = weekly|}
  &TEST
    {|show| persons? = more than four, container size? = 60, emptying? =
monthly|}
</recommender>
```

Fig. 2. The Waste Disposal Strategy recommender knowledge base (view source (edit) mode).

The definition of an item assortment in WEEVIS starts with the $\[mathcal{B}PROD-UCTS$ tag (see Figure 2). The first line represents the attributes separated by the exclamation mark. In our example, the item assortment is specified by the *name*, *sizep*, the container size, *emptyingp*, the emptying frequency, and *pricep*, the price of the waste disposal plan. Each of the next lines represents an item with the values related to the attributes, in our example there are three items specified: *Small Plan*, *Medium Plan*, and *Large Plan*.

The second aspect starts with the &QUESTIONS tag. In our example the following user requirements are defined: *persons*, specifies the number of persons living in the household (one to two, three to four, more than four) and *maxprice* specifies the upper limit regarding the price of the waste disposal plan. Furthermore, *emptying* represents the sequence in which the dustbins will be emptied, weekly or monthly, and *container size*, the preferred size of the dust container, 120 or 60.

The third aspect represents the definition of the constraints. Starting with the &CONSTRAINTS tag in WEEVIS different types of constraints can be defined. For the first constraint in our example the &INCOMPATIBLE keyword is used to describe incompatible combinations of requirements. The first incompatibility constraint describes an incompatibility between the number of persons in the household (*persons*) and the *container size*. For example, a waste disposal plan with (container size) 60 must not be recommended to users who live in a household with more than four persons. Filter constraints describe relationships between requirements and items, for example, maxprice \geq pricep, i.e., the price of a waste disposal plan must be equal or below the maximum accepted price.

2.3 Recommender Knowledge Base

A recommendation knowledge base can be represented as a CSP (Constraint Satisfaction Problem) [12] on a formal level. The CSP has two sets of variables $V (V = U \cup P)$ and the constraints $C = PROD \cup COMP \cup FILT$ where $u_i \in U$ are variables describing possible user requirements (e.g., *persons*) and $p_i \in P$ are describing item properties (e.g., *emptyingp*). Each variable v_i has a domain d_j of values that can be assigned to the variable (e.g., *one to two, three to four* or *more than four* for the variable *persons*). Furthermore, there are three different types of constraints:

- COMP represents incompatibility constraints of the form $\neg X \lor \neg Y$
- *PROD* the products with their attributes in disjunctive normal form (each product is described as a conjunction of individual product properties)
- *FILT* the given filter constraints of the form $X \to Y$

The knowledge base specified in Figure 2 can be transformed into a constraint satisfaction problem where &QUESTIONS represents U, &PRODUCTS represents P and &CONSTRAINTS represents PROD, COMP, and FILT. Based on this knowledge representation WEEVIS is able to determine recommendations that take into account a specified set of user requirements. The results collected are represented as unary constraints ($R = \{r_1, r_2, ..., r_k\}$). Finally the determined set of solutions (recommended items) is presented to the user.

2.4 Diagnosis and Repair of Requirements

In situations where requirements $r_i \in R$ (unary constraints defined on variables of U such as *emptying* = *monthly*) are inconsistent with the constraints in C, we are interested in a subset of these requirements that should be adapted to be able to restore consistency. On a formal level we define a *requirements diagnosis task* and a corresponding *diagnosis* (see Definition 1).

Definition 1 (Requirements Diagnosis Task). Given a set of requirements R and a set of constraints C (the recommendation knowledge base), the requirements diagnosis task is to identify a minimal set Δ of constraints (the diagnosis) that has to be removed from R such that $R - \Delta \cup C$ is consistent.

As an example $R = \{r_1 : persons = more than four, r_2 : maxprice = 600, r_3 : emptying = monthly, r_4 : containersize = 60\}$ is a set of requirements inconsistent with the defined recommendation knowledge. The recommendation knowledge base induces two minimal conflict sets (CS) [7] in R which are CS_1 : $\{r_1, r_4\}$ and CS_2 : $\{r_1, r_3\}$. For these requirements we can derive two diagnoses: Δ_1 : $\{r_3, r_4\}$ and Δ_2 : $\{r_1\}$. For example, to achieve consistency of Δ_1 at least

 r_3 and r_4 have to be adapted. Such diagnoses can be determined on the basis of a HSDAG (hitting set directed acyclic graph) (e.g. [13]).

Determining conflict sets [7] at first and afterwards constructing a HSDAG (hitting set directed acyclic graph) to identify diagnoses tends to become inefficient especially in interactive settings. Direct diagnosis algorithms like FAST-DIAG [8] reduce this two-step process to one step by calculating diagnoses directly without conflict determination. This was the major motivation for integrating FASTDIAG [8] into the WEEVIS environment. Like QUICKXPLAIN [7], FASTDIAG is based on a divide-and-conquer approach that enables the calculation of minimal diagnoses without the calculation of conflict sets. In WEEVIS the derived diagnosis are used as a basis for determining repair actions, which lead to the alternative solutions that are be presented to the user. A repair action is a concrete change of one or more user requirements in R on the basis of a diagnosis such that the resulting R' is consistent with C.

Definition 2 (Repair Task). Given a set of requirements $R = \{r_1, r_2, ..., r_k\}$ inconsistent with the constraints in C and a corresponding diagnosis $\Delta \subseteq R$ $(\Delta = \{r_l, ..., r_o\})$, the corresponding repair task is to determine an adaption $A = \{r'_l, ..., r'_o\}$ such that $R - \Delta \cup A$ is consistent with C.

In WEEVIS, repair actions are determined conform to Definition 2. For each diagnosis Δ determined by FASTDIAG, the corresponding solution search for $R - \Delta \cup C$ returns a set of alternative repair actions (represented as adaptation A). In the following, all solutions that satisfy $R - \Delta \cup A$ are shown to the user (see the right hand side of Figure 1).

Diagnosis determination in FASTDIAG is based on a total lexicographical ordering of the customer requirements [8]. This ordering is derived from the sequence of the entered requirements. For example, if $r_1 : persons = morethan four$ has been entered before $r_3 : emptying = monthly$ and $r_4 : containersize = 60$ then the underlying assumption is that r_3 and r_4 are of lower importance for the user and thus have a higher probability of being part of a diagnosis. In our working example $\Delta_1 = \{r_3, r_4\}$. The corresponding repair actions (solutions for $R - \Delta_1 \cup C$) is $A = \{r'_3 : emptying = weekly, r'_4 : containersize = 120\}$, i.e., $\{r_1, r_2, r_3, r_4\} - \{r_3, r_4\} \cup \{r'_3, r'_4\}$ is consistent. The item that satisfies $R - \Delta_1 \cup A$ is $\{LargePlan\}$ (see in Figure 2). The identified items (p) are ranked according to their support value (see Formula 1).

$$support(p) = \frac{\#adaptions \ in \ A}{\#requirements \ in \ R} \tag{1}$$

3 Performance Evaluation

3.1 Description of the evaluation

We have conducted a performance evaluation with the goal to highlight the ability of WEEVIS to calculate repair actions and if no solutions could be found. Therefore we set up an experiment with three WEEVIS recommenders based on the e-government example presented in Section 2. To illustrate the performance of WEEVIS, the knowledge base was extended and deployed with different complexity regarding the number of solutions (&PRODUCTS tag in WEE-VIS), user requirements (&QUESTIONS tag WEEVIS), and constraints (&CON-STRAINTS tag WEEVIS) (see Table 1). According to these three attributes the knowledge bases were classified as *Small*, *Medium*, and *Large*. To fit the attributes of knowledge base *Small* from Table 1, the running example (see Figure 2) was adapted by adding one question, two products and removing the last two constraints. The *Medium* and *Large* knowledge base are extended versions of the running example.

Knowledge base	Number of solutions /	
	requirements / constraints	
Small	5/5/5	
Medium	20/10/15	
Large	50/15/30	

Table 1. The different knowledge bases with sizes *Small*, *Medium* and *Large* used for the performance comparison.

3.2 Results of the evaluation

To provide an optimal user experience a focus of WEEVIS is to provide instant feedback after every interaction. Interacting with a WEEVIS recommender starts with the the entering of new requirements and the subsequent calculation of solutions for these requirements. If no solution could be found WEEVIS calculates one or more diagnoses and the complementing alternative products. With this performance evaluation we show that WEEVIS can identify at least one alternative solution even for large knowledge bases within recommended user interface response times [14]:

- below 100ms, the user feels that the system reacts instantaneously
- 1,000ms is the upper limit for keeping the users thought uninterrupted
- 10,000ms is the upper limit for keeping the user's focus on the dialogue

For the first performance evaluation the goal was to measure the time needed for calculating the corresponding solutions to given requirements. After assigning answers to the questions for the three different knowledge bases, the resulting values are depicted in Table 2. The performance values in Table 2 show that for each of the knowledge bases WEEVIS identifies solutions fast enough to provide instantaneous feedback from the user interface. If no solution could be found due to inconsistencies between the requirements and the knowledge base, Table 3 shows the time needed to identify at least one alternative solution on the basis of one preferred diagnosis, Table 4 shows the time consumption of calculating all possible solutions. WEEVIS is able to calculate either one, two, three or all diagnoses and the corresponding alternative solutions. By taking the response time boundaries for user interfaces into account, the experiment shows that for *small* and *medium* knowledge bases it's possible to calculate all minimal diagnoses within acceptable response times (see Table 3). When it comes to *large* knowledge bases the presented alternative solutions can be reduced to increase the performance of the user interface instead (see Table 4).

	time for identifying solutions
Small	< 1ms
Medium	< 1ms
Large	< 2ms

Table 2. This table shows the time needed to come up with solutions for three knowledge bases. Even for the largest knowledge base the overall time is far below the limit of *an instantaneously reaction*(100ms).

	diagnosis	repair	overall time
	calculation	identification	
Small	< 1ms	< 1ms	< 1ms
Medium	93ms	16ms	109ms
Large	499ms	19ms	518ms

Table 3. This table shows the time needed to come up with at least *one* alternative solution for each of the three knowledge bases. Even for the largest knowledge base the overall time is below the limit of *interrupt a users thought* (1,000ms).

	diagnosis	repair	overall time
	calculation	identification	
Small	97ms	16ms	113ms
Medium	969ms	20ms	989ms
Large	3,028ms	60ms	3,088ms

Table 4. This table shows the time needed to come up with *all* possible alternative solutions for each of the three knowledge bases. For the largest knowledge base the overall time for repair calculation takes about three seconds which is above the recommended time boundaries for *interrupting the user's flow of thought*.

4 Conclusion

In this paper we presented WEEVIS which is an open constraint-based recommendation environment. By exploiting the advantages of Mediawiki, WEEVIS provides an intuitive basis for the development and maintenance of constraintbased recommender applications. The results of our experiment show that due to the integrated direct diagnosis algorithms the WEEVIS user interface provides good the response times for common interactive settings.

References

- A. Felfernig and R. Burke. Constraint-based Recommender Systems: Technologies and Research Issues. In 10th International Conference on Electronic Commerce, page 3. ACM, 2008.
- A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. An Integrated Environment for the Development of Knowledge-based Recommender Applications. *International Journal of Electronic Commerce*, 11(2):11–34, 2006.
- G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing*, *IEEE*, 7(1):76–80, 2003.
- J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, 40(3):77–87, 1997.
- M. Pazzani and D. Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine learning*, 27(3):313–331, 1997.
- R. Reiter. A Theory of Diagnosis From First Principles. Artificial intelligence, 32(1):57–95, 1987.
- U. Junker. QUICKXPLAIN: Preferred Explanations and Relaxations for Overconstrained Problems. In Proceedings of the 19th National Conference on Artifical Intelligence (AAAI'04), pages 167–172, 2004.
- A. Felfernig, M. Schubert, and C. Zehentner. An Efficient Diagnosis Algorithm for Inconsistent Constraint Sets. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 26:53–62, 2012.
- R. Bakker, F. Dikker, F. Tempelman, and P. Wogmim. Diagnosing and Solving Over-determined Constraint Satisfaction Problems. In 13th International Joint Conference on Artificial Intelligence, pages 276–281, Chambery, France, 1993.
- 10. G. Friedrich. Interactive Debugging of Knowledge Bases. In International Workshop on Principles of Diagnosis (DX'14), pages 1–4, Graz, Austria, 2014.
- J. Baumeister, J. Reutelshoefer, and F. Puppe. KnowWE: a Semantic Wiki for Knowledge Engineering. *Applied Intelligence*, 35(3):323–344, 2011.
- A. Mackworth. Consistency in Networks of Relations. Artificial Intelligence, 8(1):99–118, 1977.
- A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner. Consistency-based Diagnosis of Configuration Knowledge Bases. *Artificial Intelligence*, 152(2):213– 234, 2004.
- 14. J. Nielsen. Usability engineering. Elsevier, 1994.

10