Towards Understanding Cognitive Aspects of Configuration Knowledge Formalization

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

Knowledge about formal (semantic) interpretations of natural language domain descriptions is crucial for avoiding communication overheads between domain experts and knowledge engineers. In this paper we report preliminary results of an empirical study in which we investigated in which way natural language statements are formalized by knowledge engineers. We summarize the findings of our study and discuss aspects to be taken into account in order to avoid additional (often not needed) iterations in configuration knowledge engineering processes. This work is exploratory and intended to figure out open issues for future work.

Categories and Subject Descriptors

D.2 [**SOFTWARE ENGINEERING**]: Requirements/Specification—*Elicitation methods*

General Terms

Algorithms, Theory

Keywords

Variability Modeling, Knowledge Engineering, Cognitive Aspects, Empirical Studies

1. INTRODUCTION

Knowledge evolution [1, 2] can be characterized as the task of integrating new knowledge into a knowledge base (e.g., a configuration knowledge base or a feature model).

VaMoS '15 January 21 - 23 2015, Hildesheim, Germany, Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3273-6/15/01...$15.00 .
http://dx.doi.org/10.1145/2701319.2701327

In this context, the translation of informally described domain knowledge into a formal representation is a major source of faulty constraints and resulting redundancies and inconsistencies (see, for example, Felfernig et al. [3]).

The major goal of this paper is to analyze statements typically used for specifying configuration (variability) knowledge and to point out potential sources and reasons for erroneous translations (formalizations). For demonstration purposes, we focus on a constraint-based representation [4, 5] which is a wide-spread representation used for the specification of configuration models and feature models [6, 7, 8]. For a detailed discussion of formal configuration knowledge representations and their (dis)advantages, we refer to [9].

A neglected area of research are issues related to knowledge understandability and problems when transforming informally specified knowledge into a formal representation [10]. In this paper we (1) sketch typical ways in which configuration knowledge is translated into a corresponding formal representation and (2) present preliminary results of an empirical study which analyzes modeling errors in configuration knowledge engineering processes.

The remainder of this paper is organized as follows. In Section 2 we provide a short overview of concepts used for configuration knowledge representation. Thereafter, we present the results of an empirical study which focused on analyzing which types of errors can occur when translating informally defined configuration knowledge into a formal representation (Section 3). Threats to validity are discussed in Section 4. A discussion of related work is provided in Section 5. The paper is concluded with Section 6.

2. KNOWLEDGE REPRESENTATION

Often used concepts for formalizing the properties of a configurable product are introduced in this section. First, we introduce a formal definition of a configuration task and a corresponding configuration (solution).

Definition: Configuration Task. A configuration task can be defined as a constraint satisfaction problem (V, D, C) [4, 5] where $V = \{v_1, v_2, ..., v_n\}$ is a set of finite domain variables, $D = \{dom(v_1), dom(v_2), ..., dom(v_n)\}$ represents a set of corresponding variable domains, and $C = C_{KB} \cup REQ$ represents a set of constraints. C_{KB} is the configuration knowledge base (the configuration model) and REQ represents a set of user (customer) requirements.

Definition: Configuration. A configuration (solution) S for a given configuration task (V, D, $C_{KB} \cup REQ$) is represented by an assignment $S = \{ins(v_1), ins(v_2), ..., ins(v_n)\}$ where $ins(v_i) \in dom(v_i)$ and S is complete and consistent with the constraints in $C_{KB} \cup REQ$.

On the basis of these definitions, we now discuss different modeling concepts that can be used to express configuration knowledge on a formal level. Note that we only discuss concepts investigated in our empirical study, for a detailed overview of modeling concepts for representing variability knowledge we refer to [9, 11].

Incompatibilities and Compatibilities. Given two finite domain variables v_a and v_b with the corresponding variable domains dom(v_a) = {a1,a2,a3} and dom(v_b) = {b1,b2,b3}, an *incompatibility* constraint states that specific values of v_a must not be combined with specific values of v_b. An example of such an incompatibility is *incomp(v_a(a1),v_b(b2))* which states that the values *a1 and b2* of the corresponding variables must not be combined in the same configuration.[1] In contrast, *compatibility constraints* specify possible combinations of specific variable values. For example, *comp(v_a(a1),v_b(b1))* states that the combination of v_a=*a1*, v_b=*b1* is one (among several) allowed combination(s) in the set of possible configurations. In this example, no other combination is allowed to be part of a configuration result. Alternative representations and nuances in semantic interpretations will be discussed in Section 3. In our empirical study we were interested in the following general questions related to (in)compatibilities.

- Q1: is the relationship between compatibility and incompatibility specifications clear for knowledge engineers and what are major aspects to be taken into account when specifying such constraints?
- Q2: what are preferred formalizations of (in)compatibilities and which modeling traps exist?

Requires. Given two finite domain variables v_a and v_b with the corresponding variable domains dom(v_a) = {a1,a2,a3} and dom(v_b) = {b1,b2,b3}, a requires constraint states that given specific values of v_a, the domain of v_b has to be restricted in a certain fashion. An example of such a requires relationship is *requires(v_a(a1),v_b({b2,b3}))* which states that if v_a has the value *a*1, v_b must be instantiated with values out of {b2,b3}. In our empirical study we were interested in the following related questions.

- Q3: what are major aspects to be taken into account when specifying requires relationships?
- Q4: what are preferred formalizations of requires relationships and which modeling traps exist?

[1] In the area of software product lines [12], incompatibilities are expressed in terms of *excludes* relationships (which have the same semantics as incompatibilities).

Additional Issues. Beside basic (in)compatibility and requirements relationships, we were interested in the question to which extent knowledge engineers are able to detect redundancies in constraint sets and what are the related efforts (Q5). Finally, we investigated how existing configuration knowledge bases are translated into a corresponding textual representation (Q6). The reason for this question was to attain an overview of variations in semantic interpretations.

3. EMPIRICAL STUDY

For the purposes of our empirical study, we introduced variables and corresponding domain definitions from the product domain of financial services. These variables serve as a basis for the inclusion of additional constraints. Additional constraints are specified on an informal level and the task of the subjects of our study was to select a corresponding formal semantics. This way, we are able to get an impression of how natural language based configuration knowledge representations are typically interpreted by knowledge engineers. The variables and domains defined for our financial services configuration knowledge base are the following.

V = {willingness to take risks (wr), return rate (rr), investment period(ip)}.
$dom(wr) = dom(rr)$ = {low, medium, high}.
$dom(ip)$={shortterm, mediumterm, longterm}.

In our study, N=10 knowledge engineers (also denoted as "subjects") with expertise in the development and maintenance of configuration knowledge bases were confronted with knowledge engineering tasks (all subjects were male). The subjects successfully completed their studies of Computer Science (at two Austrian universities) and had 2-5 years of experiences in the development of constraint-based applications. 50% of the subjects already had experiences in the development of commercial configuration and recommendation knowledge bases. All the subjects had both, experiences in the textual specification of constraints (e.g., when using constraint solving libraries) and in the graphical specification of constraints (feature models and tabular-based configuration knowledge representations). Subjects with industrial background were recruited from two companies providing configuration and recommendation technologies.

The knowledge engineering tasks and related results of the study are discussed in the following paragraphs. In this context, $\surd$ means "correct alternative" and $\times$ "not a correct alternative" for the given task. Clearly, this information is only provided for improving the readability of the paper but has not been disclosed to the study participants. In the following, we describe each task and then discuss (on an informal level) typical modeling problems that occurred.

Task 1 – Interdependencies between Compatibilities and Incompatibilities. Subjects were confronted with the following informal statement and related alternative formalizations. The task was to select appropriate knowledge representations (see Table 1) and to explicitly define criteria to be taken into account when specifying (in)compatibilities.

In our example (Table 1), Alternatives 1 and 2 are correct representations. Alternative 3 should not be selected since a combination of compatibilities and incompatibilities leaves room for semantic interpretations (what is the semantics of a value combination that is not represented as compatibility and also not represented as incompatibility?). Therefore, Alternative 3 is not a reasonable option.

Natural Language Statement (s)	*Appropriate Formalization?*
A low willingness to take risks is incompatible with a high expected return rate.	1) {incompatible(rr=high, wr=low)}. $\surd$
	2) {compatible(rr=high, wr=medium), compatible(rr=high, wr=high), ..., compatible(rr=low, wr=high).} $\surd$
	3) combination of compatible and incompatible statements. $\times$

Table 1: Interdependencies between Compatibilities and Incompatibilities.

In Task 1, each of the subjects selected correct formalizations (only one subject additionally selected the 3rd alternative). An important criterion (specified by the subjects) when formalizing natural language domain descriptions seems to be that formalizations should "intuitively" reflect the natural language text. Not having an "intuitive" translation triggers additional cognitive efforts. One important criteria regarding the choice between compatibility and incompatibility constraints is the following: if a majority of value combinations is compatible, the (minority) incompatibilities should be specified (and vice-versa).

Task 2 – Requires: Same LHS. Subjects were confronted with the following statement and related formalizations (see Table 2). The informal statement can be formalized by Alternative 1 (RHS with $\wedge$-connected conditions) or Alternative 3 (two separate requires conditions).

Natural Language Statement (s)	*Appropriate Formalization?*
A high expected return rate requires a high willingness to take risks. A high expected return rate requires a longterm investment period.	1) rr=high $\rightarrow$ (wr = high $\wedge$ ip = longterm). $\surd$
	2) rr=high $\rightarrow$ (wr = high $\vee$ ip = longterm). $\times$
	3) rr=high $\rightarrow$ wr = high. rr=high $\rightarrow$ ip = longterm. $\surd$

Table 2: Requires: Same LHS.

The results related to Task 2 are the following. There was no preferred formalization. The selected formalizations were equally distributed between Alternative 1 and Alternative 3. No participant selected both correct alternatives. Alternative 1 seems to be a more compact representation, however, further experiments are needed to figure out which type of representation has a higher degree of maintainability.

Task 3 – Requires: Same RHS. Subjects were confronted with the following informal statement and related alternative formalizations (see Table 3). The informal statement can be formalized as Alternative 1 (two separate conditions) or Alternative 2 (LHS with $\vee$-connected conditions). Alternative 3 is not valid.

Natural Language Statement (s)	*Appropriate Formalization?*
A shortterm investment period as well as a high expected return rate require a high willingness to take risks.	1) ip=shortterm $\rightarrow$ wr = high. rr=high $\rightarrow$ wr = high. $\surd$
	2) ip=shortterm $\vee$ rr=high $\rightarrow$ wr = high. $\surd$
	3) (ip = shortterm $\rightarrow$ wr = high) $\vee$ (rr = high $\rightarrow$ wr = high). $\times$

Table 3: Requires: Same RHS.

In Task 3, subjects had no preferred formalization and the selected formalizations were equally distributed between Alternative 1 and Alternative 2 (only one subject additionally chose the 3rd alternative). The majority of subjects did not choose both alternatives. Alternative 2 has a slightly more direct translation, however, further studies are needed to analyze in more detail the relative advantages of the representation alternatives.

Task 4 – Requires: OR in RHS. Subjects were confronted with the following informal statement and related alternative formalizations (see Table 4). The informal statement can be formalized by Alternative 2 or Alternative 3. Alternative 1 and Alternative 4 are not valid.

Natural Language Statement (s)	*Appropriate Formalization?*
A high expected return rate requires a longterm investment period or a high willingness to take risks.	1) rr = high $\rightarrow$ ip = longterm. rr = high $\rightarrow$ wr = high. $\times$
	2) rr = high $\rightarrow$ ip = longterm $\vee$ wr = high. $\surd$
	3) (rr = high $\rightarrow$ ip = longterm) $\vee$ (rr = high $\rightarrow$ wr = high). $\surd$
	4) (rr = high $\rightarrow$ ip = longterm) $\wedge$ (rr = high $\rightarrow$ wr = high). $\times$

Table 4: Requires: OR in RHS.

The results related to Task 4 are the following. The majority of subjects (80%) selected Alternative 2. Two subjects also selected (the faulty) Alternatives 1 and 4. Alternative 2 seems to be a "direct" translation and also be the more "compact" one compared to Alternative 3.

Task 5 – Detectability of Redundancies. This task (see Table 5) was defined in terms of a knowledge base (represented as CSP) with redundant constraints, i.e., constraints that are in some cases not needed for expressing the intended semantics. More formally, a constraint $c_i \in C$ is redundant if $C - \{c_i\} \models c_i$. In this context, examples of redundant constraints are c_0 and c_5 since both are covered by c_9.

Variables and Domains	*Redundant Constraints?*
V = $\{v_1, ..., v_{10}\}$ dom(v_i = {1,2,3})	$c_0 : v_8 \neq v_9$. ×
	$c_1 : v_9 \neq v_{10}$. √
	$c_2 : v_2 \neq v_9$. √
	$c_3 : v_3 = 1$. √
	$c_4 : v_{10} \neq v_8$. √
	$c_5 : v_9 \neq v_8$. ×
	$c_6 : v_4 > v_2$. √
	$c_7 : v_2 \neq v_{10}$. √
	$c_8 : v_8 \geq v_7$. √
	$c_9 : v_9 > v_8$. √
	$c_{10} : v_2 \neq v_8$. √

Table 5: Detectability of Redundancies.

In Task 5, there were no majorities regarding the selection of the correct alternative(s). However, subjects who chose a faulty alternative invested less time for analyzing the problem (avg. 113.0 sec. vs. 153.3 sec.). This result shows that an automated redundancy detection mechanism is needed since redundant constraints could not be identified correctly even in small settings (10 variables and 10 constraints).

Task 6 – Expressing Compatibility Properties. Subjects were confronted with the following informal statement and related alternative formalizations (see Table 6). The informal statement can be formalized by Alternative 4 – all other alternatives are not valid since the compatibility is expressed in one direction (information about the other direction is not available).

Natural Language Statement (s)	*Appropriate Formalization?*
A high expected return rate is only compatible with a high willingness to take risks.	1) rr = high ↔ wr = high. ×
	2) rr = high ∧ wr = high. ×
	3) rr = wr. ×
	4) rr = high → wr = high. √

Table 6: Expressing Compatibility Properties.

The results related to Task 6 are the following. The semantics of compatibilities seems to be unclear in this context. The term "A is only compatible with B" implies a direction of the compatibility, i.e., A is only compatible with B but this does not mean that B is only compatible with A. However, an equivalence is often assumed by knowledge engineers and in many cases (60%) Alternative 1 has been selected as correct solution. In order to avoid such situations, the other direction should be specified as well (e.g., "B is only compatible with A"). An insight in this context is that "directed" incompatibilities should rather be specified as "requires" constraints. Otherwise, potential misinterpretations of knowledge engineers could occur.

Task 7 – Preferred Incompatibility Representations. Subjects were confronted with the following informal statement and related alternative formalizations (see Table 7). The informal statement can be formalized by both alternatives.

Natural Language Statement (s)	*Appropriate Formalization?*
A shortterm investment period is incompatible with a high expected return rate.	1) $\neg(ip = shortterm \wedge rr = high)$. √
	2) $ip = shortterm \rightarrow \neg rr = high$. √

Table 7: Preferred Incompatibility Representations.

The results related to Task 7 are the following. There is no clear "winner" but most of the subjects did not select both alternatives. In Task 7, the second alternative seems to be more or less a 1:1 representation of the natural language specification whereas Alternative 1 is a wide-spread representation of incompatibilities in the configuration context.

Task 8 – Combined Properties in Requires Relations. Subjects were confronted with the following informal statement and related alternative formalizations (see Table 8). The statement can be formalized by Alternative 3.

Natural Language Statement (s)	*Appropriate Formalization?*
A return rate which is not low and not medium requires an investment period which is not shortterm and not mediumterm.	1) rr ≠ low → ip ≠ shortterm ∧ ip ≠ *mediumterm*. rr ≠ medium → ip ≠ shortterm ∧ ip ≠ *mediumterm*. ×
	2) rr = high ↔ ip = longterm. ×
	3) ¬ ip = longterm → ¬ rr = high. √

Table 8: Combined Properties in Requires Relations.

The results related to Task 8 are the following. All subjects chose Alternatives 1 and 2 as the correct ones. This is a direct consequence of the complex textual description which has easier to understand alternative representations, for example, *a high expected return rate requires a longterm investment period*. The complexity of the textual description can be explained by the explicit representation of irrelevant settings (i.e., *not low*, *not medium*, *not shortterm*, and *not mediumterm*) instead of directly specifying the relevant (allowed) ones. There are parallels to the selection between compatibility and incompatibility relationships (see Task 1) since incompatibilities should only be specified if their number is low. Alternative 3 has not been identified as correct solution – a major reason is also that there is no direct correspondence between the formalization and the corresponding textual representation.

Task 9 – Alternative Incompatibility Formulation. Subjects were confronted with the following informal statement and related alternative formalizations (see Table 9). The statement should be formalized by Alternative 1.

Natural Language Statement (s)	*Appropriate Formalization?*
A shortterm investment period is only compatible with a low willingness to take risks (and vice-versa).	1) ip = shortterm $\leftrightarrow$ wr = low. $\surd$
	2) ip = shortterm $\wedge$ wr = low. $\times$
	3) ip = shortterm $\rightarrow$ wr = low. $\times$

Table 9: Alternative Incompatibility Formulation.

The results related to Task 9 are the following. The majority of subjects (90%) chose the correct alternative (1). In contrast to Task 6, the semantics of the compatibility is clear due to the fact that both directions are clearly specified.

Task 10 – Textual Representation of Configuration Models. Subjects were confronted with the configuration (feature) model depicted in Figure 1. This model had to be described on a textual level. The formal semantics of feature models is shown in Table 10 (see also [9]). The result related to Task 10 is the following. A major issue with the textual explanations of the formalization of Figure 1 was the *description of alternatives*. In 90% of the textual descriptions no clear "or" semantics could be identified, i.e., the "or" was mentioned in the text but no further information regarding the type of "or" relationship could be identified. As a consequence, alternatives have to be described in a more concise fashion (e.g., in terms of "either or") in order to avoid unintended formalizations.

4. THREATS TO VALIDITY

We are aware of the fact that the sample size of our initial user study is rather small. However, our goal was to gain initial insights to where semantic interpretations of natural language domain descriptions can become problematic. The results of this initial study are an important input for future work which will include empirical studies with larger user communities. A major focus of our user study were different types of compatibilities and incompatibilities. Other modeling concepts such as generalization hierarchies, partonomies, and hierarchies of optional and mandatory relationships (in feature models) have not been taken into account which limits the study results to specific constraint types. Additional modeling concepts and constraints will be a major focus of our future studies. We also want to emphasize that the examples used in the user study are solely based on the authors' experiences from industry projects but do not claim to be complete, i.e., a major focus of our future work will be a more in-depth analysis of the way domain experts specify product domain knowledge and knowledge engineers interpret these formulations. This will require an in-depth analysis of industrial product domain descriptions as well as a more in-depth study design that allows a more detailed answering of questions such as *In which way do domain experts specify their domain knowledge and what are major sources of incorrect formalizations of knowledge engineers?*, *In which way should domain experts specify product domain knowledge such that the number of faulty formalizations can be minimized?*, and *How do knowledge engineers describe specific formalizations and what are major sources of misunderstandings by domain experts?* In this context we will also take into account approaches to ambiguity handling which already play an important role, for example, in the context of software requirements engineering [13].

Relationship/Constraint	*Semantics*
mandatory(P,C)	$P \leftrightarrow C$
optional(P,C)	$C \rightarrow P$
or(P,$C_1, C_2, ..., C_n$)	$P \leftrightarrow (C_1 \vee C_2 \vee ... \vee C_n)$
alternative (P,$C_1, C_2, ..., C_n$)	$(C_1 \leftrightarrow (\neg C_2 \wedge ... \wedge \neg C_n \wedge P)) \wedge (C_2 \leftrightarrow (\neg C_1 \wedge ... \wedge \neg C_n \wedge P)) \wedge ... \wedge (C_n \leftrightarrow (\neg C_1 \wedge ... \wedge \neg C_{n-1} ... \wedge P))$
requires(P,C)	$P \rightarrow C$
excludes(P,C)	$\neg P \vee \neg C$

Table 10: Semantics of feature model concepts (P, C, and C_i represent individual features).

5. RELATED WORK

Knowledge understandability is a crucial factor for successful knowledge evolution [14, 15, 16]. Baumeister et al. [17] discuss different types of refactoring methods for knowledge bases. For example, if a question or a variable value in a knowledge base is never used, these items can be proposed for deletion. Other examples of refactoring operations are the simplification of questions by reducing the set of possible answers, completely deleting irrelevant questions which do not reduce the solution space, and integrating rules with the same right hand side. In the same line, Gil and Tallis [18] show how to support change operations by providing explicit rules that help to assure the correctness of knowledge base change operations. Mehrotra et al. [19] show the application of a clustering algorithm in the context of understanding axioms in ontologies. Identified clusters are visualized and

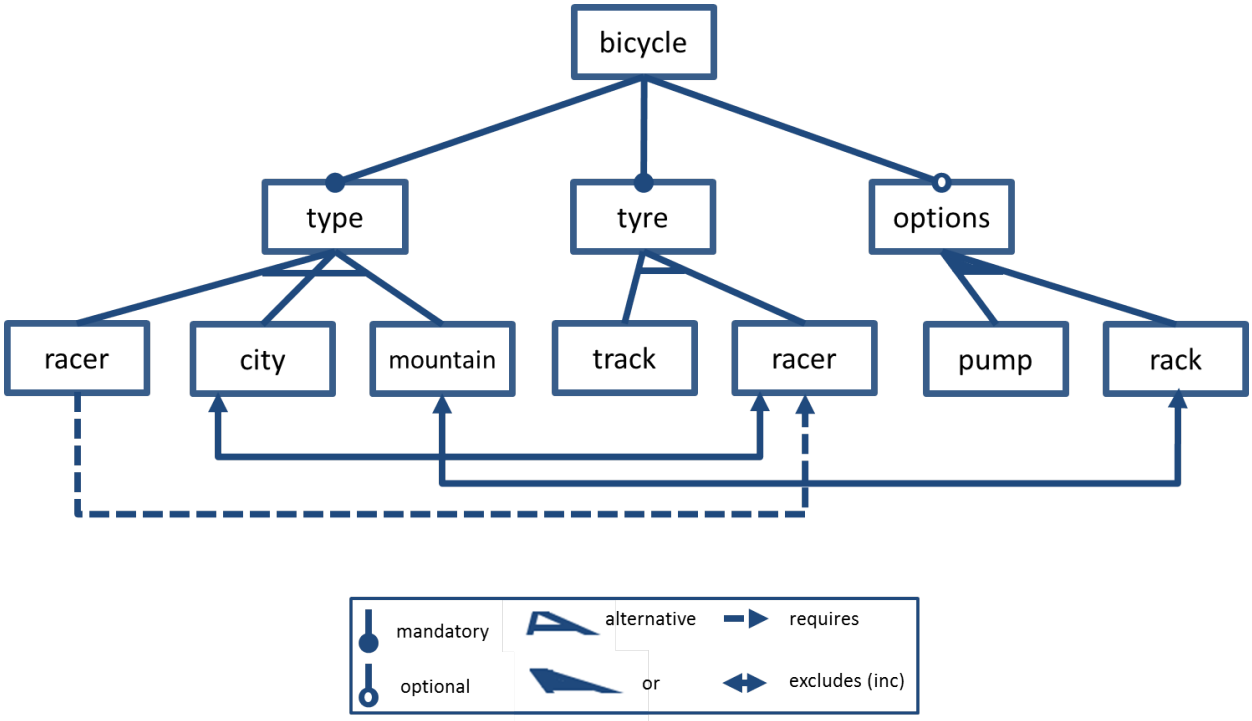

Figure 1: Configuration (Feature) Model used in the empirical study (participants had to explain the feature model on a textual (informal) level).

can be inspected by knowledge engineers. Felfernig et al. [10] present the results of an empirical study dedicated to the analysis of easy to understand knowledge patterns. The authors discuss a couple of rules that should be taken into account when building knowledge bases (using a constraint-based representation). Examples are the correct usage of relational expressions and implications. On the basis of this work, Felfernig et al. [20] analyzed the understandability of further knowledge patterns (e.g., the understandability of different types of requires and compatibility representations) and showed the applicability of recommendation technologies (collaborative filtering and clustering algorithms) as a means to improve knowledge understandability. The authors of [21] introduce concepts that allow the automation of knowledge base testing and debugging on the basis of model-based diagnosis [22]. Felfernig et al. [23] extend this approach by showing how to identify leading diagnoses, i.e., diagnoses with a high probability of being the source of a given inconsistency beetween a test suite and a (configuration) knowledge base. In the line of this work, Felfernig et al. [24] introduce concepts for identifying redundant constraints in knowledge bases, i.e., constraints that do not change the semantics (set of possible solutions) of a knowledge base. Felfernig et al. [25] integrate previous research results presented in [21, 20] by showing how to apply conflict [26] and redundancy detection algorithms [24] for the detection of well-formedness violations. A well-formedness rule is a kind of generalized test case that describes (un)intended structures in knowledge bases. For example, if a variable domain value can not be part of any configuration, it does not make sense to include this value in the variable domain. As an alternative, Felfernig et al. [25] show how to explain such situations in terms of an indication which model elements have to be changed in such a way that all well-formedness rules are fulfilled.

6. CONCLUSIONS

In this paper, we presented the results of an empirical study which focused on modeling issues in the context of configuration model development. We provided an overview of semantic interpretations of natural language descriptions typically applied by knowledge engineers and discussed basic misunderstandings that can occur in such modeling contexts. Related examples are faulty translations of directed incompatibilities (also known as *requires* constraints), no "direct" correspondence between textual domain knowledge representations and correspnding formalizations, and underspecified or-relationships making a concise formalization infeasible.

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. Volume 6496 of LNCS., Springer (2010) 112–128

[3] Felfernig, A., Friedrich, G.E., Jannach, D.: UML as Domain Specific Language for the Construction of Knowledge-based Configuration Systems. International Journal of Software Engineering and Knowledge Engineering **10**(4) (2000) 449–469

[4] Mackworth, A.: Consistency in Networks of Relations. AI Journal **8** (1977) 99–118
[5] Freuder, E.: In pursuit of the holy grail. Constraints **2**(1) (1997) 57–61
[6] Benavides, D., Ruiz-Cortes, A., Trinidad, P.: Using constraint programming to reason on feature models. In: 17th International Conference on Software Engineering and Knowledge Engineering, Taipei, Taiwan (2005) 677–682
[7] Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J.: Knowledge-based Configuration – From Research to Business Cases. Elsevier, Morgan Kaufmann (2014)
[8] 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
[9] Hotz, L., Felfernig, A., Stumptner, M., Ryabokon, A., Bagley, C., Wolter, K.: Configuration Knowledge Representation and Reasoning. In Felfernig, A., Hotz, L., Bagley, C., Tiihonen, J., eds.: Knowledge-based Configuration – From Research to Business Cases. Elsevier (2013)
[10] Felfernig, A., Mandl, M., Pum, A., Schubert, M.: Empirical knowledge engineering: Cognitive aspects in the development of constraint-based recommenders. In: 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2010), Cordoba, Spain (2010) 631–640
[11] Benavides, D., Segura, S., Ruiz-Cortes, A.: Automated analysis of feature models 20 years later: A literature review. Information Systems **35** (2010) 615–636
[12] Kang, K., Cohen, S., Hess, J., Novak, W., Peterson, S.: Feature-Oriented Domain Analysis Feasibility Study. Tech. Rep. CMU/SEI-90-TR-021 (1990)
[13] Berry, D., Kamsties, E., Krieger, M.: From Contract Drafting to Software Specification: Linguistic Sources of Ambiguity. University of Waterloo (2003)
[14] Bachant, J., McDermott, J.: R1 Revisited: Four Years in the Trenches. AI Magazine **5**(3) (1984) 21–32
[15] Soloway, E., Bachant, J., Jensen, K.: Assessing the Maintainabiliy of XCON-in-RIME: Coping with the Problem of very large Rule-bases. In: Proc. of AAAI-87, Seattle, Washington, USA (July 13–17 1987) 824–829
[16] Barker, V., O'Connor, D., Bachant, J., Soloway, E.: Expert systems for configuration at Digital: XCON and beyond. Communications of the ACM **32**(3) (1989) 298–318
[17] Baumeister, J., Puppe, F., Seipel, D.: Refactoring methods for knowledge bases. In: EKAW 2004. Volume 3257 of LNAI., Springer (2004) 157–171
[18] Gil, Y., Tallis, M.: A script-based approach to modifying knowledge bases. In: AAAI/IAAI 1997, Providence, Rhode Island (1997) 377–383
[19] Mehrotra, M., Bobrovnikoff, D., Chaudhri, V., Hayes, P.: A clustering approach for knowledge base analysis. In: 18th National Conference on Artificial Intelligence. (2002)
[20] 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
[21] Felfernig, A., Friedrich, G., Jannach, D., Stumptner, M.: Consistency-based diagnosis of configuration knowledge bases. Artificial Intelligence **152**(2) (2004) 213 – 234
[22] Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence **32**(1) (1987) 57–95
[23] Felfernig, A., Reiterer, S., Stettinger, M., Tiihonen, J.: Intelligent techniques for configuration knowledge evolution. In: VAMOS 2014, Hildesheim, Germany (2014)
[24] Felfernig, A., Zehentner, C., Blazek, P.: Corediag: Eliminating redundancy in constraint sets. In: 22nd International Workshop on Principles of Diagnosis, Murnau, Germany (2011) 219–224
[25] Felfernig, A., Benavides, D., Galindo, J., Reinfrank, F.: Towards Anomaly Explanation in Feature Models. In: Workshop on Configuration, Vienna, Austria (2013) 117–124
[26] Junker, U.: QuickXPlain: preferred explanations and relaxations for over-constrained problems. In: Proceedings of the 19th National Conference on Artifical Intelligence. AAAI 2004, AAAI (2004) 167–172