

An Integrated Knowledge Engineering Environment for Constraint-based Recommender Systems

Stefan Reiterer¹

Abstract. Constraint-based recommenders support customers in identifying relevant items from complex item assortments. In this paper we present a constraint-based environment already deployed in real-world scenarios that supports knowledge acquisition for recommender applications in a MediaWiki-based context. This technology provides the opportunity to directly integrate informal Wiki content with complementary formalized recommendation knowledge which makes information retrieval for users (readers) easier and less time-consuming. The user interface supports recommender development on the basis of intelligent debugging and redundancy detection. The results of a user study show the need of automated debugging and redundancy detection even for small-sized knowledge bases.

1 Introduction

Constraint-based recommenders support the identification of relevant items from large and often complex assortments on the basis of an explicitly defined set of recommendation rules [3]. Example item domains are digital cameras and financial services [5, 8, 9]. For a long period of time the engineering of recommender knowledge bases (for constraint-based recommenders) required that knowledge engineers are technical experts (in the majority of the cases computer scientists) with the needed technical capabilities [14]. Developments in the field moved one step further and provided graphical engineering environments [5], which improve the accessibility and maintainability of recommender knowledge bases. However, users still have to deal with additional tools and technologies which is in many cases a reason for not applying constraint-based environments.

Similar to the idea of Wikipedia to allow user communities to develop and maintain Wiki pages in a cooperative fashion, we introduce the WEEVIS² environment, which supports the community-based development of constraint-based recommender applications within a Wiki environment. WEEVIS has been implemented on the basis of MediaWiki³, which is an established standard Wiki platform. Compared to other types of recommender systems such as collaborative filtering [19] and content-based filtering [25], constraint-based recommender systems are based on an underlying recommendation knowledge base, i.e., recommendation knowledge is defined explicitly. WEEVIS is already applied by four Austrian universities (within the scope of recommender systems courses) and two companies for the purpose of prototyping recommender applications in the financial services domain.

The user interface of the WEEVIS environment provides intelligent mechanisms that help to make development and maintenance operations easier. Based on model-based diagnosis techniques [12, 17, 26], the environment supports users in the following situations: (1) if no solution could be found for a set of user requirements, the system proposes repair actions that help to find a way out from the "no solution could be found" dilemma; (2) if the constraints in the recommender knowledge base are inconsistent with a set of test cases (situation detected within the scope of regression testing of the knowledge base), those constraints are shown to the users (knowledge engineers) who are responsible for the faulty behavior of the knowledge base; (3) if the recommender knowledge base includes redundant constraints, i.e., constraints that – if removed from the knowledge base – logically follow from the remaining constraints, these constraints are also determined in an automated fashion and shown to knowledge engineers.

The major contributions of this paper are the following. (1) on the basis of a working example from the domain of *financial services*, we provide an overview of the diagnosis and redundancy detection techniques integrated in the WEEVIS environment. (2) we report the results of an empirical study which analyzed the usability of WEEVIS functionalities.

The remainder of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we present an overview of the recommendation environment WEEVIS and discuss the included knowledge engineering support mechanisms. In Section 4 we present results of an empirical study that show the need of intelligent diagnosis and redundancy detection support. In Section 5 we discuss issues for future work, with Section 6 we conclude the paper.

2 Related Work

Based on original static Constraint Satisfaction Problem (CSP) representations [15, 20, 29], many different types of constraint-based knowledge representations have been developed. Mittal and Falkenhainer [22] introduced dynamic constraint satisfaction problems where variables have an activity status and only active variables are taken into account by the search process. Stumptner et al. [28] introduced the concept of generative constraint satisfaction where variables can be generated on demand within the scope of solution search. Compared to existing work, WEEVIS supports the solving of static CSPs on the basis of conjunctive queries where each solution corresponds to a result of querying a relational database. Additionally, WEEVIS includes diagnosis functionalities that help to automatically determine repair proposals in situations where no solution could be found [12].

¹ SelectionArts Intelligent Decision Technologies GmbH, Austria, email:stefan.reiterer@selectionarts.com

² www.weevis.org.

³ www.mediawiki.org.

A graphical recommender development environment for single users is introduced in [5]. This Java-based environment supports the development of constraint-based recommender applications for on-line selling platforms. Compared to Felfernig et al. [5], WEEVIS provides a wiki-based user interface that allows user communities to develop recommender applications. Furthermore, WEEVIS includes efficient diagnosis [12] and redundancy detection [13] mechanisms that allow the support of interactive knowledge base development.

A Semantic Wiki-based approach to knowledge acquisition for collaborative ontology development is introduced in [2]. Compared to Baumeister et al. [2], 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. Furthermore, WEEVIS includes intelligent debugging and redundancy detection mechanisms which make development and maintenance operations more efficient. We want to emphasize that intended redundancies can exist, for example, for the purpose of better understandability of the knowledge base. If such constraints are part of a knowledge base, these should be left out from the redundancy detection process.

A first approach to a conflict-directed search for hitting sets in inconsistent CSP definitions was introduced by Bakker et al. [1]. In this work, minimal sets of faulty constraints in inconsistent CSP definitions were identified on the basis of the concepts of model-based diagnosis [26]. In the line of Bakker et al. [1], Felfernig et al. [4] introduced concepts that allow the exploitation of the concepts of model-based diagnosis in the context of knowledge base testing and debugging. Compared to earlier work [4, 24], WEEVIS provides an environment for development, testing, debugging, and application of recommender systems. With regard to diagnosis techniques, WEEVIS is based on more efficient debugging and redundancy detection techniques that make the environment applicable in interactive settings [12, 16, 21].

3 The WEEVIS Environment

In its current version, WEEVIS supports scenarios where user requirements can be defined in terms of functional requirements [23]. The corresponding recommendations (solutions) are retrieved from a predefined set of alternatives (also denoted as item set or product catalog). Requirements are checked with regard to their consistency with the underlying item set (consistency is given if at least one solution could be identified). If no solution could be found, WEEVIS repair alternatives are determined on the basis of direct diagnosis algorithms [12]. This way, WEEVIS does not only support item selection but also consistency maintenance processes on the basis of intelligent repair mechanisms [6].

WEEVIS is based on the idea that a community of users cooperatively contributes to the development of a recommender knowledge base. The environment supports knowledge acquisition processes on the basis of tags that can be used for defining and testing recommendation knowledge bases. Using WEEVIS, standard Wikipedia pages can be extended with recommendation knowledge that helps to represent domain knowledge in a more accessible and understandable fashion. The same principles used for the developing Wikipedia pages can also be used for the development and maintenance of recommender knowledge bases, i.e., in the *read* mode recommenders can be executed and in the *view source* mode recommendation knowledge can be defined and adapted. This way, rapid prototyping processes can be supported in an intuitive fashion (changes

to the knowledge can be immediately experienced by switching from the *view source* to the *read* mode). In the *read* mode, knowledge bases can as well be tested and in the case of inconsistencies (some test cases were not fulfilled within the scope of regression testing) corresponding diagnoses are shown to the user.

3.1 Overview

The website www.weevis.org provides a selection of different recommender applications (full list, list of most popular recommenders, and recommenders that have been defined previously) that can be tested and extended. Most of these applications have been developed within the scope of university courses on recommender systems (conducted at four Austrian universities). WEEVIS recommenders can be integrated seamlessly into standard Wiki pages, i.e., informally defined knowledge can be complemented or even substituted with formal definitions.

In the following we will present the concepts integrated in the WEEVIS environment on the basis of a working example from the domain of financial services. In such a recommendation scenario, a user has to specify his/her requirements regarding, for example, the expected capital guarantee level of the financial product or the amount of money he or she wants to invest. A corresponding WEEVIS user interface is depicted in Figure 1 where requirements are specified on the left hand side and the corresponding recommendations are displayed in the right hand side.

Each recommendation (item) has a corresponding support value that indicates the share of requirements that are currently supported by the item. A support value of 100% indicates that each requirement is satisfied by the corresponding item. If the support value is below 100%, corresponding repair alternatives are shown to the user, i.e., alternative answers to questions that guarantee the recommendation of at least one item (with 100% support).

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

3.2 WEEVIS Syntax

Constraint-based recommendation requires the explicit definition of questions and possible answers, items and their properties, and constraints (see Figure 2).

In WEEVIS the tag `&QUESTIONS` enumerates the set of user requirements where, for example, *pension* specifies whether the user wants a financial product to support his private pension plan [yes, no] and *maxinvestment* specifies the amount of money the user wants to invest. Furthermore, *payment* represents the frequency in which the payment should be done [once, periodical], *payout* specifies the frequency the customer gets a payout from the financial product (out of [once,monthly]), and *guarantee* the expected capital guarantee [low, high].

An item assortment can be specified in WEEVIS using the `&PRODUCTS` tag (see Figure 2). In our example, the item (product) assortment is specified by values related to the attributes *name*; *guaranteep*, the capital guarantee the product provides; *payoutp*, the payout frequency of the product; *mininvestp* the minimal amount of



Questions	Solutions	Support
<p>pension?</p> <p>yes ✓</p>	 SecureFin	80,00 %
<p>maxinvestment?</p> <p>13500 Euro ✓</p>	 DynamicFin	60,00 %
<p>payment?</p> <p>periodical ✓</p>		
<p>payout?</p> <p>once) monthly ✗</p>		
<p>guarantee?</p> <p>high) low ✗</p>		

Figure 1. A simple financial service recommender (WEEVIS read mode).

money for the financial service. Three items are specified: *SecureFin*, *BonusFin*, and *DynamicFin*.

Incompatibility constraints describe incompatible combinations of requirements. Using the *&INCOMPATIBLE* keyword, we are able to describe an incompatibility between the variables *pension* and *guarantee*. For example, financial services with *low* guarantee must not be recommended to users interested in a product that supports their private pension plan. Filter constraints describe relationships between requirements and items, for example, $maxinvest \geq mininvest$, i.e., the amount of money the user is willing to invest must exceed the minimal payment necessary for the financial product.

In addition the recommendation knowledge base itself, WEEVIS supports the specification of test cases that can be used for the purposes of regression testing (see also Section 3.4). After changes to the knowledge base, regression tests can be triggered by setting the *—show—* tag, that specifies whether the recommender system user interface should show the status of the test case (satisfied or not).

3.3 Recommender Knowledge Base

Recommendation knowledge can be represented as a CSP [20] with the variables V ($V = U \cup P$) and the constraints $C = COMP \cup PROD \cup FILT$ where $u_i \in U$ are variables describing possible user requirements (e.g., *pension*) and $p_i \in P$ are describing item properties (e.g., *payout*). Furthermore, *COMP* represents incompatibility constraints of the form $\neg X \vee \neg Y$, *PROD* the products with their attributes in disjunctive normal form (each product is described as a conjunction of individual product properties), and *FILT* the given filter constraints of the form $X \rightarrow Y$.

The knowledge base specified in Figure 2 can be translated into a corresponding CSP where *&QUESTIONS* represents U , *&PRODUCTS* represents P and *PROD*, and *&CONSTRAINTS* represents

COMP and *FILT*. On the basis of such a definition, WEEVIS is able to calculate recommendations that take into account a specified set of requirements. Such requirements are represented as unary constraints (in our case $R = \{r_1, r_2, \dots, r_k\}$).

If requirements $r_i \in R$ are inconsistent with the constraints in C , we are interested in a subset of these requirements that should be adapted in order 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.

An example of a set of requirements inconsistent with the defined recommendation knowledge is $R = \{r_1 : pension = yes, r_2 : maxinvest = 13500, r_3 : payment = periodical, r_4 : payout = once, r_5 : guarantee = high\}$. The recommendation knowledge base induces two minimal conflict sets (*CS*) [18] in R which are $CS_1 : \{r_1, r_5\}$ and $CS_2 : \{r_4, r_5\}$. For these conflict sets we have two diagnoses: $\Delta_1 : \{r_4, r_5\}$ and $\Delta_2 : \{r_1\}$. The pragmatics, for example, of Δ_1 is that at least r_4 and r_5 have to be adapted in order to be able to find a solution. How to determine such diagnoses on the basis of a HSDAG (hitting set directed acyclic graph) is shown, for example, in [4].

In interactive settings, where diagnoses should be determined in an efficient fashion [12], hitting set based approaches tend to become too inefficient. The reason for this is that conflict sets [18] have to be determined as an input for the diagnosis process. This was the major motivation for developing and integrating FASTDIAG [12] into the WEEVIS environment. Analogous to QUICKXPLAIN [18], this algorithm is based on a divide-and-conquer based approach that en-

```

A very simple "Financial Service Recommender" recommender.
<recommender>
  &PRODUCTS
  {!name!guaranteep!payoutp!#mininvestp!
  |SecureFin|high|once|10000|
  |BonusFin|low|once|3000|
  |DynamicFin|low|monthly|1000|
  }
  &QUESTIONS
  {|pension?(yes, no)|
  |maxinvestment?#(1000,15000,500, Euro)|
  |payment?(once, periodical)|
  |payout? (once, monthly)|
  |guarantee? (high, low)|}
  &CONSTRAINTS
  {
  |pension? = yes &INCOMPATIBLEWITH guarantee? = high|
  |pension? = yes &INCOMPATIBLEWITH payout? = once|
  |maxinvestment? >= #mininvestp|
  |&IF guarantee? = high &THEN guaranteep = high|
  |&IF guarantee? = low &THEN guaranteep = low|
  |&IF guarantee? = high &THEN guaranteep <> low|
  |&IF payout? = once &THEN payoutp = once|
  |&IF payout? = monthly &THEN payoutp = monthly|}
  &TEST
  {|show| pension? = yes, guarantee? = low, payout? = once|}
</recommender>

```

Figure 2. Financial services knowledge base (view source (edit) mode).

ables the determination of minimal diagnoses without the determination of conflict sets. A minimal diagnosis Δ can be used as basis for determining repair actions, i.e., concrete measures to change user requirements in R such that the resulting R' is consistent with C .

3.4 Diagnosis and Repair of Requirements

Definition 2 (Repair Task). Given a set of requirements $R = \{r_1, r_2, \dots, r_k\}$ inconsistent with the constraints in C and a corresponding diagnosis $\Delta \subseteq R$ ($\Delta = \{r_l, \dots, r_o\}$), the corresponding repair task is to determine an adaption $A = \{r'_l, \dots, r'_o\}$ 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 (currently, the first $n=3$ leading diagnoses are determined), the corresponding solution search for $R - \Delta \cup C$ returns a set of alternative repair actions (represented as adaptation A). In the following, all products 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 [12]. This ordering is derived from the order in which a user has entered his/her requirements. For example, if $r_1 : pension = yes$ has been entered before $r_4 : payout = once$ and $r_5 : guarantee = high$ then the underlying assumption is that r_4 and r_5 are of lower importance for the user

and thus have a higher probability of being part of a diagnosis. In our working example $\Delta_1 = \{r_4, r_5\}$. The corresponding repair actions (solutions for $R - \Delta_1 \cup C$) is $A = \{r'_4 : payout = monthly, r'_5 : guarantee = low\}$, i.e., $\{r_1, r_2, r_3, r_4, r_5\} - \{r_4, r_5\} \cup \{r'_4, r'_5\}$ is consistent. The item that satisfies $R - \Delta_1 \cup A$ is $\{DynamicFin\}$ (see in Figure 2). The identified items (p) are ranked according to their support value (see Formula 1).

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

3.5 Regression Testing

WEEVIS supports regression testing processes by the definition and execution of (positive) test cases which specify the intended behavior of the knowledge base. If some of the test cases are not accepted by the knowledge base (are inconsistent with the knowledge base), the causes of this unintended behavior have to be identified. On a formal level a *recommender knowledge base (RKB) diagnosis task* can be defined as follows (see Definition 3).

Definition 3 (RKB Diagnosis Task). Given a set C (recommender knowledge base) and a set $T = \{t_1, t_2, \dots, t_q\}$ of test cases t_i , the diagnosis task is to identify a minimal set Δ of constraints (the diagnosis) that have to be removed from C such that $\forall t_i \in T : C - \Delta \cup \{t_i\}$ is consistent.

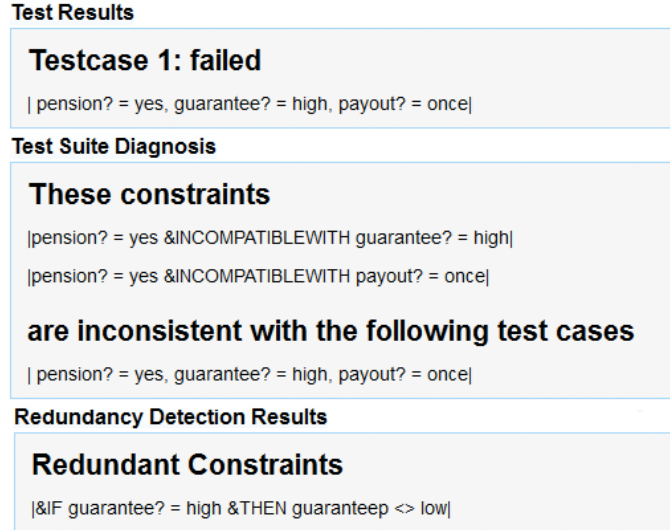


Figure 3. WEEVIS maintenance support: diagnosis and redundancy detection.

An example test case inducing an inconsistency with C is t : $pension = yes$ and $guarantee = high$ and $payout = once$ (see Figure 2). In this context, t induces two conflicts in C which are CS_1 : $\neg(pension = yes \wedge guarantee = high)$ and CS_2 : $\neg(pension = yes \wedge payout = once)$. In order to make C consistent with t , both incompatibility constraints have to be deleted from C , i.e., are part of the diagnosis Δ (see Figure 3).

In contrast to the hitting set based approach [4], WEEVIS includes a FASTDIAG based approach for knowledge base debugging which is more efficient and can therefore be applied in interactive settings [12]. In this context, diagnoses are searched in C (the test cases used for regression testing are assumed to be correct). In the case of requirements diagnosis, the total ordering of the requirements is related to user preferences. In the case of knowledge base diagnosis [4, 16], the ordering is currently derived from the ordering of the constraints in the knowledge base.

3.6 Identifying Redundancies

To support users in identifying redundant constraints in recommender knowledge bases, the COREDIAG [13] algorithm has been integrated into the WEEVIS environment. COREDIAG relies on QUICKXPLAIN [18] and is used for the determination of minimal cores (minimal non-redundant constraint sets). On a formal level a *recommendation knowledge base (RKB) redundancy detection task* can be defined as follows (see Definition 4).

Definition 4 (RKB Redundancy Detection Task). Let c_a be a constraint of C (the recommendation knowledge base) and \overline{C} the logical negation (the complement or inversion) of C . Redundancy can be analyzed by checking $C - \{c_a\} \cup \overline{C}$ for consistency - if consistency is given, c_a is non-redundant. If this condition is not fulfilled, c_a is said to be redundant. By iterating over each constraint of C , executing the non-redundancy check $C - \{c_a\} \cup \overline{C}$, and deleting redundant constraints from C results in a set of non-redundant constraints (the minimal core).

As an example, the knowledge base shown in Figure 2 contains

redundancies. Consequently, the corresponding set of constraints C does not represent a minimal core. Taking a closer look at the knowledge base it appears that two individual filter constraints are redundant with each other. More precisely, either the constraint *&IF guarantee? = high &THEN guaranteep = high* or the constraint *&IF guarantee? = high &THEN guaranteep <> low* can be removed from the knowledge base (in our example, the latter is proposed as redundant by COREDIAG – see Figure 3). In the general case, higher cardinality constraint sets can be removed, not only cardinality-1 sets as in our example [13].

Similar to the diagnosis of inconsistent requirements the COREDIAG algorithm is based on the principle of divide-and-conquer: whenever a set S which is a subset of C is inconsistent with \overline{C} , it is or contains a minimal core, i.e., a set of constraints which preserve the semantics of C . COREDIAG is based on the principle of QUICKXPLAIN [18]. As a consequence a minimal core (minimal set of constraints that preserve the semantics of C) can be interpreted as a minimal conflict, i.e., a minimal set of constraints that are inconsistent with \overline{C} . Based on the assumption of a strict lexicographical ordering [12] of the constraints in C , COREDIAG determines preferred minimal cores.

4 Empirical Study

4.1 Study Design

We conducted an experiment to highlight potential reductions of development and maintenance efforts facilitated by the WEEVIS debugging and redundancy detection support. For this study we defined four knowledge bases that differed with regard to the number of constraints, variables, faulty constraints, and redundancies (see Table 1). Based on these example knowledge bases, the participants had to find solutions for the following two types of tasks:

1. *Diagnosis task:* The participants had to answer the question which minimal set Δ of faulty constraints has to be removed from C ($C = COMP \cup FILT$) such that there exists at least one solution for $((C - \Delta) \cup PROD)$.

2. *Redundancy detection task*: The participants had to answer the question which constraints in $C = COMP \cup FILT$ are redundant (if $C - \{c_a\} \cup \bar{C}$ is inconsistent then the constraint c_a is redundant).

knowledge base	number of constraints /variables /faulty constraints /test cases /redundancies
kb_1 (redundant)	5/5/0/0/2
kb_2 (inconsistent)	5/5/1/2/0
kb_3 (redundant)	10/10/0/0/4
kb_4 (inconsistent)	10/10/2/4/0

Table 1. Knowledge bases used in the empirical study.

The participants (subjects $N=20$) of our experiment were separated into two groups (groups A and B). All subjects were students of Computer Science (20% female, 80% male) who successfully completed a course on constraint technologies and recommender systems. Each subject had to complete the assigned tasks on his/her own on a sheet of paper and they had to track the time for each task. In our experiment we randomly assigned the participants to one of the two test groups shown in Table 2. This way we were able to compare the time efforts of identifying faulty constraints and redundancies in knowledge bases as well as to estimate error rates related to the given tasks.

testgroup	1 st knowledge base	2 nd knowledge base
A ($n = 10$)	kb_1 (redundancy detection)	kb_4 (diagnosis)
B ($n = 10$)	kb_2 (diagnosis)	kb_3 (redundancy detection)

Table 2. Each subject had to complete one diagnosis and one redundancy detection task. Members of group A had a redundancy detection task of lower complexity and a higher complexity diagnosis detection task (randomized order). Vice-versa members of group B had to solve a higher complexity redundancy detection and a lower complexity diagnosis task.

4.2 Study Results

The *first goal* of our experiment was to analyze time efforts and error rates related to the identification of faulty constraints in recommender knowledge bases. The first hypothesis tested in our experiment was the following:

Hypothesis 1: Even low-complexity knowledge bases trigger the identification of faulty diagnoses (note that all knowledge bases used in the experiment can be interpreted as low-complexity knowledge bases [13]).

The average time effort for identifying minimal diagnoses in knowledge base kb_2 was 281.3 seconds, the average time needed to identify diagnoses in kb_4 was 497.5 seconds. The results show a significantly higher error rate when the participants had to identify the faulty constraints in the more complex knowledge base (see Table 3). Hypothesis 1 can be confirmed by the results in Table 3 that show that even simple knowledge bases trigger high error rates and increasing time efforts. With the automated diagnosis detection mechanisms integrated in WEEVIS, reductions of related error rates and time efforts can be expected.

	groupB (kb_2)	groupA (kb_4)
average time (sec.)	281.3	497.5
correct (%)	50.0	10.0
incorrect (%)	50.0	90.0

Table 3. Time efforts and error rates related to the completion of diagnosis tasks.

The *second goal* of our experiment was to analyze time efforts and error rates related to the identification of redundant constraints in recommender knowledge bases. The second hypothesis tested in our experiment was the following:

Hypothesis 2: Even low-complexity knowledge bases trigger the faulty identification of redundant constraints.

The average time for identifying redundant constraints in knowledge base kb_1 was 189.2 seconds, for kb_3 337.4 seconds were needed. The results show a significantly higher error rate when the participants had to identify redundant constraints in the more complex knowledge base (see Table 4). Hypothesis 2 can be confirmed since even for low complexity knowledge bases error rates related to redundancy detection tasks are high. With the automated redundancy detection mechanisms integrated in WEEVIS, reductions of related error rates and time efforts can be expected.

	groupA (kb_1)	groupB (kb_3)
average time (sec.)	189.2	337.4
correct (%)	40.0	0.0
incorrect (%)	60.0	100.0

Table 4. Time efforts and error rates related to the completion of redundancy detection tasks.

5 Future Work

There are a couple of issues for future work. The current WEEVIS version does not include functionalities that allow the learning/prediction of user preferences. The importance of individual user requirements is based on the assumption that the earlier a requirement has been specified the more important it is. In future versions we want to make the modeling of preferences more intelligent by integrating, for example, learning mechanisms that derive requirements importance distributions on the basis of analyzing already completed recommendation sessions.

Diagnoses and redundancies are currently implemented on the level of constraints, i.e., intra-constraint diagnoses and redundancies are not supported. In future WEEVIS versions we want to integrate fine-granular analysis methods that will help to make analysis and repair of constraints even more efficient. A major research challenge in this context is to integrate intelligent mechanisms for diagnosis discrimination [27] since in many scenarios quite a huge number of alternative diagnoses exists. In such scenarios it is important for knowledge engineers to receive recommendations of diagnoses that are reasonable. This challenge has already been tackled in the context of diagnosing inconsistent user requirements (see, e.g., [6]), however, heuristics with high prediction quality for knowledge bases have not been developed up to now [10, 11].

A major issue for future work is to integrate alternative mechanisms for knowledge base development and maintenance. The knowledge engineer centered approach to knowledge base construction leads to scalability problems in the long run, i.e., knowledge engineers are not able to keep up with the speed of knowledge base related change and extension requests. An alternative approach to knowledge base development and maintenance is the inclusion of concepts of Human Computation [7, 30] which allow a more deep integration of domain experts into knowledge engineering processes on the basis of simple micro tasks. Resulting micro contributions can be automatically integrated into constraints part of the recommendation knowledge base.

Finally, we are interested in a better understanding of the key factors that make knowledge bases understandable. More insights and answers related to this question will help us to better identify problematic areas in a knowledge base which could cause maintenance efforts above average. A first step in this context will be to analyze existing practices in knowledge base development and maintenance with the goal to figure out major reasons for the knowledge acquisition bottleneck and how this can be avoided in the future.

6 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 constraint-based recommender applications. WEEVIS is already applied by four Austrian universities within the scope of recommender systems courses and also applied by companies for the purpose of prototyping recommender applications. The results of our empirical study indicate the potential of reductions of error rates and time efforts related to diagnosis and redundancy detection. In industrial scenarios, WEEVIS can improve the quality of knowledge representations, for example, documentations can at least partially be formalized which makes knowledge more accessible – instead of reading a complete documentation, the required knowledge chunks can be identified easier.

REFERENCES

- [1] 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*, pp. 276–281, Chambéry, France, (1993).
- [2] J. Baumeister, J. Reutelschöfer, and F. Puppe, ‘KnowWE: a Semantic Wiki for Knowledge Engineering’, *Applied Intelligence*, **35**(3), 323–344, (2011).
- [3] A. Felfernig and R. Burke, ‘Constraint-based Recommender Systems: Technologies and Research Issues’, in *10th International Conference on Electronic Commerce*, p. 3. ACM, (2008).
- [4] A. Felfernig, G. Friedrich, D. Jannach, and M. Stumptner, ‘Consistency-based Diagnosis of Configuration Knowledge Bases’, *Artificial Intelligence*, **152**(2), 213–234, (2004).
- [5] 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).
- [6] A. Felfernig, G. Friedrich, M. Schubert, M. Mandl, M. Mairitsch, and E. Teppan, ‘Plausible Repairs for Inconsistent Requirements’, in *IJCAI*, volume 9, pp. 791–796, (2009).
- [7] A. Felfernig, S. Haas, G. Ninaus, M. Schwarz, T. Ulz, M. Stettinger, K. Isak, M. Jeran, and S. Reiterer, ‘RecTurk: Constraint-based Recommendation based on Human Computation’, in *RecSys 2014 CrowdRec Workshop*, pp. 1–6, Foster City, CA, USA, (2014).
- [8] A. Felfernig, K. Isak, K. Szabo, and P. Zachar, ‘The VITA Financial Services Sales Support Environment’, pp. 1692–1699, Vancouver, Canada, (2007).
- [9] A. Felfernig and A. Kiener, ‘Knowledge-based Interactive Selling of Financial Services with FSAdvisor’, in *17th Innovative Applications of Artificial Intelligence Conference (IAAI05)*, pp. 1475–1482, Pittsburgh, Pennsylvania, (2005).
- [10] A. Felfernig, S. Reiterer, M. Stettinger, and J. Tiihonen, ‘Intelligent Techniques for Configuration Knowledge Evolution’, in *VAMOS Workshop 2015*, pp. 51–60, Hildesheim, Germany, (2015).
- [11] A. Felfernig, S. Reiterer, M. Stettinger, and J. Tiihonen, ‘Towards Understanding Cognitive Aspects of Configuration Knowledge Formalization’, in *VAMOS Workshop 2015*, pp. 117–124, Hildesheim, Germany, (2015).
- [12] 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).
- [13] A. Felfernig, C. Zehentner, and P. Blazek, ‘COREDIAG: Eliminating Redundancy in Constraint Sets’, *International Workshop on Principles of Diagnosis (DX’11)*, 219–224, (2011).
- [14] G. Fleischanderl, G. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner, ‘Configuring Large Systems Using Generative Constraint Satisfaction’, *IEEE Intelligent Systems*, **13**(4), 59–68, (1998).
- [15] E. Freuder, ‘In Pursuit of the Holy Grail’, *Constraints*, **2**(1), 57–61, (1997).
- [16] G. Friedrich, ‘Interactive Debugging of Knowledge Bases’, in *International Workshop on Principles of Diagnosis (DX’14)*, pp. 1–4, Graz, Austria, (2014).
- [17] Russell Greiner, Barbara A. Smith, and Ralph W. Wilkerson, ‘A Correction to the Algorithm in Reiter’s Theory of Diagnosis’, *Artificial Intelligence*, **41**(1), 79–88, (1989).
- [18] U. Junker, ‘QUICKXPLAIN: Preferred Explanations and Relaxations for Over-constrained Problems’, in *AAAI*, volume 4, pp. 167–172, (2004).
- [19] 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).
- [20] A. Mackworth, ‘Consistency in Networks of Relations’, *Artificial Intelligence*, **8**(1), 99–118, (1977).
- [21] J. Marques-Silva, F. Heras, M. Janota, A. Previtto, and A. Belov, ‘On Computing Minimal Correction Subsets’, in *IJCAI 2013*, pp. 615–622, Peking, China, (2013).
- [22] S. Mittal and B. Falkenhainer, ‘Dynamic Constraint Satisfaction’, in *National Conference on Artificial Intelligence*, pp. 25–32, (1990).
- [23] S. Mittal and F. Frayman, ‘Towards a Generic Model of Configuration Tasks’, in *IJCAI*, volume 89, pp. 1395–1401, (1989).
- [24] B. O’Sullivan, A. Papadopoulos, B. Faltings, and P. Pu, ‘Representative Explanations for Over-constrained Problems’, in *Twenty-Second AAAI Conference on Artificial Intelligence (AAAI-07)*, eds., R. Holte and A. Howe, pp. 323–328, Vancouver, Canada, (2007). AAAI Press.
- [25] M. Pazzani and D. Billsus, ‘Learning and Revising User Profiles: The Identification of Interesting Web Sites’, *Machine learning*, **27**(3), 313–331, (1997).
- [26] R. Reiter, ‘A Theory of Diagnosis From First Principles’, *Artificial intelligence*, **32**(1), 57–95, (1987).
- [27] K. Shchekotykhin and G. Friedrich, ‘Diagnosis discrimination for ontology debugging’, in *ECAI 2010*, pp. 991–992, (2010).
- [28] M. Stumptner, G. Friedrich, and A. Haselböck, ‘Generative Constraint-based Configuration of Large Technical Systems’, *AI EDAM*, **12**(04), 307–320, (1998).
- [29] E. Tsang, *Foundations of Constraint Satisfaction*, volume 289, Academic press London, 1993.
- [30] L. VonAhn, ‘Human Computation’, in *Technical Report CM-CS-05-193*, (2005).