Intelligent Techniques for Configuration Knowledge Evolution

Alexander Felfernig
Institute for Software Technology, TU Graz
Inffeldgasse 16b/II
A-8010, Graz, Austria
felfernig@ist.tugraz.at

Stefan Reiterer
Institute for Software Technology, TU Graz
Inffeldgasse 16b/II
A-8010, Graz, Austria
reiterer@ist.tugraz.at

Martin Stettinger
Institute for Software Technology, TU Graz
Inffeldgasse 16b/II
A-8010, Graz, Austria
stettinger@ist.tugraz.at

Juha Tiihonen
Department of Computer Science, University of Helsinki
Gustaf Hällströmin katu 2b
FI-00014, Helsinki, Finland
juha.tiihonen@cs.helsinki.fi

ABSTRACT

Automated testing and debugging of knowledge bases (such as configuration knowledge bases and feature models) is an important contribution to manage knowledge evolution efficiently. However, existing approaches rely on the assumption of consistent test suites which are always kept up-to-date within the scope of different knowledge base maintenance cycles. In this paper we introduce diagnosis techniques that actively guide stakeholders (knowledge engineers and domain experts) in the process of testing and debugging knowledge bases. These techniques take into account faulty test cases and constraints and recommend diagnoses which are the source of a given inconsistency.

Categories and Subject Descriptors

D.2 [**SOFTWARE ENGINEERING**]: Testing and Debugging —*Debugging aids, Diagnostics*

General Terms

Algorithms, Theory

Keywords

Configuration, Feature Models, Automated Debugging

1. INTRODUCTION

The task of knowledge evolution [1] is to incorporate new knowledge into an existing knowledge base (including the corresponding test suite).

ACM 978-1-4503-3273-6/15/01... $15.00 .
http://dx.doi.org/10.1145/2701319.2701320

Knowledge evolution has to tackle a number of challenges as there are consistency preservation, change minimality, and locality of changes [2]. Knowledge evolution is an error-prone process often accompanied by remarkable efforts related to testing and debugging new versions of a knowledge base – see, for example, [3, 4, 5, 6]. In order to tackle this challenge, a couple of approaches have been developed: (1) Model-based knowledge representations such as description logics [7], SAT [8], constraint satisfaction problems [9], and logic-based approaches [10] help to clearly separate domain knowledge from problem solving knowledge. This allows to focus development and maintenance operations and avoids the need of adapting problem solving knowledge as a result of changes in the domain knowledge and vice-versa. (2) Graphical development environments support a compact knowledge representation and thus increase understandability and maintainability [11, 12, 13]. (3) Automated testing and debugging helps to more easily identify erroneous elements (e.g., constraints) in knowledge bases and thus helps to further increase the efficiency of knowledge base development [4].

One of the still missing links to more efficient knowledge base testing and debugging is a guidance of the user when performing such operations. For example, after executing a couple of maintenance operations on the knowledge base and adapting (including additional) test cases, it becomes unclear where erroneous definitions are located – in the new version of the knowledge base, in the new or adapted set of test cases, or in both. In this paper we introduce an approach to the automated identification of erroneous elements in the knowledge base (feature model or configuration knowledge base) and the associated test suite.

The major contributions of this paper are the following. (1) We extend existing approaches that rely on the assumption of consistent and validated test cases (see, e.g., [4]) with a diagnosis approach that supports the determination of preferred diagnoses which can be located in the knowledge base itself but as well in the set of test cases. (2) In order to systematically identify the aforementioned diagnoses we exploit the domain knowledge of knowledge engineers and domain experts (the community) and thus are able to determine preferred diagnoses, i.e., diagnoses with a high probability of representing the true source of observed faulty behavior.

The remainder of this paper is organized as follows. In Section 2 we introduce an example configuration knowledge base from the domain of financial services. Thereafter we introduce diagnosis approaches that assist knowledge engineers in knowledge base evolution (Section 3). Threats to validity are discussed in Section 4. A discussion of related and future work is provided in Section 5. The paper is concluded with Section 6.

2. WORKING EXAMPLE: TESTING & DEBUGGING

The diagnosis approaches presented in this paper are applicable to different types of knowledge representations such as description logics [7], SAT [8], constraint satisfaction [9], logic-based approaches [10] and feature models [14, 15, 16]. In the context of knowledge base development, diagnosis approaches help to automatically identify the sources of inconsistencies (faulty rules/constraints) in knowledge bases. For demonstration purposes we focus on a constraint-based knowledge representation that can be derived, for example, directly from a feature model [17].

Since we exploit constraint technologies, we have to introduce a set of *variables* (V=$\{v_1, v_2, ..., v_n\}$), the corresponding *variable domains* (D=$\{dom(v_1), dom(v_2), ..., dom(v_n)\}$), and a set of *constraints* (C=$\{c_1, c_2, ..., c_m\}$). The triple (V, D, C) represents a constraint satisfaction problem (CSP) [9] which can be defined as follows.

Definition (Constraint Satisfaction Problem and Solution – CSP). A CSP is a triple (V, D, C) where V represents a set of variables (V=$\{v_1, v_2, ..., v_n\}$), D describes the corresponding variable domains (D=$\{dom(v_1), dom(v_2), ..., dom(v_n)\}$), and C is a set of constraints (C=$\{c_1, c_2, ..., c_m\}$). A solution for a given CSP is represented by a complete set of variable assignments which is consistent with the set of constraints.

For each CSP we assume the existence of a set of *test cases* T = $\{t_1, t_2, ..., t_k\}$ (the test suite). The set T = $\{t_1, t_2, ..., t_k\}$ represents a set of (positive) test cases (specify the intended behavior of the knowledge base) which can be exploited for the purpose of regression testing.[1] A knowledge base accepts a given set of test cases if it accepts each individual test case, i.e., there exists (existential quantification for test cases) at least one solution for (V, D, C $\cup$ $\{t_i\}$) for each $t_i \in$ T.

The following configuration knowledge base supports the construction of user preferences with regard to a set of financial services. In this example, wr represents the *willingness to take risks*, ip represents the *intended investment period*, and rr represents the *expected return rate*.

$V = \{wr, ip, rr\}$

$C = \{c_1, c_2, c_3, c_4, c_5\}$
$T = \{t_1, t_2, t_3, t_4\}$
$dom(wr) = \{low, medium, high\}$
$dom(ip) = \{shortterm, mediumterm, longterm\}$
$dom(rr) = \{3-6\%, 6-9\%, >9\%\}$
$c_1 : wr = medium \rightarrow ip \neq shortterm$
$c_2 : wr = high \rightarrow ip = longterm$
$c_3 : ip = longterm \rightarrow (rr = 3-6\% \vee rr = 6-9\%)$
$c_4 : rr = \ >9\% \rightarrow wr = high$
$c_5 : rr = 6-9\% \rightarrow (wr \neq low \wedge wr \neq medium)$

$t_1 : wr = high \wedge rr = \ >9\%$
$t_2 : rr = 6-9\% \wedge wr = medium$
$t_3 : ip = shortterm \wedge wr = medium$
$t_4 : wr = high \wedge ip = mediumterm$

If there exists at least one test case t_i in T that is inconsistent with the constraints in C, i.e., inconsistent(C $\cup$ $\{t_i\}$) then t_i induces a conflict in C (*assuming that C itself is consistent*).[2] A definition of the notion of a *conflict set* is the following.

Definition (Conflict Set – CS). A conflict set is a set of constraints $CS \subseteq C \cup \{t_i\}$ which is inconsistent ($t_i \in T$). A conflict set CS is minimal, if there does not exist a conflict CS' with $CS' \subset CS$.

The conflict sets of our example knowledge base are $CS_1 : \{c_2, c_3, t_1\}$, $CS_2 : \{c_5, t_2\}$, $CS_3 : \{c_1, t_3\}$, and $CS_4 : \{c_2, t_4\}$. These conflict sets are also minimal since $\nexists CS'_i$ with $CS'_i \subset CS_i$. Note that this is a new way to define conflict sets compared to the original work of Felfernig et al. [4] since in our case, test cases are regarded as first class entities of the diagnosis process. This view allows us to consider test cases as faulty elements in the diagnosis process. Note that conflicts do not have to include elements from both, C and T; they can be composed solely of constraints in C or *individual test cases* in T. Since test cases are existentially quantified, only single test cases can be part of a conflict. In the following we assume that the constraints in C are consistent, i.e., conflicts are induced by individual test cases.

The complete set of conflict sets for a given C and a corresponding set of test cases T can be determined on the basis of conflict detection algorithms such as QuickXPlain [19] in combination with a corresponding hitting set (diagnosis) algorithm as presented in [18]. For example, if $C \cup \{t_i\}$ is inconsistent, QuickXPlain can be activated with the set $C \cup \{t_i\}$. The problem of determining a diagnosis from conflict sets can be defined as follows.

Definition (Diagnosis Problem). A diagnosis problem is defined by a tuple (V, D, C, T) where V is a set of variables, D represents the corresponding variable domains, C is a set of constraints, and T represents a set of test cases.

On the basis of this definition of a diagnosis problem, we introduce the definition of a corresponding diagnosis.

Definition (Diagnosis). A diagnosis for a given diagnosis problem defined by a tuple (V, D, C, T) is a set $\Delta \subseteq C \cup T$ such that $\forall t_i \in T - \Delta$: consistent(C - $\Delta \cup \{t_i\}$) . A diagnosis

[1] The inclusion of negative test cases is easy but omitted here for reasons of readability – for details see [4].

[2] Note that the concepts discussed in this paper can also be applied if the knowledge base itself is inconsistent.

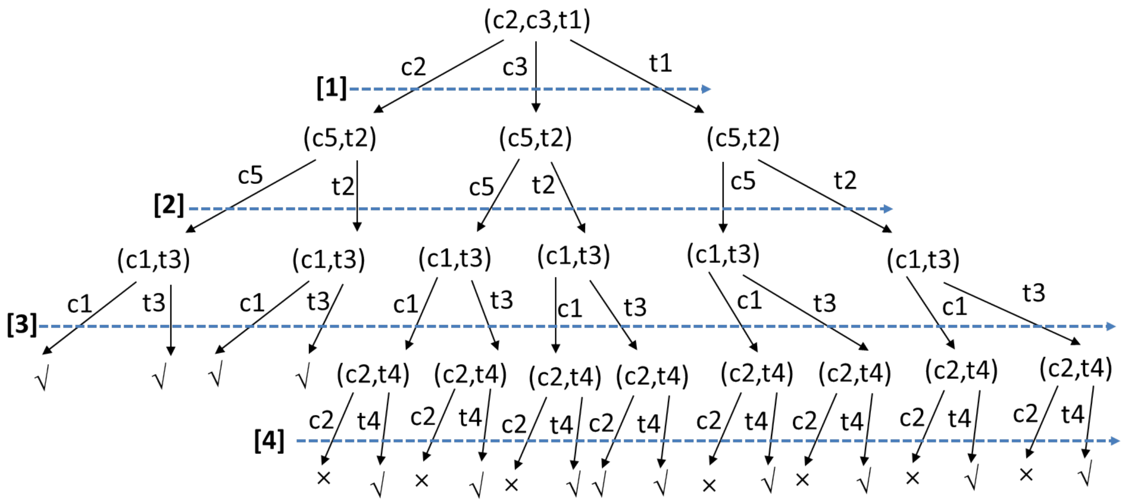

Figure 1: Hitting Set Directed Acyclic Graph (HSDAG) [18] for the example configuration knowledge base (and the corresponding test cases) – expansion strategy = breadth-first (BF). In this context, $\surd$ indicates that a diagnosis has been found (the first identified diagnosis is $\Delta_1 : \{c_1, c_2, c_5\}$ which represents the path from the root to the first leaf denoted with $\surd$.) and $\times$ indicates a search state which can be closed since no solutions (minimal diagnoses) will be found when further expanding the path.

Δ is minimal if $\nexists\ \Delta'$ with $\Delta' \subset \Delta$.

A diagnosis can be regarded as a recommendation of a set of knowledge base elements (in our case constraints) that represent a source of an inconsistency, i.e., should be analyzed within the scope of *testing and debugging operations*. In contrast to the original work of Felfernig et al. [4], this definition allows the inclusion of test cases as first class entities into the diagnosis process, i.e., test cases can be elements of a diagnosis. Such situations especially occur if test cases are faulty or outdated. The diagnoses which can be derived from the four identified conflict sets are depicted in Figure 1. In this example, diagnoses are determined in breadth-first fashion – further alternatives will be discussed in Section 3.

The identified diagnoses are $\Delta_1 : \{c_1, c_2, c_5\}$, $\Delta_2 : \{c_2, c_5, t_3\}$, $\Delta_3 : \{c_1, c_2, t_2\}$, $\Delta_4 : \{c_2, t_2, t_3\}$, $\Delta_5 : \{c_1, c_3, c_5, t_4\}$, $\Delta_6 : \{c_3, c_5, t_3, t_4\}$, $\Delta_7 : \{c_1, c_3, t_2, t_4\}$, $\Delta_8 : \{c_2, c_3, t_2, t_3\}$, $\Delta_9 : \{c_3, t_2, t_3, t_4\}$, $\Delta_{10} : \{c_1, c_5, t_1, t_4\}$, $\Delta_{11} : \{c_5, t_1, t_3, t_4\}$, $\Delta_{12} : \{c_1, t_1, t_2, t_4\}$, $\Delta_{13} : \{t_1, t_2, t_3, t_4\}$.

Calculating Diagnoses. Diagnoses that take into account both, test cases *and* constraints, can be determined using the hitting set directed acyclic graph (HSDAG) approach [18] where diagnosis search is performed in a breadth-first fashion. Each node in the HSDAG represents a minimal conflict set which can be determined using conflict detection algorithms such as QuickXPlain [19].

In contrast to existing approaches to knowledge base debugging [4], *individual* test cases are as well allowed to be part of a diagnosis. As a consequence, the check whether a diagnosis has been identified has to be performed by evaluating the condition $\forall t_i \in T - H(n) : consistent(C - H(n) \cup t_i)$. *H(n)* represents the set of constraints and test cases that have been selected as diagnosis elements in the current HSDAG search path [18]. If the condition is fulfilled, a diagnosis has been identified, otherwise further conflicts exist and have to be resolved by expanding the HSDAG. For an algorithmic description of HSDAG search practices we refer to [4, 6, 18].

Typically, there are quite a number of alternative diagnoses. In such a situation, knowledge engineers are interested in receiving recommendations that include the most relevant ones. In the following section we introduce basic methods that can help to determine relevant diagnoses.[3]

3. RECOMMENDING TESTING & DEBUGGING OPERATIONS

The following criteria for recommending testing and debugging operations are based on our experiences in configuration model development. We are aware of the fact that in-depth user studies are needed in order to figure out which combinations of these (and further) criteria provide the best recommendations - such studies are within the scope of our future work.

Breadth-First (BF). BF search (BFS) allows to determine minimal-cardinality diagnoses, i.e., diagnoses with the lowest possible number of elements. In our example, there are four diagnoses of cardinality 3 (see Figure 1). Cardinality-based ranking of diagnoses relies on the assumption that the lower the cardinality, the higher the probability that the diagnosis elements are responsible for the observed misbehavior. This approach allows to focus on a small number of constraints (and test cases) that are possibly responsible for the faulty behavior. The relevance of diagnoses determined by BFS can be characterized by Formula 1. The result of applying Formula 1 is depicted in Table 1.

[3]Relevant diagnoses are also denoted as leading diagnoses – see, for example [20].

diagnosis	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7	Δ_8	Δ_9	Δ_{10}	Δ_{11}	Δ_{12}	Δ_{13}
relevance	**0.33**	0.33	0.33	0.33	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
ranking	**1**	2	3	4	5	6	7	8	9	10	11	12	13

Table 1: Relevance (and ranking) of diagnoses based on *breadth-first search*. The higher the relevance of a diagnosis, the more it is recommended to analyze (and adapt) the included constraints and test cases.

$$relevance(\Delta) = \frac{1}{cardinality(\Delta)} \tag{1}$$

However, the assumptions of BF search do not always hold and there exist further relevant aspects that should be taken into account when ranking (recommending) diagnoses, i.e., determining leading diagnoses [20, 21]. Some of these aspects will be discussed in the following. Note that we do not recommend to use specific heuristics to determine preferred diagnoses but show how to combine such heuristics within the scope of ensemble-based diagnosis (ENS).

Actuality-guided Diagnosis (ACT). Knowledge actuality plays a crucial role in the context of many knowledge-based systems, for example, configuration knowledge is subject of frequent change operations resulting from changes in the component information, technical constraints, and restrictions stemming from marketing and sales. In this context, new components are inserted and older components and constraints sometimes become outdated [22].

Information about knowledge growing old can also be exploited for determining a ranking for candidate diagnoses, i.e., the older a constraint (measured in terms of the time elapsed since a constraint has been maintained by a knowledge engineer), the higher is the probability that the constraint is irrelevant or not up to date.[4] An example of how to estimate the actuality of a knowledge base element x is the time since the last user access – see Formula 2.

$$actuality(x) = \frac{1}{time\ since\ last\ user\ access\ (x) + 1} \tag{2}$$

More sophisticated approaches to estimate actuality are to take into account how often a constraint was activated during the search for a solution (e.g., a constraint in implicative form can remain inactive if its precondition is not fulfilled) or how often a constraint contributed to the induction of an inconsistency in the user requirements. For example, the two requirements $ip = shortterm$ and $wr = medium$ can not be satisfied – in this case, constraint c_1 induces an inconsistency in the given set of requirements. Taking into account both aspects (activations and inconsistencies), results in the estimation of Formula 3.

$$actuality(x) = \left(1 - \frac{\#inconsistencies(x)}{\#activations(x)}\right) \tag{3}$$

By exploiting the results of this formula, we are also able to determine the relevance of a diagnosis Δ (see Formula 4).

[4]We are aware of the fact that this assumption might not hold in general – for a related discussion we refer to [22].

$$relevance(\Delta) = \frac{1}{\Sigma_{x\in\Delta} actuality(x)} \tag{4}$$

An example of actuality-based diagnosis ranking (based on time since last user access) is shown Tables 2 and 3. The corresponding search graph is depicted in Figure 2. The [values] are representing the relevance values of the H(n) expansions – following a best-first search regime, the node with the highest relevance factor is expanded next.[5]

Rating-guided Diagnosis (RATE). Both, breadth first (BF) and actuality-based diagnosis (ACT) approaches are based on the observation of the properties of individual elements of a knowledge base. An important additional aspect to be taken into account is the feedback of individual stakeholders (s_i), i.e., how stakeholders rate the quality of knowledge base elements. Quality ratings for these elements can be provided in different ways, for example, in terms of a one-dimensional rating on a scale of 1..5. An example of determining a quality rating for knowledge base elements is depicted in Table 4.

On the basis of a given quality rating of the individual knowledge base elements, we are able to determine the relevance of a diagnosis (see Formula 5).

$$relevance(\Delta) = \frac{1}{\Sigma_{x\in\Delta} rating(x)} \tag{5}$$

The result of applying Formula 5 to our example is depicted in Table 5.

Utility-guided Diagnosis (UTILITY). Another alternative is to rate the individual knowledge base elements with regard to different knowledge engineering relevant *interest dimensions* such as *maintainability, understandability, efficiency in search, redundancy degree, and correctness.* Given ratings (provided by knowledge engineers) for each knowledge base element for each of the specific interest dimensions, we can derive an overall utility for a specific knowledge base element (constraint or test case) and on the basis of this utility can determine the overall relevance of the related diagnosis (see Formula 7). The lower the utility of the individual knowledge base elements, the higher is the relevance of the corresponding diagnosis, i.e., the higher is the probability for the diagnosis to be a candidate indicating relevant changes in the knowledge base. The utility of a specific knowledge base element x (constraint or test case) can be determined, for example, on the basis of Formula 6.

[5]For reasons of space limitations we do not depict the search trees of the following diagnosis ranking heuristics, however, the search approach remains the same: best-first node expansion strategy = expansion of node H(n) with highest *relevance*.

constraints and test cases	c_1	c_2	c_3	c_4	c_5	t_1	t_2	t_3	t_4
time since last access	12.0	100.1	14.2	370.9	1.1	6.2	1.2	19.2	6.5
actuality (time)	1.07	1.009	1.06	1.002	1.47	1.13	1.45	1.049	1.13

Table 2: Example times (e.g., days) since last access and related actuality values.

diagnosis	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7	Δ_8	Δ_9	Δ_{10}	Δ_{11}	Δ_{12}	Δ_{13}
relevance	0.282	0.2834	0.2833	**0.285**	0.211	0.212	0.212	0.219	0.213	0.208	0.2092	0.209	0.21
ranking	4	3	2	**1**	10	8	6	5	7	13	12	11	9

Table 3: Relevance (and corresponding ranking) of diagnoses based on *actuality*.

$$utility(x) = \frac{\Sigma_{d \in Dimensions} val(x,d)}{|Dimensions|} \tag{6}$$

$$relevance(\Delta) = \frac{1}{\Sigma_{x \in \Delta} utility(x)} \tag{7}$$

An example of determining the relevance of diagnoses based on the utility-based approach is given in Tables 6 and 7.

Ensemble-guided Diagnosis (ENS). The diagnosis prediction mechanisms discussed up to now are based on a single hypothesis. The basic idea of ensemble-based diagnosis is to derive diagnosis relevance by exploiting the individual predictions (a set of hypotheses) of the aforementioned approaches. There are different possibilities of integrating individual predictions – one such approach will be introduced now. For each of the first-n ranks (in our case 1..6) we assign a corresponding rank utility (see Table 8).

For the ensemble-based ranking of individual diagnoses, we apply Formula 8.

$$relevance(\Delta) = \Sigma_{r \in R} occurrences(\Delta, r) \times rankutility(r) \tag{8}$$

In Formula 8, R denotes the set of rank positions (in our case 1..6), *occurrences(Δ, r)* denotes the number of times Δ has the rank prediction r, and *rankutility(r)* denotes the utility of rank r (see Table 8).

Table 9 summarizes the individual predictions determined by the methods BF, ACT, RATE, and UTILITY. Based on the information shown in Table 9, we are able to determine the ensemble-based relevance of diagnoses. The result of applying Formula 8 is shown in Table 10. Note that the ensemble-based approach is open to the inclusion of additional diagnosis ranking heuristics. Their development is seen as an issue of future work. Further aspects that can be taken into account when ranking diagnoses are, for example, well-formedness rules [23, 24] and knowledge about the organization principles of knowledge [25].

4. THREATS TO VALIDITY

In this paper we focus on introducing an approach that allows to include ranking criteria for knowledge base diagnoses. These criteria are basic ones that have been defined on the basis of our experiences in configuration (feature) model development and maintenance. We are aware of the fact that these metrics have to be further evaluated within the scope of user studies in industrial configuration environments with a potential result that more complex metrics are needed for improving the diagnosis prediction quality. However, results of the envisioned studies can be easily integrated in the proposed diagnosis approach in terms of additional search guidelines.

Although the knowledge base diagnosis problems analyzed in this paper fall into the class NP-complete and beyond [26], empirical studies show that the performance of conflict detection and diagnosis algorithms as the ones exploited in this paper is sufficient to support knowledge engineers in interactive settings [4]. Recent developments in the field focus on conflict-free (direct) hitting set determination that further boosts the efficiency of diagnosis search [21].

Two of the presented approaches to determine diagnosis relevance are based on additional user input (*rating-guided diagnosis* and *utility-guided diagnosis*). If applied in industrial settings, it could be the case that domain experts and knowledge engineers refuse to provide this additional information. However, approaches to include communities of users into knowledge engineering processes have already shown to be successful and have the potential to reduce related scalability problems [27].

5. RELATED WORK

A first approach to the determination of minimal sets of faulty constraints in a knowledge base based on model-based diagnosis was introduced by Bakker et al. [28]. This approach focuses on inconsistencies in knowledge bases. The work of Felfernig et al. [4] extends this approach by the integration of test cases that are used to induce inconsistencies in a (configuration) knowledge base. This approach is based on the assumption of the correctness of individual test cases. A similar approach to diagnosis-based knowledge base (feature model) debugging has been introduced by Benavides et al. [6]. Since erroneous changes do not only concern the knowledge base, but also test suites, automated testing and debugging approaches have also to take into consideration faulty test cases. The approach presented in this paper introduces definitions of a diagnosis problem and a corresponding diagnosis that allow the inclusion of test cases into a diagnosis and thus allow to consider them as first-class entities in diagnosis processes.

Kreuz et al. [22] introduce an approach to estimate knowl-

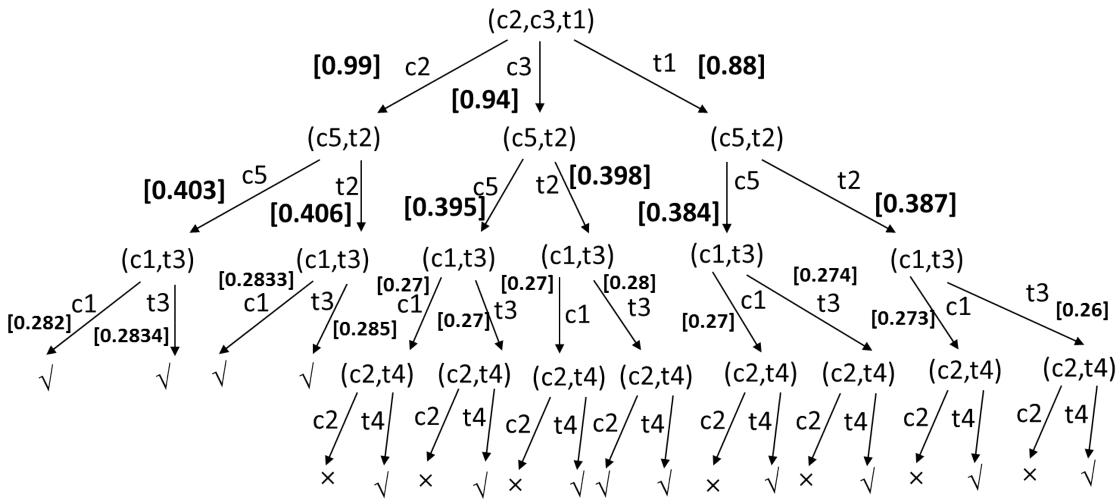

Figure 2: Hitting Set Directed Acyclic Graph (HSDAG) for example configuration knowledge base (and the corresponding test cases) – expansion strategy = actuality (ACT).

element	s_1	s_2	s_3	s_4	s_5	s_6	average rating
c_1	2.0	2.0	1.0	5.0	5.0	1.0	2.6
c_2	1.0	2.0	2.0	4.0	4.0	3.0	2.6
c_3	1.0	2.0	1.0	3.0	3.0	2.0	2.0
c_4	4.0	4.0	4.0	2.0	3.0	3.0	3.3
c_5	1.0	1.0	3.0	3.0	2.0	3.0	2.1
t_1	4.0	5.0	5.0	5.0	3.0	5.0	4.5
t_2	3.0	4.0	5.0	5.0	4.0	5.0	4.3
t_3	4.0	4.0	2.0	4.0	4.0	4.0	3.6
t_4	3.0	4.0	2.0	4.0	4.0	4.0	3.5

Table 4: *Rating* of individual knowledge base elements by the stakeholders $s_1, ..., s_6$.

diagnosis	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7	Δ_8	Δ_9	Δ_{10}	Δ_{11}	Δ_{12}	Δ_{13}
relevance	**0.137**	0.12	0.105	0.095	0.098	0.089	0.081	0.08	0.075	0.079	0.073	0.067	0.063
ranking	**1**	2	3	5	4	6	7	8	10	9	11	12	13

Table 5: Relevance (and corresponding ranking) of diagnoses based on *ratings*.

edge usefulness in configuration scenarios (e.g., how often a component was part of a configuration result). Our work extends this approach by introducing additional heuristics to estimate the relevance of knowledge chunks (constraints and test cases) with regard to change operations. Felfernig et al. [29] introduce an approach to the automated repair of scoring rules in knowledge-based recommendation scenarios. By focusing on a very specific constraint type (scoring rules) this approach is able to automatically determine repair actions for scoring rules and thus handle the ranking of recommendation results in an optimal fashion. The diagnosis approach presented in this paper is not restricted to specific constraint types.

Calvanese et al. [2] discuss issues related to the evolution of description logics based knowledge bases. They introduce different postulates to be taken into account by knowledge evolution processes (e.g., coherence preservation). Flouris et al. [1] provide a discussion of different types of knowledge evolution activities and discuss corresponding research contributions. Compared to the existing state of the art, we contribute a new approach to the handling of faulty test cases and heuristics that help to rank diagnosis candidates with regard to their relevance for change operations.

Beside the management of inconsistencies in knowledge bases, redundancies are of major concern due to the fact that they can trigger additional overheads related to knowledge understanding and maintainability. An overview of existing research on redundancy detection in knowledge bases can be found in [30]. Beside an overview of the existing state of the art in redundancy detection, the authors introduce a new approach to redundancy detection in distributed knowledge engineering scenarios. In contrast, our work focuses on the identification of the sources of a given inconsistency.

Cognitive complexity is a major source of faulty knowledge. Felfernig et al. [23] discuss basic organization principles

element	maintainability	understandability	efficiency	irredundancy	correctness	utility
c_1	4.0	4.0	1.0	5.0	5.0	3.8
c_2	2.0	2.0	5.0	4.0	4.0	3.4
c_3	1.0	2.0	4.0	3.0	3.0	2.6
c_4	4.0	4.0	4.0	3.0	3.0	3.6
c_5	1.0	1.0	2.0	3.0	2.0	1.8
t_1	4.0	5.0	5.0	5.0	3.0	4.4
t_2	3.0	4.0	5.0	5.0	4.0	4.2
t_3	4.0	4.0	2.0	4.0	4.0	3.6
t_4	3.0	4.0	2.0	4.0	4.0	3.4

Table 6: *Utility* (avg.) of individual knowledge base elements.

diagnosis	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7	Δ_8	Δ_9	Δ_{10}	Δ_{11}	Δ_{12}	Δ_{13}
relevance	0.111	**0.114**	0.088	0.089	0.086	0.088	0.071	0.072	0.072	0.075	0.0758	0.063	0.064
ranking	2	**1**	4	3	6	5	11	9	10	8	7	13	12

Table 7: Relevance (and corresponding ranking) of diagnoses based on *utilities*.

rank	1	2	3	4	5	6
utility	1.000k	100k	10k	1k	100	10

Table 8: Predefined utilities of individual rank positions, for example, a diagnosis ranked on the first position has the utilitiy 1.000.000, i.e., has a higher utility than the sum of all other utilities.

method/rank	1	2	3	4	5	6
BF	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6
ACT	Δ_4	Δ_3	Δ_2	Δ_1	Δ_8	Δ_7
RATE	Δ_1	Δ_2	Δ_3	Δ_5	Δ_4	Δ_6
UTILITY	Δ_2	Δ_1	Δ_4	Δ_3	Δ_6	Δ_5

Table 9: Ranking of diagnoses determined by the individual prediction mechanisms.

diagnosis	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6	Δ_7	Δ_8	Δ_9	Δ_{10}	Δ_{11}	Δ_{12}	Δ_{13}
relevance	**2.101k**	1.210k	121k	1.011k	1.110	120	10	100	0	0	0	0	0
ranking	**1**	2	4	3	5	6	8	7	9	10	11	12	13

Table 10: Relevance (and corresponding ranking) of diagnoses based on the *ensemble-based* approach. Note that Δ_9 .. Δ_{13} did not achieve a ranking among 1..6, therefore their relevance is 0.

(e.g., in which to represent constraints in implicative form) that have to be taken into account to improve the understandability of constraint-based knowledge representations. The work of Felfernig et al. [25] builds upon these results and presents further results related to the application of recommendation technologies (collaborative filtering and clustering approaches) for knowledge base organization. Specific well-formedness rules for knowledge bases (feature models) are introduced by Felfernig et al. [24] who show how to automatically detect and repair violations of well-formedness rules. An example of such a rule is the detection of variable values which can not occur in a configuration (result). Explanations for such rule violations are determined on the basis of the principles of model-based diagnosis. Finally, [31] show how to apply the basic principles of function point analysis in the context of configuration scenarios.

6. CONCLUSIONS

In this paper we presented a new approach to the automated diagnosis of knowledge bases such as configuration knowledge bases and feature models. In this context, diagnoses are indicators of potential sources of inconsistencies in knowledge bases. Our approach allows the consideration of test cases as first-class entities in diagnosis processes. Due to the fact that many diagnoses can be the outcome of a diagnosis process, mechanisms are provided that allow to rank the diagnoses with regard to the probability of being the source of the faulty behavior of the knowledge base (manifested in terms of inconsistencies between constraints in the knowledge base and test cases). Our future work will include the development of further diagnosis ranking heuristics and their evaluation within the scope of long-term user studies.

7. REFERENCES

[1] Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., Antoniou, G.: Ontology Change: Classification and Survey. Knowledge Engineering Review **23**(2) (2008) 117–152

[2] Calvanese, D., Kharlamov, E., Nutt, W., Zheleznyakov, D.: Evolution of DL-Lite knowledge

bases. In: ISWC 2010. Number 6496 in LNCS, Springer (2010) 112–128

[3] Soloway, E., Bachant, J., Jensen, K.: Assessing the Maintainabiliy of XCON-in-RIME: Coping with the Problem of very large Rule-bases. In: AAAI 1987, Seattle, Washington, USA (July 13–17 1987) 824–829

[4] Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M.: Consistency-based diagnosis of configuration knowledge bases. Artificial Intelligence **152**(2) (2004) 213 – 234

[5] Fleischanderl, G.: Suggestions from the software engineering practice for applying consistency-based diagnosis to configuration knowledge bases. In: DX-02, Semmering, Austria (2002) 33–35

[6] Trinidad, P., Benavides, D., Duran, A., Ruiz-Cortez, A., Toro, M.: Automated error analysis for the agilization of feature modeling. Journal of Systems and Software **81** (2008) 883–896

[7] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press (2003)

[8] Marques-Silva, J., Sakallah, K.: Grasp: a search algorithm for propositional satisfiability. IEEE Transactions on Computers **48**(5) (1999) 506–521

[9] Freuder, E.: In pursuit of the holy grail. Constraints **2**(1) (1997) 57–61

[10] Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1-2) (2002) 181–234

[11] Felfernig, A.: Standardized configuration knowledge representations as technological foundation for mass customization. IEEE Transactions on Engineering Management **54** (2007) 41–56

[12] Noy, N., Sintek, M., Decker, S., Crubezy, M., Fergerson, R., Musen, M.: Creating semantic web contents with protege-2000. IEEE Intelligent Systems **16**(2) (2001) 60–71

[13] Rabiser, R., Vierhauser, M., Lehofer, M., Grünbacher, P., Männistö, T.: Configuring and Generating Technical Documents, in: Knowledge-based Configuration – From Research to Business Cases, Elsevier, Morgan Kaufmann (2014) 241–250

[14] Benavides, D., Segura, S., Ruiz-Cortes, A.: Automated analysis of feature models 20 years later: A literature review. Information Systems **35** (2010) 615–636

[15] Hotz, L., Felfernig, A., Stumptner, M., Ryabokon, A., Bagley, C., Wolter, K.: Configuration Knowledge Representation and Reasoning, in: Knowledge-based Configuration – From Research to Business Cases, Elsevier, Morgan Kaufmann (2014) 41–72

[16] Myllärniemi, V., Tiihonen, J., Raatikainen, M., Felfernig, A.: Using answer set programming for feature model representation and configuration. In: Workshop on Configuration, Novi Sad, Serbia (2014) 1–8

[17] Benavides, D., Ruiz-Cortes, A., Trinidad, P.: Using constraint programming to reason on feature models. In: SEKE 2005, Taipei, Taiwan (2005) 677–682

[18] Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **32**(1) (1987) 57–95

[19] Junker, U.: QuickXPlain: Preferred Explanations and Relaxations for Over-constrained Problems. In: AAAI 2004. (2004) 167–172

[20] Felfernig, A., Schubert, M., Reiterer, S.: Personalized Diagnosis for Over-Constrained Problems. In: IJCAI 2013, Peking, China (2013) 1990–1996

[21] Felfernig, A., Schubert, M., Zehentner, C.: An efficient diagnosis algorithm for inconsistent constraint sets. AI for Engineering Design, Analysis, and Manufacturing (AIEDAM) **26**(1) (2012) 53–62

[22] Kreuz, I., Roller, D.: Knowledge growing old in reconfiguration context. (1999) 54–58

[23] Felfernig, A., Mandl, M., Pum, A., Schubert, M.: Empirical Knowledge Engineering: Cognitive Aspects in the Development of Constraint-Based Recommenders. In: IEA/AIE 2010, Cordoba, Spain (2010) 631–640

[24] Felfernig, A., Benavides, D., Galindo, J., Reinfrank, F.: Towards Anomaly Explanation in Feature Models. In: Workshop on Configuration, Vienna, Austria (2013) 117–124

[25] Felfernig, A., Reiterer, S., Stettinger, M., Reinfrank, F., Jeran, M., Ninaus, G.: Recommender Systems for Configuration Knowledge Engineering. In: Workshop on Configuration, Vienna, Austria (2013) 51–54

[26] Cook, S.: The complexity of theorem-proving procedures. In: STOC 1971. (1971) 151–158

[27] Felfernig, A., Haas, S., Ninaus, G., Schwarz, M., Ulz, T., Stettinger, M., Isak, K., Jeran, M., Reiterer, S.: Recturk: Constraint-based recommendation based on human computation. In: RecSys'2014 CrowdRec Workshop, Foster City, CA, USA (2014) 1–6

[28] Bakker, R., Dikker, F., Tempelman, F., Wogmim, P.: Diagnosing and solving over-determined constraint satisfaction problems. In: IJCAI'93, Chambery, France (1993) 276–281

[29] Felfernig, A., Schippel, S., Leitner, G., Reinfrank, F., Isak, K., Mandl, M., Blazek, P., Ninaus, G.: Automated Repair of Scoring Rules in Constraint-based Recommender Systems. AI Communications **26**(2) (2013) 15–27

[30] Felfernig, A., Reinfrank, F., Ninaus, G., Blazek, P.: Redundancy Detection in Configuration Knowledge. In Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J., eds.: Knowledge-based Configuration – From Research to Business Cases. Elsevier (2013)

[31] Felfernig, A., Salbrechter, A.: Applying Function Point Analysis to Effort Estimation in Configurator Development. In: PETO 2004, Kopenhagen, Denmark (2004) 109–119